Talking to Ceptre: A Natural Language Interface

by

Samantha Ong

A thesis submitted to the
faculty of Wesleyan University
in partial fulfillment of the requirements for the
Degree of Bachelor of Arts
with Departmental Honors from the College of Letters
and with Departmental Honors in Computer Science

Middletown, Connecticut  April 2018
Abstract

A natural language interface provides a simple way of communicating, by means of everyday language, with a computer. When you say “Hey Alexa”, you are interfacing with the program code which controls Alexa's functionality by means of natural language. This is broadly what we wish to do in this thesis — to develop a method of interfacing with computers that is not dependent on one's knowledge of programming. Our interface supports the Ceptre programming language, developed by Chris Martens in the thesis “Programming Interactive Worlds with Linear Logic” [Martens, 2015] to allow one to formulate a narrative in terms of linear logic formulas by equating proof search with narrative description. Ceptre tackles problems surrounding the design of interactive worlds underlying narrative and games, and focuses on formalizing the notions of narrative generativity and causality. Such interactive worlds are shown to lend themselves to formal modeling by means of Jean-Yves Girard’s Linear Logic, and the correspondence may be extended to narratives in general. By approaching the problem of interactive game-design from the perspective of programming language design, Martens creates a new abstraction by means of which narrative structure can be understood.

Our approach to the design of a natural language interface for Ceptre draws upon the correspondence architecture of LFG which establishes correspondences between different levels of linguistic representation. We will thus think of Ceptre program code as a level of linguistic representation similar to LFG c- and f-structures. This allows us to map from f-structures generated by LFG and a newly defined ζ-structure, sets of which can be composed to form a Ceptre program. Underlying our approach is the claim by Richard Montague that “there is...no important theoretical difference between natural languages and the artificial languages of logicians” [Montague, 1970], which allows us to treat natural languages the same way as we do formal languages. Thus, though
Ceptre is not a natural language, we can handle it in the same way as we do English. Our LFG f-structure gives us enough information to construct a semantic representation in the Ceptre programming language, which can then be used for inference as designed. The contribution of this thesis is thus this extension to the Ceptre language, a natural language interface which does not use statistical methods as is common in current day NLP research.
cum auxilio neminis hoc opus exactus est.
creator vult lectori gratias agere.

FOR MY FRIENDS AND FAMILY
Contents

Chapter 1. Introduction 1
   1. Computational Linguistics and Natural Language Processing 2
   1.1. The problem with statistical modeling 4
   2. Current Work 4

Chapter 2. A Historical Overview of Linguistic Theory 7
   1. Structuralism 8
   1.1. The Distributional Approach 11
   2. Chomsky’s Revolution 12
   3. Formal Semantics 17
   4. Current Approach 18

Chapter 3. Lexical Functional Grammar 22
   1. Formal Architecture 23
   1.1. Constituent Structure 23
   1.2. Functional Structure 23
   1.3. Correspondence Architecture 27
   1.4. Semantic Structures 30

Chapter 4. Linear Logic for Language Modeling 36
   1. Glue Semantics 36
   2. Linear Logic for Narrative Formalization 39

Chapter 5. Ceptre 42
   1. Ceptre as meaning-representation 44
CHAPTER 1

Introduction

Warren Weaver begins the project of computational linguistics with a memo written “with one hope only — that it might possibly serve in some small way as a stimulus to someone else, who would have the techniques, the knowledge, and the imagination to do something about it”. Inspired by the remarks of mathematician W.W. Reichenbach with regards to the shared features of basic logical structures across varying languages, Weaver gives birth to the idea of machine translation by putting forth a proposal for an approach to translation heavily influenced by his experience with wartime cryptography \[\text{Weaver, 1949}\].

Weaver’s approach to the linguistic problem of translation was novel; he was applying mathematical techniques directly to the field of linguistics, harnessing newly available computational powers. This interdisciplinary approach was furthered by Chomsky’s *Syntactic Structures* (1957) which provided an analysis of the expressiveness and suitability of formal grammars with regards to the syntactic relations of natural language, a crucial development for computational linguistics due to the correspondence between regular grammars and finite state machines. Syntax captured the semantic predicate-argument relation through the surface phrase configuration, defining the way sentences may be constructed; Grammar was realized as a deductive system that could be handled through computational methods. The computational implementation of formal grammars thus ensured consistency and allowed an analysis of complexity. Computational Linguistics thus became “an exercise in creating and implementing the formal systems that were increasingly seen as constituting the core of linguistic theory” \[\text{Kay, 2003}\]
1. Computational Linguistics and Natural Language Processing

In this thesis, we begin by differentiating the field of computational linguistics from natural language processing. The two terms are generally considered interchangeable, but we use computational linguistics to denote research into computational methods of processing natural language data with strong foundations in a given formal linguistic theory designed to allow algorithmic modeling. Natural Language Processing then refers to the data-driven approach taken from general machine learning methods. What it means for a machine to process text differs from application to application. In the task known as Named Entity Recognition, the general goal is for the system to be able to extract named entities from some natural language input. Thus, given some sentence

Joe went to Whole Foods on his way to NYC

we could require that the entities “Joe : person”, “Whole Foods : company” and “NYC : location” be identified. In other cases, such as in Text Summarization, we are given a large chunk of input text from which some shorter version, containing the same amount of information, is to be generated. Both distinct examples describe some problem in which the given input is some natural language sentence/s and the output is some form of information that is to be extracted or generated from the input. Broadly speaking, this is the shared concern of computational linguistics and NLP. Both research areas are interested in the way in which computational power can be harnessed to manipulate information given in the form of natural language. NLP, however, draws significantly more inspiration from statistical machine learning methods than computational linguistics. Thus, much research in NLP deals with the task of processing natural language by means of statistical techniques, often in conjunction with machine learning. Such techniques work by extracting some probability distribution from input data such as an annotated corpus from which future predictions can be made.

A common example of this is the n-gram language model, which models the probability distribution of a corpus in order to determine the most likely next item in
a sequence of \((n - 1)\) elements. That is, an n-gram model will calculate

\[
P(x_i|x_{i-(n-1)}, \ldots, x_{i-1})
\]

An NLP problem commonly tackled with the use of n-gram language models is speech recognition, which uses the n-gram language model to determine the most likely input utterance. Other applications of n-grams include text categorization \cite{Cavnar1994}, machine translation \cite{Marino2006}, and automatic language identification \cite{Zissman1994}. Such statistical language models allow problems in the domain of natural language to be rephrased in abstract terms.

An example of a general machine learning algorithm applied to the domain of natural language is **supervised learning**, which has been used by Pedersen and Bruce for the task of word-sense disambiguation \cite{Pedersen1997}. Supervised Learning involves “learning” some hypothesis function \(h\) from some input values. More formally, given some input values \(x^i\), the goal is to learn a function \(h(x)\) which approximates the target values \(y^i\). In **linear regression**, the target \(y^i\) is approximated as a linear function of input \(x^i\), where \(y^i\) is continuous valued. Given discrete target values to be approximated as a linear function of input values, the problem simplifies to **logistic regression**.

In the context of natural language processing, the supervised learning (SL) approach requires some feature-annotated corpus from which the model can be trained. The standard SL approach to named entity recognition essentially reads in a collection of annotated text from which a list of named entities can be extracted. Once entities have been identified from the training data, the algorithm then is able to approximate the discrete target values — entity types — by means of the learned hypothesis function. SL has also been used for word-sense disambiguation \cite{Pedersen1997} where the problem involves determining the most likely meaning of an ambiguous word by means of a new supervised learning algorithm which defines the probabilistic model used for classification from a sequence of models.
1. INTRODUCTION

1.1. The problem with statistical modeling. Both the n-gram language model and the supervised learning approach use a stochastic approach to linguistic phenomena, reflecting the trend towards statistical methods for linguistic problems. Though statistical models dominate in many areas of natural language applications, the most obvious criticism of such methods is that they do not accurately reflect the way in which humans process linguistic data. This claim is held by Chomsky, who argues that “one’s ability to produce and recognize grammatical utterances is not based on notions of statistical approximation and the like.” [Chomsky, 2002]. Thus, if we understand the central task of NLP and computational linguistics as that of recognizing or generating natural language, it appears that the success of statistical methods in NLP is not due to any fundamental relationship with language. It could be equally successful for some incoherent language, annotated in any which way. The central criticism thus lies in statistical NLP’s disregard for the intrinsic structure of natural language, the grammatical rules that are implicit in a native speaker’s processing of language. When a statistical NLP system offers up the word “rose” as the most likely next element in the sequence “The red...”, it is purely on the basis that “rose” is the most likely word to come next in the sequence given the corpus it was trained on. If the corpus it was trained on involved texts concerned with red sparrows, the system would offer up “sparrow” as the most likely word. This approach to linguistic analysis derives its foundation from structuralism, emerging most strongly in the work of Zellig Harris. We will present a brief overview of the path leading to the claim of the Distributional Hypothesis to understand its relation with this statistical approach. Chomsky’s refutation of the Distributional Hypothesis guides our approach to the integration of linguistic theory in the design of natural language interfaces.

2. Current Work

In this thesis, we will be considering an approach to the implementation of a natural language interface by drawing upon linguistic theories in the manner of computational linguistics which does not draw upon the distributional approach favored by data-driven
methods of natural language processing. We, like Chomsky, are unsatisfied that linguistic analysis can be achieved through statistical methods and thus return to a solid linguistic foundation for our approach. We have developed a tool which establishes a correspondence between natural language input and specified rules in the Ceptre programming language, inspired by the way Lexical Functional Grammar (LFG), a descendant of Chomsky’s Generative Grammar, allows different levels of linguistic information to be represented in distinct formal structures, related by piecewise correspondence. Our approach thus does not reflect current trends in NLP research which focus on shallow linguistic processing methods used in conjunction with machine learning; rather than relying on such statistical methods to deal with natural language, we return to the rules of language itself and depend on a theoretical linguistic framework to model it. In this way, this thesis identifies more with the field of computational linguistics rather than natural language processing.

We primarily rely on the correspondence architecture in LFG to motivate our approach to the design of a natural language interface by means of some linguistic theory. The correspondence architecture relates distinct levels of linguistic representation. The constituent structure organizes the input expression into phrases whereas the functional structure represents the abstract relations existing within lexical entries. These two structures are placed in correspondence, thus representing different types of information about the same input expression. The functional structure has the characteristic of being able to represent certain linguistic information universally, since the functional organization of languages is generally consistent. The LFG framework thus provides a way to move from the surface structure of a natural language expression to its functional structure which can be then compared with the functional structure derived from the surface structure of the same expression in a different language.

We will not specifically be dealing with translation across natural languages, but rather a process of translation into programming language in which the meaning of the natural language expression is preserved. That is, we want to extract the semantic content from our input to manipulate it in such a way that the same content is expressible
in programming language. The LFG framework introduces a third level of linguistic representation — the semantic structure — to allow for such semantic analysis. The approach to semantic structures covered in this thesis uses f-structures as the principle determinant of semantic composition, and this mirrors the path we take in our natural language interface. Specifically, we use this general approach of establishing structural correspondences between linguistic structures to map from f-structures to ζ-structures, a version of semantic structures. From a set of ζ-structures, we can build a Ceptre program.

We begin by considering the development of computational linguistics and current trends in order to highlight specific concerns surrounding the formal modeling of natural language. Then, we consider the Lexical Functional Grammar (LFG) framework as a formal system for grammatical representation, with an emphasis on the central idea of structural correspondence implicit within its architecture. The use of linear logic for semantic assembly is mirrored in narrative formalization as presented in [Bosser et al., 2010] where a narrative is represented by a sequent and narrative actions are modeled through linear logic implication. The correspondence between the PRED attribute of the f-structure representing the natural language expression denoting a specific narrative state and the representation of that same narrative state in Ceptre provides starting point for our design for a natural language interface, and thus we show how the syntax-semantics interface of Lexical Functional Grammar can be directly applied to the design of a natural language interface.
A Historical Overview of Linguistic Theory

The seduction of statistical NLP, despite Chomsky’s claim that “probabilistic models give no particular insight into some of the basic problems of syntactic structure” [Chomsky, 2002], is due to the legitimacy of its methodology, which is derived from linguistic structuralism. Research into computational methods for handling natural language began in the field of machine translation. After Warren Weaver’s manuscript of 1949 came the Georgetown Demonstration of 1954 which demonstrated the automatic translation of approximately sixty Russian sentences to English by means of only six grammar rules and an extremely limited lexicon of approximately 250 words. Considering the infancy of computer technology at this stage in history\(^1\), the idea that non-numerical data could similarly be processed by machines was daring. But the 1966 A.L.P.A.C report showed that progress in the field of machine translation was slower than expected, leading the U.S. government to cease funding research in the field for two decades. The report identified a need for basic research into computational linguistics, stating that

The state of linguistics is such that excellent research that has value in itself is essential if linguistics is to make [the aforementioned] contributions. Such research must make use of computers...But we do not yet have good, easily used, commonly known methods for having computers deal with language data. [Pierce and Carroll, 1966]

The need for formal linguistic theory to lend itself to algorithmic modeling, as identified in the A.L.P.A.C report, is a central concern of computational linguistics. The

---

\(^1\)Punch cards were still used to store digital information represented by the presence or absence of holes.
establishment of linguistics as a science is commonly attributed to Ferdinand de Saussure, whose posthumous publication of the *Course in General Linguistics* in 1916 was praised by American linguist Leonard Bloomfield (1887-1949) for “its clear and rigorous demonstration of fundamental principles” [Bloomfield and Hockett, 1970]. The school of computational linguistics is generally understood as being interested in the computational applications of formal linguistic theory, which was much influenced by Saussure’s understanding of the structure of language as being central to its study.

### 1. Structuralism

Saussure’s *Course in General Linguistics* is most famous for its presentation of the distinction between “langue” and “parole”, its understanding of language to be “the social side of speech...a sort of contract signed by the members of a community” [Saussure, 1959, p. 14], and its definition of language as “a system of signs in which the only essential thing is the union of meanings and sound-images, and in which both parts of the sign are psychological”. The linguistic unit is the product of an associative process in which concepts are associated with their linguistic representations as “sound-images” in the brain [Saussure, 1959, p.11]. This psychological correspondence is claimed to trigger a physiological response causing the production of sound-waves, modeling the way in which someone speaks. The linguistic sign is then understood by defining the sound-image as the *signifier*, and the concept as the *signified*; thus, the *sign* refers to the association of *signifier* and *signified* [Saussure, 1959, p.67]. The sign itself is arbitrary\(^2\) in nature; there is no rule which states definitively that the signifier “dog” is associated with the signified concept of a four-legged domestic pet which barks any more than there is a rule stating that “canine” signifies the same mental fact.

The place of meaning within Saussure’s linguistic science manifests itself in his discussion of values. Specifically, for Saussure, “language is only a system of pure \(^2\)Saussure emphasizes that by arbitrary he means that there exists no “natural connection” between the signifier and the signified, and in this way he understands language as “a system of pure values which are determined by nothing except the momentary arrangement of its terms” [Saussure, 1959, p.80]
values, it is enough to consider the two elements involved in its functioning: ideas and sounds” [Saussure, 1959, p.111]. Linguistics thus deals with the form produced by the combination of elements of sound and thought. The notion of linguistic value only exists when it is considered to be something which emanates from the form,

When [values] are said to correspond to concepts, it is understood that the concepts are purely differential and defined not by their positive content but negatively by their relations with the other terms of the system. Their most precise characteristic is in being what the others are not. [Saussure, 1959, p.117]

By defining concepts “not by their positive content but negatively by their relations with the other terms of the system”, Saussure focused linguistic study on form rather than content. Meaning, or “value”, if it exists, must derive from the structure, the set of relations within the system. It cannot be external to the structure of language. This was the general claim of structuralism, and it manifests itself in the work of Leonard Bloomfield who is credited with the establishment of American structuralism with the publication of Language in 1933. Bloomfieldian structuralism was the primary foundation for many linguistic theories popular until the 1950s, and it held that “linguistic study must always start from the phonetic form and not from the meaning”. Specifically, Bloomfield believed that

“Non linguists (unless they happen to be physicalists) constantly forget that a speaker is making noise, and credit him, instead, with the possession of impalpable ‘ideas’. It remains for linguists to show, in detail, that a speaker has no ‘ideas’, and that the noise is sufficient - for the speaker’s words act with a trigger-effect upon the nervous system of his speech-fellows” [Bloomfield, 1936]

In contrast to Saussure whose linguistic theory is dependent upon a mentalist framework with its consideration of “mental facts”, Bloomfield moves American linguistics away from considerations of meaning towards a structural analysis of language
based on taxonomy, minimizing its focus on semantic considerations. In *Language or Ideas* [Bloomfield, 1936], he holds that mentalistic terminology such as ‘consciousness’, ‘mind’, ‘perception’, and ‘ideas’ should be excised from the realm of linguistics. The argument stems from considerations of scientific methodology, a belief that “science can speak only in physical terms”, which results from the philosophical tension between perception and conception. This tension is highlighted with a passage from Karl Pearson’s *The Grammar of Science* from which Bloomfield identifies three different entities involved in the scientific method — actual ‘perceptual’ objects, speech-forms, and ‘ideas’. Thus, the scientific method as formulated concerns the move from perceptual objects, those which actually exist (i.e. Pearson’s perceptual straight line) and correspond to a speech-form (i.e. “straight-line”), to the “concept of a straight line”. It is this conceptual understanding of the speech-form, which Bloomfield rejects as “useless and confusing” for the task of linguistic theory, which is removed from linguistic science.

Bloomfield’s conception of linguistic theory makes no place for “mental facts”; indeed, his is an argument against mentalism, in which it is held that “speech-forms reflect unobservable, non-physical events in the minds of speakers and hearers; for every speech-form which is uttered, one only needs then to claim the occurrence of a corresponding mental event” [Bloomfield, 1943]. Meaning is defined as “the situation in which the speaker utters [a linguistic form] and the response it calls forth in its hearer” and thus is defined *situationally*, outside the domain of language. In this way, Bloomfield understands the study of meaning to be external to the study of language.

This was the structuralist approach to linguistics, “a scientific description of language in terms or relations between units, irrespective of any properties which may be displayed by these units but which are not relevant to the relations or deducible from the relations” [Hjelmslev, 1972]. What matters are the relations existing within a linguistic system, rather than specific linguistic properties encapsulated within the units associated by such relations - the semantic values. The structuralist approach focuses
on language itself, attempting to define those properties which can be correctly stated as part of the nature of language by looking at its structure.

1.1. The Distributional Approach. Bloomfieldian influences are inescapable in the work of Zellig Harris who claimed that “language can be described in terms of a distributional structure” i.e. in terms of the occurrence of parts (ultimately sounds) relative to other parts. Harris’ descriptive linguistics is offered as one which is “complete without intrusion of other features such as history or meaning” [Harris, 1954, p.146], and his central contribution to linguistic science came in the form of a “discovery method” for phonemes and morphemes deriving from his distributional conception of language. Harris carries Bloomfield’s antimentalism to its extreme in his methodology which understands linguistic meaning to stem from “the relations in which the elements of the structure take part” [Harris, 1968, p.2].

What guides Harris’ work is the Distributional Hypothesis, that a word is characterized by the company it keeps. Thus, to understand language one must look at “the company it keeps”; the “members of those classes which regularly occur together, and the order in which these classes occur” [Harris, 1954, p.146]. That there is “not only a body of facts about the regular occurrence of elements in a language but also a structure of relative occurrence (distribution)” [Harris, 1954, p.148] is the cornerstone on which Harris builds the field of distributional semantics, which conceives of semantics in terms of distribution. He argues that the claim stating that “language is not merely a bag of words but a tool with particular properties which have been fashioned in the course of its use” does not prevent the application of his approach to the problem of meaning in language; rather, he holds that

If we investigate in this light the areas where there are no simple distributional regularities, we will often find interesting distributional relations, relations which tell us something about the occurrence of elements and which correlate with some aspect of meaning.

In certain important cases, it will even prove possible to state certain
aspects of meaning as functions of measurable distributional relations.

[Harris, 1954, p. 156]

Harris’ approach to linguistic science draws upon Bloomfieldian structuralism in his claim that “the linguistic meanings which the structure carries can only be due to the relations in which the elements of the structure take part” [Harris, 1968, p.2] Thus, for Harris, meanings are functions of “measurable distributional relations” which make up the linguistic science.

We can understand Harris’ approach to be the methodological foundation upon which statistical natural language processing derives its legitimacy. By taking language as something that can be modeled distributionally, Harris sets the stage for large-scale linguistic processing which abstracts away from properties of language itself can still be applied to semantic problems. Our earlier example of word-sense disambiguation can be approached in this way: given a text corpus, we can analyze its distribution in order to extract words with similar meanings by approaching the notion of meaning similarity through the lens of the distribution hypothesis wherein similarity of meaning corresponds to similarity of distribution. Thus, words with similar distributions are considered to be similar in meaning. The distribution can thus be used in this way to differentiate between words senses by considering the context in which the word to be disambiguated appears.

The distributional hypothesis can be applied in this way to natural language due to its central assumption: that language is defined by difference; the linear nature of natural language with its sequence of word tokens or phonemes can only be “organized into discrete elements which have been defined on the basis of differences among the continuous phenomena” [Harris, 1968, p.6].

2. Chomsky’s Revolution

The structuralist school of thought held the project of linguistics to be descriptive in nature, and Chomsky understands that
[The fundamental assumption of structural linguistics] is that procedures of segmentation and classification, applied to data in a systematic way, can isolate and identify all types of elements that function in a particular language, along with the constraints that they obey. A catalogue of these elements, their relations, and their restrictions of ‘distribution’ would, in most structuralist views, constitute a full grammar of the language” [Chomsky, 1971, p.5]

He argues that the achievements of the structural linguists is methodological, rather than substantive - they establish that language can be studied in a formal way. Chomsky takes this methodological assumption and applies it to his construction of a grammatical theory known as generative grammar “that deals with the mechanisms of sentence construction, which establish a sound-meaning relation in this language” [Chomsky, 1971, p.6]. Heavily influenced by the distinction made in Universal Grammar between “deep structure” and “surface structure” [Chomsky, 1971, p.2], Chomsky also understands the aim of linguistic science in light of the aims of universal grammar, stating that “the primary interest of a correct grammar is that it provides the basis for substantiating or refuting a general theory of linguistic structure which establishes general principles concerning the form of grammar”. Deep structure is described to be “that aspect of the syntactic description that determines its semantic interpretation” while surface structure is “that aspect of the syntactic description that determines its phonetic form.” [Chomsky, 1971, p.12]. The “syntactic description”, is claimed to be a “neutral technical notion” defined to be “an (abstract) object of some sort, associated with the sentence, that uniquely determines its semantic interpretation...as well as its phonetic form” [Chomsky, 1971, p.10]. The key here is the association between an abstract object corresponding to the sentence and a “semantic interpretation” in light of a syntactic description.

The transformational aspect of generative grammar formally related distinct tree-structures, and we will briefly outline the way two distinct structures can be related
2. A HISTORICAL OVERVIEW OF LINGUISTIC THEORY

A phrase-structure grammar is represented by some schema of the form — i.e. $S \to NP \ VP$ and $NP \to (Det)N$ — allowing sentences to be parsed to some tree structure on the basis of their components’ syntactic role in the sentence as in 1a. These phrase-structure rules essentially demarcate syntactic constituents of a sentence, allowing us to view the expression in terms of its constituent phrases. Thus, we see that the rule $S \to NP \ VP$ says that the (S)entence is made up of a Noun Phrase (NP) and a Verb Phrase (VP). A (DET)erminer occurring to the left of a (N) forms an NP, whereas a VP is formed by a (V)erb and its complements. This example can be used to illustrate that English is not context-free. Under CFG rules, we may freely substitute any other verb for “saw” without consideration of context. Thus we should be able to use the verb “throw”. However, the sentence “the girl see the boy” is ungrammatical — “see” is substituted without reference to the fact that “girl” is a
singular noun. The phrase-structure rules must thus be context-sensitive to account for subject-predicate number agreement.

\[
\begin{align*}
S & \rightarrow NP \quad VP \\
NP & \rightarrow DET \quad N \\
VP & \rightarrow V \quad Prt \\
   & \rightarrow NP \\
   & \rightarrow DET \quad N \\
   & \rightarrow the \quad shoe \\
   & \rightarrow the \quad shoe
\end{align*}
\]

**Figure 2**

Context-sensitive phrase-structure rules are still unable to account for the synonymity of sentences 1 and 2. Consider the trees in Figure 2, the transformational aspect of generative grammar seeks to account for linguistic phenomena such as particle movement, as illustrated. Given the construction which consists of a (V)erb and a particle (PRT) as in “put down”, the particle can be shifted to the right of the NP to produce a synonymous sentence by developing a formal relationship between these trees\(^4\). The specifics of the transformational aspects of general grammar are complicated

\(^4\)The transformational framework developed in *Syntactic Structures* allows one phrase structure tree to be transformed into another given that the tree is able to be partitioned by the *structural index*, which is a variable denoting the structure required for the transformation to occur. This variable is related to a given transformation which is applied if the structure is present. This transformation is specified by a given *structural change* from which the resultant tree is derived.
and technical; what is important is the idea of establishing a way to move between forms of linguistic representation. Chomsky augments a phrase structure grammar with a notion of transformation between resultant tree-structures to reach a final surface structure output in order to define a formal system of rules reflective of his understanding of grammar as a generative device. A given grammar for a language is thus able to generate all grammatical sequences of that language. There will be no ungrammatical sentences generated since the grammar is simply a set of rules defining how to create elements in the language.

Chomsky’s criticism of linguists in the Bloomfieldian tradition lies in their focus on surface structure with willful neglect of deep structure. Bloomfield’s linguistics is a conscious rejection of mentalist philosophy, of Saussurean “mental facts”; meaning was considered external to the structure of language so that even when it is considered analyzable, as through the use of the distributional hypothesis, it is not central to the study of linguistics. This can be understood as a focus on syntax independent of semantics. For Chomsky’s conception of grammar, this was unacceptable; A grammar was to consist of a syntactic, semantic and phonological component; the syntactic component includes a base which generates deep structures which act as inputs to the purely interpretative semantic component in order to obtain a semantic interpretation and thus “the grammar assigns semantic interpretations to signals, this association being mediated by the recursive rules of the syntactic component” [Chomsky, 1971, p.67] In this way, Chomsky moved semantics back into the domain of linguistic science by highlighting the “striking correspondences between the structures and elements that are discovered in formal, grammatical analysis and specific semantic structures” [Chomsky, 1971, p.110] Generative grammar thus makes explicit the mechanism of linguistic processing by establishing the relation between signals and their semantic interpretations as a system of rules, which requires there to be “a general language-independent means for representing the signals and semantic interpretations that are interrelated by the grammars of particular languages.” [Chomsky, 1971, p.9]
3. Formal Semantics

Chomsky’s understanding of the syntax-semantics relationship led to the proposal for a semantic theory developed in the framework of generative grammar. Jerrold Katz and Jerry Fodor’s publication of “The Structure of a Semantic Theory” attempts a characterization of the form of such a theory, isolating the projection problem in their analysis of the domain of semantics [Katz and Fodor, 1963].

Katz and Fodor understand that

A synchronic description of a natural language seeks to determine what a fluent speaker knows about the structure of his language which enables him to use and understand its sentences...the speaker’s knowledge of his language take the form of rules which project the finite set of sentences he has fortuitously encountered to the infinite set of sentences of the language [Katz and Fodor, 1963, p.171] and define the projection problem to be that of formulating such rules. The proposed solution lies in considering the way in which a native speaker would understand a novel sentence, and at its core lies the key assumption of linguistic compositionality, implicit in the way Katz and Fodor understand the process of meaning assembly. Specifically, that

On the basis of his knowledge of the grammatical properties and the meanings of the morphemes of the language, the rules which the speaker knows enable him to determine the meaning of a novel sentence in terms of the manner in which the parts of the sentence are combined to form the whole. [Katz and Fodor, 1963]

The notion that linguistic knowledge was compositional in character derives from the Principle of Compositionality. The Principle is commonly attributed to Gottlob Frege (1848-1925), and consists of understanding the whole as a composition of its parts when taken in conjunction with Katz and Fodor’s claim of linguistic compositionality wherein
The meaning of a compound expression is a function of the meaning of its parts and the syntactic rule by which they are combined \[ \text{[Janssen, 2001]} \]

and can be generalized into the form stated by T.M.V. Janssen in which

The meaning of a compound expression is built up from the meanings of its parts \[ \text{[Janssen, 1986, p.2]} \]

The significant departure signaled in this approach to language as opposed to the structuralist tradition is in its consideration of “the parts” in its description of the whole. The principle assumes that the pieces making up a compound expression can a) be delimited into “pieces”, and b) have meaning in isolation, and c) be used to determine the meaning of the compound expression. This is the foundational assumption used by Richard Montague in his conception of Montague Grammar. Note the relationship established between syntax and semantics by the principle: syntax, the structure of language, makes the distinction between the whole and the parts; semantics, the meaning of language, is dependent on the meaning of the parts contributing to the meaning of the whole. This connection between syntax and semantics was hinted at in Chomsky’s understanding of the way deep structure determines semantic interpretation by means of syntactic description, though Chomsky never explicitly holds the relationship to be compositional in nature. The syntax-semantics interface can thus loosely be defined as the relationship between syntax and semantics, and Montague Grammar approaches it by way of compositionality. Though attributed to Frege, it never comes up explicitly in his writings and the formulation here is the principle in its general form as used in the linguistic theories we will be discussing for the remainder of this thesis.

4. Current Approach

The approach that makes up the rest of the text concerns the use of this fundamental principle of compositionality which underlies linguistic theory as a natural bridge between natural language and its application with regards to communicating with computers. Up to this point, we have highlighted the development of modern
linguistics beginning from its conception in Saussure’s *Course in General Linguistics*. Saussure’s structuralist approach to language influenced Bloomfield’s rejection of mentalist approaches, putting forth the idea that the theory of meaning is external to that of language. Meaning, for Bloomfield, is situational and concerns the response triggered by the experience of the linguistic form, an actual observable entity. The study of language thus should only concern that form reflecting the structure of language. Harris’s distributional approach methodologizes Bloomfield’s theory, yet provides room for meaning in linguistics by defining it in terms of the overall structure of language as expressed by its distribution. This can be seen to be the theoretical cornerstone of statistical NLP, as Harris’ approach to linguistic analysis finds itself uniquely suited to the probabilistic models derived from machine learning. We draw upon Chomsky’s denial of Harris’ claim that language can be studied in terms of linguistics in our approach which brings back a notion of semantic structure for natural language interface design analogous to Chomsky’s resurrection of deep structure in linguistic science. This connection made between the syntactic structure of a natural language expression and its semantic context constitutes the syntax-semantics interface, explicit reference to which is made in Richard Montague’s introduction to *Universal Grammar*, which provides a mathematical treatment of natural language. This formulation is developed upon his claim that

> There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory [Montague, 1970]

The implication of this claim is clear: mathematical logic can be applied to the syntax-semantics interface, thus allowing for its use for the treatment of natural language. We can thus use some “mathematically precise theory” to comprehend the universal syntax and semantics of language, whether it be natural language or “the artificial languages
of logicians”. Montague defines meaning as “entities that serve as interpretations of expressions” [Montague, 1970, p.379], and this understanding of meaning as related to interpretation motivates our approach to natural language within a computational framework.

We will be considering a problem that lies at the intersection of linguistics and computer science — that of designing natural language interfaces. This can be more generally understood to be that of allowing humans to communicate with computers using natural language. We distinguish this from the broader task of natural language understanding, the definition of which is dependent upon what we take the meaning of “understanding” to be. We will consider the problem of designing natural language interfaces by means of Montague’s definition of meaning as entities serving as interpretations of expressions. Under this definition, meaning can be taken as a tool allowing for expressions in natural language to be “interpreted”.

To define what it means to interpret a natural language utterance, it is necessary to look at the central problem which lies at the heart of natural language interfaces — the Grammatical Mapping Problem, which is

\[
\text{the problem of computing the mapping } \Gamma \text{ between the surface form of the utterance and its meaning (the claims it makes about the world, the discourse, the speaker, and so forth)} \quad \text{[Kaplan, 1987, p.349]}
\]

Under this formulation, the mapping \( \Gamma \) relates surface form to its interpretation, syntax to semantics. Thus, \( \Gamma \) allows us to get from the surface form to the meaning-entity in order to ‘interpret’ the surface form expression. The problem of designing natural language interfaces is defined as being concerned with the mapping between the expression in natural language (its surface form) and the interpretation of the expression by the computer. Since the natural way in which we communicate with computers is programmatically i.e. by means of programming languages, we understand the computer’s interpretation of the expression \textit{in a given programming language}. Thus, the mapping problem as applied to natural language interfaces can be generalized to the problem of
computing the mapping between expressions in natural languages to expressions in pro-
gramming languages — the system must interpret the natural language utterance that is given to it as an expression. To elucidate this problem, we will consider the frame-
work of Lexical Functional Grammar (LFG) and its potential for use in the process of interpretation.
CHAPTER 3

Lexical Functional Grammar

Chomsky’s Generative Grammar uses both a syntactic representation (phrase structure tree) and a defined transformation to map between distinct representations of the same formal structure. This notion of mapping between structures, as generalized by the Grammatical Mapping Problem, allows us to relate form with meaning, syntax with semantics. Lexical Functional Grammar, as developed by Joan Bresnan and Ronald Kaplan in the 1970s [Kaplan and Bresnan, 1995], is built upon a notion of structural correspondence to replace the transformation mechanism of generative grammar. Rather than establishing a mapping between distinct phrase-structure trees, LFG establishes one between structures of distinct formal types. Proposed as a formal framework for syntactic modeling, it seeks to be both “computationally tractable” and “psycholinguistically adequate”. Psycholinguists, like computational linguists, are interested in algorithms to model linguistic processing capabilities, but specifically within the context of human cognition in order to “model or elucidate the language processing capabilities of human speakers and listeners”. The psycholinguistic approach, tracable to Chomsky’s claim that linguistic ability is innate, motivates LFG’s aim to “create a theory that could form the basis of a realistic model for linguistic learnability and language processing” [Heine et al., ].

The LFG framework is concerned with two representations of syntactic structures — the constituent structure and the functional structure — from which linguistic analysis can proceed. They are broadly analogous to Chomsky’s surface structure and deep structure. LFG is a generative grammar in the Chomskyian tradition in which different types of linguistic information is modeled using distinct formal representations related through the correspondence architecture. The motivation behind developing
structural correspondence between different formal structures lies in the observation that different representations capture different information dependencies. A phrase-structure tree, as taken from generative grammar, reflects the surface structure of the natural language expression, preserving linearity and word-order; functional structure seeks to represent abstract, internal grammatical relations. The correspondence architecture of LFG establishes a formal relationship between these parallel representations of linguistic information by means of a piecewise correspondence function $\phi$. The observation that functional organization across languages is relatively consistent motives the use of the LFG framework — where word order is almost certainly not preserved across languages, relations expressed by the constituent elements of the sentence such as subject-of, object-of etc are preserved in the functional structure.

1. Formal Architecture

1.1. Constituent Structure. The constituent structure (c-structure) of LFG is a phrase structure tree in the style of Chomsky’s generative grammar in which phrase structure rules define the way a sentence can be constructed from syntactic parts$^1$. The c-structure, as shown in Figure 1a thus encodes phrasal dominance and precedence relations - the “surface structure” of a natural language utterance. As in generative grammar, it is derived in the usual way by means of phrase structure rules with help of a lexicon, a minimal version of which is given in Figure 1a. An example of a c-structure representation for the sample sentence “Jack needs flour” is given in Figure 2a

1.2. Functional Structure. The functional structure as described within the LFG framework is represented as an attribute-value matrix which encodes grammatical relations such as subject and object as shown in Figure 2b. The relations encoded by the f-structure, the functional syntactic information, are used to determine

$^1S \rightarrow NP\ VP$ is a phrase structure rule denoting the observation that a sentence can be broken down into a noun phrase and a verb phrase is represented by the rule given in 3, along with the observations that a noun-phrase can be comprised of a determinative with a noun, and a verb-phrase is made up of a verb and a noun-phrase.
semantic composition. The use of f-structures to represent predicate-argument relations cross-linguistically is motivated by the observation that predicate-argument relations are encoded similarly in both configurational and non-configurational languages [Heine et al., ]. Thus, such relations can be considered to be universal. The approach to semantic assembly within the LFG framework we will be taking espouses the claim that f-structures act as the principle determinant of meaning composition - they tell us how to combine the meaning of the whole from its parts. This is achieved by extending the correspondence architecture, which establishes a structural correspondence $\phi$ between c-structure nodes and f-structure units$^2$, to the correspondence between f-structures and semantic structures ($\sigma$-structures). The most important attribute of a f-structure for our purposes will be the PRED attribute, which gives the meaningful

$^2$We will return to the central idea behind the correspondence architecture of LFG in the next section.
representation of the f-structure and acts as the part of the f-structure which is most important in the process of semantic assembly.

Grammaticality of a natural language utterance is defined within the LFG framework in terms of f-structure completeness and coherence where

- An f-structure is *locally complete* if and only if it contains all the governable grammatical functions that its predicate governs. An f-structure is *complete* if and only if it and all its subsidiary f-structures are locally complete.
- An f-structure is *locally coherent* if and only if all the governable grammatical functions that it contains are governed by a local predicate. An f-structure is *coherent* if and only if it and all its subsidiary f-structures are locally coherent.

[Kaplan and Bresnan, 1995, p.65]
Thus, there is a formal notion of acceptability and thus grammaticality defined in terms of f-structures in LFG.

Predicate-argument structures have been used in the field of Information Extraction (IE) [Surdeanu et al., 2003], where a domain-independent IE paradigm based on predicate-argument structure is proposed. The approach is compared to that of pattern-matching by means of event templettes, which are “frame-like structures with slots representing basic event information”. The method essentially develops a mapping between predicate-argument structures and templette slots in order to extract information from customizable domains, and the paper demonstrates the improved performance of the new paradigm. The utilization of predicate-argument relationships will be crucial in our approach to natural language interfaces, and a motivator for the choice of LFG as a linguistic framework stems from its observation of the universality of such relations and its potential for semantic analysis.
1.3. **Correspondence Architecture.** The correspondence architecture of LFG posits that there exists some piecewise correspondence function \( \phi \) which relates different kinds of linguistic information, each modeled by “distinct, simultaneously present grammatical structures, each having its own formal representation”. We have seen that LFG focuses on two levels of syntactic representation: c-structure and f-structure. The correspondence architecture allows a relationship to be established between the distinct formal structures (i.e. phrase-structure tree and attribute-value matrix) such that the nodes of the c-structure map to f-structures. This correspondence model replaces Chomsky’s transformational model in which he maps between distinct structures of the *same* formal type. The crucial point of this correspondence model is that two structures in correspondence may denote different information about the same natural language expression. Given a c-structure corresponding to a natural language expression, we can map that c-structure into an f-structure to represent different information about the same natural language expression. The ability to establish correspondences between different types of structures allows different types of linguistic phenomena to be analyzed. The c-structure allows us to analyze how a sentence breaks down into constituent phrases, the f-structure states how the pieces of the sentence are related to each other in terms of grammatical functions. For our previous example of c-structure (Figure 2a), we can define the mapping between c-structure nodes and f-structure units by explicitly showing a piecewise correspondence between the structures through the addition of constraints. We assume that f-structure units can be put in correspondence with c-structure nodes following the basic illustration by Kaplan in Figure 3 as taken from [Kaplan, 1987, p.356], where the \( \phi \) correspondence function relates nodes with f-structures in such a way that “descriptions of elements in [the range of the correspondence] can be defined in terms of the elements and relations of its domain.” These two structures make up the basis of LFG, and are related by “a functional projection function \( \phi \) from c-structure nodes to f-structures” [Dalrymple, 1999, p.4], which corresponds to the general notion of mapping between structures as expressed in the grammatical mapping problem discussed earlier and is reflective of the general correspondence
architecture of LFG. Thus, we can describe f-structures in terms of c-structure nodes and relations graphically as shown, and we do so concretely for our previous example, shown in Figure 4. The notion of structural correspondence can then be formalized by including annotations on phrase-structure rules for f-structure assembly as given in Figure 5. These annotations relate c-structure nodes to f-structure values, where ↑ SUBJ can be understood as referring to the grammatical function (attribute) “SUBJ” in the f-structure which corresponds to the mother (points upwards) of the current NP node. Then, (↑ SUBJ) =↓ means that the f-structure corresponding to the mother of NP has a SUBJ attribute whose value is the f-structure mapped to by its children c-structure nodes and ↑=↓ means that the f-structure corresponding to the mother of VP is equivalent to the f-structure corresponding to the children of VP. Essentially, ↑ and ↓ refer to specific f-structures in terms of c-structure relations. Since these metavariables ↑ and ↓ simply stand for f-structures, we can alter our notation to make explicit this correspondence as in Figure 6 to derive Figure 7 where the metavariables are substituted for variables $f, g, h, i, j$ corresponding to f-structures. The metavariables are simply instantiated according to annotations on the phrase structure rule, where the ↑ in the first phrase structure rule given in Figure 5 is instantiated with $f$ for nodes NP and VP since they refer to the same c-structure node. However, ↓ is instantiated as $g$ and $h$ for NP and VP respectively since they refer to different nodes and thus, possibly, different
f-structures. Note that \( f, h, i \) end up referring to the same f-structure unit. We simplify our diagram by not including the annotations for \( NP \rightarrow N \)
1.4. Semantic Structures. The correspondence architecture allows us to move between different linguistic forms by means of a projection function which places c-structure nodes in piecewise correspondence with f-structure units. An important development to the LFG framework extends the correspondence architecture by means of
a *semantic projection* function that relates f-structures with a representation of their meaning. The semantic structure, or \( \sigma \)-structure, is thus analogous to the c- and f-structures in that each structure represents different information derived from some simultaneously existing linguistic structures. Where the c- and f-structures represent surface and functional structure, the \( \sigma \)-structure represents the meaningful structure of the natural language expression. The approach taken in this thesis with regards to semantic structures draws primarily on Kaplan and Bresnan’s identification of the crucial point of connection between f-structures and \( \sigma \)-structures in the value of the PRED attribute of the f-structure, stating that

> When the f-structure is semantically interpreted, these forms are treated as patterns for composing the logical formulas encoding the meaning of the sentence. Thus, the semantic interpretation...is obtained from the value of its PRED attribute [Kaplan and Bresnan, 1995, p.33]

This connection is utilized by Halvorsen in his approach to semantic structures by means of representing them as formulas in intensional logic, which can then be interpreted model-theoretically [Halvorsen, 1983, p.569]. The role of the semantic structure is thus to provide a method for interpreting the expression, and corresponds to Montague’s definition of meaning.

Halvorsen understands semantic structures as being derived from f-structures. Specifically, he notes that the first step of mapping from f-structures to \( \sigma \)-structures concerns the basic, meaningful expressions of the f-structure as denoted in the f-structure in the guise of *semantic forms*, or those values in an f-structure marked with single quotation marks such as ‘Jack’ and ‘flour’. Such semantic forms also include those which represent predicate argument lists as given by `need<SUBJ, OBJ>` which define how grammatical functions contained in the f-structure are associated by the relation *need* as given by the semantic form. We explicitly denote the semantic forms we are concerned with in our augmented lexicon shown in Figure 8, where each semantic form is associated with the f-structure PRED attribute reflecting a meaning for the f-structure. By simply con-
3. LEXICAL FUNCTIONAL GRAMMAR

JACK N (↑ PRED) = ‘jack’
NEEDS V (↑ PRED) = ‘need’ <↑ SUBJ, ↑ OBJ >
FLOUR N (↑ PRED) = ‘flour’

**Figure 8.** Sample minimal lexicon for examples

considering the f-structure, we can construct the semantic structure need<‘jack’, ‘flour’> by looking at the PRED attribute of the main f-structure need<SUBJ, OBJ>. From this, we see that the meaningful expression relates the SUBJ attribute with the OBJ attribute. The meaningful expressions denoting these attributes are then substituted into the semantic form to give the semantic structure.

Halvorsen’s approach to the syntax-semantics interface uses the PRED attribute of the f-structure as the basis for interpreting the meaning of the natural language expression. This is the approach we take in our implementation of a natural language interface. Specifically, we will be using the XLE implementation of an LFG parser to generate f-structures. These f-structures will allow us to construct an interpretation of the natural language expression in a language that can be understood by the machine. We use the f-structure as the basis for our correspondence with semantic structures, assuming that semantic composition is primarily determined by the functional syntactic information represented in the f-structures. Relations such as subject-of, and object-of, uniformly represented in the f-structure, are hence the main determinant of meaning composition under this view. This approach to the syntax-semantics interface allows a systematic way to assemble the meaning of the whole expression from that of the parts by means of the predicate-argument structure as reflected in the PRED attribute’s semantic form, demonstrating the principle of compositionality in practice.
Figure 9

Before presenting our natural language interface, it is necessary that we highlight a different approach to semantic assembly within the LFG framework — that of glue semantics which uses a “glue language” for semantic assembly. Rather than using the value of the PRED attribute of f-structures, the glue approach as presented by Mary Dalrymple [Dalrymple, 1999] augments the lexicon with meaning constructors, formulas which can be used in conjunction with the glue language to yield a meaning for the expression via deduction. We highlight the glue approach in the next chapter to show that meaning assembly can also be understood as a deductive process which uses the proof mechanisms of linear logic to construct the meaning of a natural language expression from the meaning of its parts. In this approach, lexical entries are assumed to contribute premises which are used to form a conclusion about the meaning of the expression on the basis of the meaning of its constituents. The augmented lexicon as used in the glue approach is shown in Figure 9. Each lexical entry now has an associated meaning constructor which is used for meaning assembly where the symbols \( \otimes \) and \( \rightarrow \) stand for linear logic conjunction and implication respectively. The linear logic symbols define the way the meaning language is assembled — the meaning constructor
for NEED states that the meaning of the entire expression is constructed by consuming
the meanings of the SUBJ and OBJ of the expression. Recall our annotations on phrase
structure rules which used metavariables $\uparrow$ and $\downarrow$ to refer to both c-structure nodes and
the f-structures they map to. Specifically recall that $\uparrow$ refers to the f-structure of the
mother node in the c-structure tree. $\uparrow_\sigma \sim Jack$ then states that the f-structure of the
mother node of N in the c-structure tree (NP in Figure 7)) has meaning ‘Jack’ while
$\uparrow_{\text{SUBJ}} \sim X$ states that the SUBJ attribute of the mother node of V (VP) has some
meaning ‘X’ which is used to assemble the meaning of $\uparrow_\sigma$. Though the $\uparrow_\sigma$ notation is
used in all three lexical entries, they technically refer to meanings of different f-structures
since they refer to different mother nodes as seen in Figure 7 and thus are instantiated
as different f-structures where the instantiated version of the meaning constructors are
given below

$$\forall X, Y. (f_{\text{SUBJ}})_\sigma \sim X \otimes (f_{\text{OBJ}})_\sigma \sim Y \rightarrow f_\sigma \sim \text{need}(X, Y)$$

$$g_\sigma \sim jack$$

$$j_\sigma \sim flour$$

In the glue approach, we include assembly instructions which dictate how compo-
nents of the f-structure combine to produce the meaning of the whole expression. These
assembly instructions are simply rules in linear logic, used in the process of meaning
deduction. The lexicon is augmented to include the meaning constructors contributed
by lexical entries in Figure 9 where $\uparrow_\sigma \sim Jack$ is an expression in the meaning language
which can be interpreted as “The semantic structure of f-structure $\uparrow$ has meaning Jack”
The glue language is used to assemble meanings, and makes up the deductive mechanism
of semantic assembly.

The crucial part meaning constructors play in semantic assembly come to light
when we consider the meaning constructor for the lexical items in the previously given
sentence “Jack needs flour” given in Figure 9 and note that we are able to derive a
meaning for the sentence given these formulas in the meaning language which can be
used in conjunction with the glue language to allow us to reassemble the meaning of the whole f-structure by considering its parts. Thus, the glue approach considers meaning assembly as a deductive process which moves from a set of premises generated by lexical entries to a conclusion about the meaning of the whole expression based on functional relations as expressed in the LFG f-structure. The way one reaches the interpretation of an expression, the meaning, is through a proof in linear logic. The use of linear logic proof for semantic interpretation can be seen in research concerning narrative formalization, which uses linear logic proofs as narrative representations. In the next chapter, we will begin by reemphasizing the role of linear logic in semantic assembly as expressed in the glue approach to highlight its similar role in narrative formalization. This will then show the connection between the LFG framework and its use in natural language interfaces.
CHAPTER 4

Linear Logic for Language Modeling

1. Glue Semantics

The use of linear logic in the glue approach to semantic assembly derives its compatibility from its resource-sensitive nature. Specifically, linear logic “introduces accounting of premises and conclusions, so that deductions consume their premises to generate their conclusions” [Dalrymple et al., 1993] thus capturing the resource-sensitivity of natural language — “that lexical items and phrases contribute exactly once to the meaning of a sentence” [Dalrymple, 1999, p.15]. That the semantic analysis of lexical item “needs” concerns the two resources “Jack” and “flour” is representable by means of linear logic. Each lexical item contributes a formula in the meaning language (meaning constructor) which is a premise used in the deduction of meaning.

A proposition in linear logic is built from the connectors $\otimes$, $\&$, $\oplus$ and $\rightarrow$ where the first two symbols stand for conjunction, and the next two stand for disjunction and implication respectively. We will primarily focus on the $\otimes$ conjunction and $\rightarrow$ implication connectives in the glue approach. The grammar for linear logic propositions $A, B, C$ is thus given by

$$A, B, C ::= X \mid A \otimes B \mid A \rightarrow B$$

where $X$ is a propositional constant. An assumption $\Gamma$ is then defined as a sequence of propositions where a judgment of the form

$$\Gamma \vdash A$$

means that from the assumption (sequence of propositions) $\Gamma$, the proposition $A$ can be concluded. The implication that $A$ can be generated from $\Gamma$ is denoted with $\vdash^1$.

\footnote{Note this notion of implication in judgments is distinct from that in propositions ($\rightarrow$).}
Linear logic is understood as a logic of resources in which no *permanent truth* is assumed as in traditional logic. Specifically, linear logic implication $A \multimap B$, can be understood as stating that a resource $B$ can be generated from the consumption of a resource $A$. $A \multimap B$ does not hold *ad infinitum*, but only so long as there exist resources $A$ available for consumption since in linear logic the assumption that ‘if a fact is used to conclude some other fact [in traditional logic], the fact is still available’ [Wadler, 1993] does not hold. Thus,

$$A \otimes (A \multimap B) \multimap A \otimes B$$

does not hold in linear logic since the formula $A \multimap B$ consumes $A$. The implication is also consumed and thus the correct formula would be

$$A \otimes (A \multimap B) \multimap B$$

Thus, by considering meaning constructors as glue language formulas, we are highlighting the fact that the semantic structures are resources available for meaning assembly and emphasizing the notion of semantic assembly as a deductive process from premises to conclusions.

The use of linear logic as a glue language for meaning assembly as presented in [Dalrymple, 1999] uses the tensor fragment of linear logic in which glue language formulas are simply a sequence of atomic formulas of the form $S \sim M$, otherwise known to be formulas in the meaning language, which are related by means of linear logic connectives $\otimes$, $\multimap$. These atomic formulas are the meaning constructors which act as the premises from which meaning assembly begins. Meaning constructors are contributed by lexical entries in the glue approach, and thus from each lexical item in the previous sentence “Jack needs flour”, we show the resources contributed by the lexical items in Figure 9. Recall our correspondence diagram from Figure 7 in which we see that $f$ SUBJ = $g$ and $h$ OBJ = $j$. Thus we get the three meaning constructors
as shown below

\[
g_\sigma \leadsto 'Jack' \\
\forall X,Y. g_\sigma \leadsto X \otimes j_\sigma \leadsto Y \rightarrow f_\sigma \leadsto need(X,Y) \\
j_\sigma \leadsto 'flour'
\]

The first formula \(g_\sigma \leadsto 'Jack'\) expresses the binary relation between semantic structure \(g_\sigma\) derived from f-structure \(g\) and the meaning ‘Jack’. ‘Jack’ is simply a term in the meaning language, the choice of which is not governed by the use of linear logic as glue. The function of linear logic is to simply allow for the assembly of these meaning constructors by means of logical rules. It is simply the mechanism by which meanings are composed. Using these equations in the glue language as premises, we can derive a meaning for the natural language utterance.

\[
g_\sigma \leadsto 'Jack' \otimes j_\sigma \leadsto 'flour' \otimes \forall X,Y. g_\sigma \leadsto X \otimes j_\sigma \leadsto Y \rightarrow f_\sigma \leadsto need(X,Y) \\
\vdash j_\sigma \leadsto 'flour' \otimes \forall Y. g_\sigma \leadsto 'Jack' \otimes j_\sigma \leadsto Y \rightarrow f_\sigma \leadsto need('Jack',Y) \\
\vdash g_\sigma \leadsto 'Jack' \otimes j_\sigma \leadsto 'flour' \rightarrow f_\sigma \leadsto need('Jack', 'flour') \\
\vdash f_\sigma \leadsto need('Jack', 'flour')
\]

Otherwise, in sequent-form

\[
A \vdash g_\sigma \leadsto 'Jack' \quad C \vdash j_\sigma \leadsto 'flour' \\
\frac{A,C \vdash g_\sigma \leadsto 'Jack' \otimes j_\sigma \leadsto 'flour'}{A,C \vdash f_\sigma \leadsto need('Jack', 'flour')} \\
A,\forall Y. g_\sigma \leadsto 'Jack' \otimes j_\sigma \leadsto Y \rightarrow f_\sigma \leadsto need('Jack',Y) \vdash f_\sigma \leadsto need('Jack', 'flour')
\]

Thus, the meaning of a natural language expression can be deduced from the meaning of its parts in this manner by means of linear logic as a compositional mechanism for meaning assembly. We also note that the semantic structure \(need('Jack', 'flour')\) represents the predicate-argument structure of the original expression after two applications of structural correspondence - the mapping from c-structure to f-structure, then from f-structure to \(\sigma\)-structure.
2. Linear Logic for Narrative Formalization

Linear logic is used as linguistic glue in glue semantics, allowing the semantic content of a natural language expression to be extracted by means of the LFG correspondence architecture. Another application of linear logic to linguistic phenomena is in the field of narrative formalization. In [Bosser et al., 2010], it is observed that linear logic implication allows for modeling of causality in terms of narrative resources. Specifically, a narrative action can be modeled in linear logic as \( A \rightarrow B \) such that its interpretation corresponds to a model of causality in which the consumption of resources in \( A \) generates resources in \( B \), thus reflecting an update the context caused by this narrative action.

This resource-oriented approach to narrative formalization by means of linear logic implication allows a story to be modeled by a sequent in which the initial narrative state is modeled on the right hand side and the final global state after the execution of narrative actions would be on the left hand side. The entire proof tree then represents the sequence of narrative actions which led to the final context.

Consider the narrative of Jack and the Beanstalk. A narrative action would be the exchange which occurs when Jack goes to the marketplace to sell his cow for some beans. The resources involved include Jack, the cow, and the beans. Specifically, the resource “cow” is consumed to generate the resource “beans” when the initial narrative state is represented by the natural language expression “Jack sells his cow”. This can be modeled by the linear logic formula

\[ \text{J} \rightarrow \text{B} \]

where \( \text{J} \) stands for “Jack sells his cow” and \( \text{B} \) stands for “Jack has beans” to represent the causality relationship between the two statements - \( \text{J} \) gives rise to \( \text{B} \). The statement \( \text{J} \) can be broken down to further define the relationships between the resources. Specifically, “his cow” means “Jack has a cow” and thus \( \text{J} \) can be rewritten using \( \text{I} \rightarrow \text{J}' \) where \( \text{I} \) stands for “Jack has a cow” and \( \text{J}' \) refers to the narrative state where “Jack has a cow” and “Jack sells the cow”, otherwise \( \text{J}' = \text{I} \otimes \text{J} \) which makes our previous
formula

\[ I \otimes J \multimap B \]

which models the transition from the initial narrative state in which Jack has a cow and Jack wants to sell the cow to the state where Jack has beans.

A narrative is then simply made up of a sequence of such narrative actions, each modeling a transition between narrative states. The ability to formally describe narrative generativity, variability and drive by means of a sequent is a direct consequence of this use of linear logic. Consider a sequent as shown with a fixed context \( \Sigma \), an extension of \( \Gamma \) which includes permanent rules. These are defined narrative actions which can be repeated in the narrative i.e. narrative action denoting movement of a character from one location to another location.

\[ \Gamma, \Delta \vdash A \]

Consumption of resources in \( \Delta \) by rules in \( \Sigma \) leads to the final global state represented on the right hand side of the sequent. The full proof, with its consumption and generation of resources, is representative of the narrative. A different order of consumption will result in a different proof, possibly leading to a different conclusion. This sequent representation is thus able to model narrative generativity - the possibility to model a different narratives given the same beginning and ending. This idea of using proof search as narrative description is the theoretical foundation motivating the design of Ceptre, which allows narrative generativity to be modeled in terms of resources and pre-defined transitions between narrative states by means of linear logic formulas. Ultimately, this approach to narrative formalization uses linear logic for narrative assembly, mirroring that of semantic assembly. Where narrative modeling is reformulated as a deductive process, semantic assembly is similarly understood in terms of a deductive process of meaning assembly. Both draw on the use of linear logic as a formalism for modeling language, thus allowing a natural relation to be established between the two representations.
We previously defined the task at hand as one in which we are required to consider correspondences between levels of representation of linguistic information. The grammatical mapping problem generalizes to that of computing the correspondence between expressions in natural languages to expressions in programming languages. The claim of universality made by f-structures with respect to predicate-argument structure allows for its suitability in cross-linguistic analysis, and the glue semantics extension to the LFG framework allows this universality to be exploited in the task of meaning assembly. Specifically, our approach uses the universality of f-structures to transform natural language inputs to formulas of linear logic as represented in the Ceptre programming language. This is a motivator behind our adoption of the LFG framework for the linguistic foundation of our approach to natural language interface design. The glue semantics extension in which lexical entries contribute linear logic formulas which act as meaning constructors is another, more central, motivator behind the use of LFG. The ability of the PRED attribute of f-structures to represent the predicate-argument structure, which is assembled by means of linear logic, makes for a natural correspondence between f-structure PRED attributes and the linear logic formula representing the meaning of the expression, as derived in glue-LFG. To understand this, we introduce the programming language that will be set in correspondence with natural language - Ceptre - in the next chapter.
CHAPTER 5

Ceptre

The use of linear logic to model narratives motivates the design of Ceptre, a specification language developed by Chris Martens. The correspondence between linear logic proof search and narrative description, as demonstrated in the previous section, is the theoretical basis for Ceptre. That is: a narrative is modeled by formulas in linear logic which define resources and rules to manipulate defined resources. The rules are the same narrative actions as previously defined, and a narrative execution proceeds by means of proof search. The ability of linear logic implication to model state change allows forward chaining proof search to act as a model of state transition for narrative formalization since the derivability captured in linear logic proofs, defined in terms of resource replacement, is representative of state change. Forward chaining proof search scrutinizes the assumptions on the left hand side of a judgment to derive a proof for the right hand side. Under this framework, narratives are thus understood in terms of their components, and narrative progression occurs in stages where an initial state is repeatedly replaced by means of defined rules in order to lead to some final state.

More formally, a Ceptre program is defined as a collection of terms, predicates and rules $\Gamma$, with an initial specification of the initial state $\Delta_0$. Types such as character, location and object which subcategorize the resources are first declared in the form

$t : \text{type}$ (i.e. character : type.) Terms are then declared with types as in jack : character which declares a resource named “jack” of type “character”. Ceptre predicates make up the narrative state, and are declared in the form

$$p \ a_0 \ldots \ a_{n-1} : \ pred$$
as shown by **has character object : pred.** Execution of the program then proceeds by means of state-transitions, where a rule applying to a subset of $\Delta_i$ generates a new $\Delta_{i+1}$ We return to the linear logic formula representing the narrative action of Jack selling his cow for some beans $I \otimes J \to B$ where we can rewrite $I$, $J$ and $B$ to explicitly refer to the resources and relationships mentioned by this formula as below.

$$I = \textbf{has} \text{ Jack cow}$$

$$J = \textbf{sell} \text{ Jack cow}$$

$$B = \textbf{has} \text{ Jack beans}$$

These formulas as rewritten with explicit reference to the relationships **has** and **sell** which hold between resources are Ceptre predicates instantiated with terms. Such predicates are *grounded* for they refer explicitly to terms. The Ceptre program thus far then consists of the rule “**has** jack cow $\otimes$ **sell** jack cow $\to$ **has** jack beans”. Then, given the initial context

$$\Delta_0 = \textbf{has} \text{ jack cow} \otimes \textbf{sell} \text{ jack cow}$$

the rule is applied to consume these premises and generate the Ceptre predicate **has** **jack beans**. Thus, the context $\Delta_1 = \textbf{has} \text{ jack beans}$ is the new context which prevents us from applying the same rule. If the rule is the only applicable rule in the Ceptre program (as in this case), we say the program has reached *quiescence*. We can no longer reuse this same rule because the resources $I$ and $J$ have been consumed to generate $B$, thus modeling the way events in narratives are dependent upon the resources available and how rules concerning narrative states are not permanent facts. Suppose there is another rule “**has** jack beans $\to$ at jack home” already defined in the program. Then, $\Delta_1 = \textbf{at} \text{ jack home}$ will be our context at quiescence since that rule can be applied to the context generated after applying the initial rule. Jack was able to exchange his cow for beans only because the narrative state contained resources which were applied to a rule to generate a new state in which the same rule may or may not hold - depending on the resources at hand.
A Ceptre program is thus a collection of predicates representing story state components and rules relating how such components are consumed to generate new story components. These predicates are initially declared as ungrounded predicates. That is, given a grounded predicate of form $p a_1 \ldots a_n$ which takes $n$ arguments, the ungrounded predicate is given by $p t_1 \ldots t_n$ where $t$ is the type of term $a$, or otherwise $t_1 = \text{type}(a_1)$.

An example Ceptre program to model the story of Jack and the Beanstalk is given in Figure 1. We define three types of story resources - character, location and object. We then define terms (i.e. jack, market) and predicates which define relationships between resources (i.e. has character object). The rules defined as part of a stage which proceeds to quiescence indicating that the process of substitution has stabilized and no more resources can be consumed given the rules of the stage. To finish our Ceptre program, we need to define an initial starting state for the narrative. We choose the state consisting of story components at jack home and has jack cow to be our initial state.

1. Ceptre as meaning-representation

The use of linear logic in both meaning assembly and narrative formalization allows us to establish a natural correspondence between the two methods of representation. Specifically, consider the sentence

Jack has a cow

which parses to a c-structure tree in Figure 2 which corresponds to f-structure in Figure 3 from which we get the $\sigma$-structure

$$f_\sigma \sim \text{has}(\text{`jack'},\text{`cow'})$$

which can be set in correspondence with the instantiated Ceptre predicate

has jack cow

For this thesis, we utilise the XLE implementation of an LFG parser which does not use the glue-approach. However, the predicate-argument structure is still reflected in the PRED attribute and thus we are able to construct the meaning of the whole from
character : type.
location : type.
object : type.

jack : character
cow : object
beans : object
home : location
market : location

at character location : pred.
has character object : pred.
sell character object : pred.

stage begin = {
sell-cow
: at jack home * has jack cow -o sell jack cow * at jack market
get-beans
: at jack market * sell jack cow -o has jack beans
}

c CONTEXT INIT = {
at jack home, has jack cow
}

FIGURE 1

the information contained in the f-structure, as done in the approach of Halvorsen. Specifically, the semantic form is still obtainable from the value of the PRED attribute
in the XLE implementation, thus giving us

\[ f \text{ PRED} = \text{HAS} < \text{SUBJ}, \text{OBJ} > \]

where “HAS” indicates the semantic association between grammatical functions SUBJ and OBJ. The ability of these semantic forms to represent predicate argument structure is crucial for our design of a natural language interface for Ceptre considering that story components can be expressed in terms of relations between resources in both programming language and natural language. The Ceptre predicate has character object : pred is a Ceptre expression defining the relation ”has” between a character resource and an object resource. This story state is equivalent to the general expression in natural language: ”A character has an object”. Suppose the predicate is grounded in variables “jack : character” and “cow : object”. Then, the predicate has jack cow is equivalent to the natural language expression Jack has a cow. This natural
correspondence between a story state predicate as expressed in Ceptre and a natural language expression forms the basis of our approach to designing a natural language interface for Ceptre.
A natural language interface for Ceptre

Ceptre allows users to model narratives by specifying narrative action by means of rules which define consumption and generation of story components. Causality in narrative action is modeled by means of linear implication. We present a method for generating Ceptre program code from natural language text by means of LFG. We previously observed a correspondence between Ceptre story components and natural language expressions. Thus, we want to be able to specify a Ceptre program by means of natural language expressions. These expressions will be given in the form of causal relationships “IF...THEN...” or “WHEN...THEN...” where the holes indicated by ellipses are filled in with sequences of natural expressions placed in conjunction by means of separator “AND”. Given such an expression i.e.

IF Anne needs candy AND Anne is at home THEN Anne needs candy AND Anne is at the store

we need to generate

(1) the set of types:
   character, object, location
(2) the set of declared atoms:
   anne : character, candy : object, store : location, home : location
(3) the set of declared predicates:
   need character object, at character location
(4) the grounded rule:
   need anne candy ⊗ at anne home → need anne candy ⊗ at anne store

Otherwise, as given by the Ceptre language definition, the four types of data making up a Ceptre program are
• type declarations of form \( t : \text{type} \)

• term declarations of form \( a : t \) where \( t \) is a type previously declared as \( t : \text{type} \)

• predicate declarations of form \( p t_1 \ldots t_n : \text{pred} \) where \( t_i \) is a previously declared type \( t_i : \text{type} \)

• rule declarations of form \( r : A \rightarrow B \) where \( r \) is an identifier for the rule, \( A \) and \( B \) are sequences of predicates related by means of linear implication

These pieces make up a Ceptre program, and we wish to be able to generate pieces of a Ceptre program by looking at f-structure representations of natural language expressions. Specifically, the natural language expression can be transformed to the grounded rule (4) by simply recalling that the f-structure PRED attribute for the natural language expression “Anne needs candy” can be trivially rewritten to denote the equivalent grounded Ceptre predicate \( \text{need} \ \text{anne} \ \text{candy} : \text{pred} \) which represents the same narrative state. Thus, we can describe a correspondence between grounded Ceptre predicates and the f-structure representation of the equivalent natural language expression. Ceptre declarations of predicates, as defined, utilize type arguments rather than term arguments. These are ungrounded predicates whereas predicates which have been instantiated with term arguments corresponding to the type arguments declared are grounded. A grounded Ceptre predicate can be mapped to its ungrounded form by applying a function \( \text{type} \) from (2) to (1) that takes a declared atom and gives the type it is declared as such that

\[
\text{type}(a : t) = t : \text{type}
\]

Then, given a grounded predicate declaration, we can transform it to an ungrounded predicate declaration in the set (3)

\[
p \ a_1 \ldots \ a_n = p \ \text{type}(a_1) \ldots \ \text{type}(a_n)
\]

These pieces make up the structure of a Ceptre program and will be generated from the f-structure by means of structural correspondence. In the structural correspondence
developed between c-structure nodes and f-structure units in LFG, we see that the c-
structure nodes constrain the values of f-structure unit attributes. Thus, in Figure 5, our
phrase-structure rules include constraints on f-structure units corresponding to specified
c-structure nodes. For our natural language interface, we require our f-structure units
to constrain the generated Ceptre structure in order to demonstrate the same structural
correspondence between f-structure and Ceptre structure.

1. Ceptre (ζ-) structure

We define our ζ-structure to be a Ceptre predicate with no ungrounded variables.
These correspond to specific natural language expressions which model specific story
states as previously observed in the example of the correspondence between Ceptre
predicate has jack cow and the predicate-argument structure for the natural language
expression Jack has a cow. This predicate argument structure is simply the value of
the PRED attribute of the f-structure corresponding to the expression as given in LFG,
and thus we have a method for mapping between natural language expressions and
Ceptre grounded predicates by means of this correspondence between ζ-structure and f-structure.

The ζ-structure, in conjunction with the information contained in the f-structure,
will allow us to build the other parts of the Ceptre program. Specifically, note that the
pred declaration for this expression which refers to the relevant TYPE for each argument
simply requires us to map from each argument in the Ceptre predicate to its associated
TYPE thus has jack cow is mapped to has type(jack) type(cow)

We have now established a piecewise structural correspondence between f-structure
and ζ-structure, which provides us with a method for constructing a valid Ceptre pro-
gram. Our natural language interface builds a Ceptre program from a set of natural
language expressions. Each natural language expression defines a forward chaining rule
in the form IF (A) THEN (B), or WHEN (A) THEN (B) where A and B are sequences of

---

1The ungrounded version (has character object0) does not enforce the fact that in the given
story rules, Jack is the character who has the object0 cow.
expressions which can each be mapped to a $\zeta$-structure. The relation between elements in sets $A$, $B$ is that of linear logic conjunction $\otimes$, and the relation between the two sets $A$ and $B$ is that of linear logic implication. Thus, we can map from the natural language expression to a function (linear implication) which takes set $A$ to set $B$, where elements of sets $A$ and $B$ are $\zeta$-structures corresponding to respective f-structure representations related by linear logic conjunction.

1.1. The Program. Consider the input file given in Figure 2 which produces the output given in Figure 3. Each sentence is put in correspondence with the linear logic formula $A \rightarrow B$. Consider the first sentence

IF Jack has a cow THEN Jack sells the cow AND Jack goes to the market

where

$A = \text{Jack has a cow}$

and

$B = \text{Jack sells the cow, Jack goes to the market}$

Each element of $A$ and $B$ is then parsed by XLE to produce an f-structure representation, given in Figure 1, which is set in correspondence with a $\zeta$-structure where

$\zeta(f) = (f\text{PRED})$

gives the $\zeta$-structure corresponding to the f-structure $f$. Then, for the f-structure $a_1$ corresponding to the sentence “Jack has a cow”, we get the $\zeta$-structure $\text{has jack cow}$ by repeatedly applying $\zeta$-correspondence to the predicate-argument structure given by the PRED attribute of LFG f-structures.

$\zeta(a_1) = (a_1 \text{ PRED})$

$= \text{has } < \zeta(a_1 \text{ SUBJ}) , \zeta(a_1 \text{ OBJ}) >$

$= \text{has jack cow}$
In this manner, we get the 3 $\zeta$-structures as shown below from the 3 input sentences.

$$A = \{ \zeta(a_1) = \text{has jack cow} \}$$

$$B = \{ \zeta(b_1) = \text{sell jack cow},$$

$$\zeta(b_2) = \text{at jack market} \}$$

The defining relation between A and B is linear implication. The defining relation between elements of sets A and B is linear logic conjunction. Thus, we get

$$\text{has jack cow} \to \text{sell jack cow} \odot \text{at jack market}$$

which is a valid Ceptre forward chaining rule defining the generation of constant pred elements sell jack cow and at jack market from the consumption of constant pred element has jack cow.

The pred declaration of these constant elements requires the generalization of arguments from grounded variables to type variables. Specifically, given the grounded predicate has jack cow, the ungrounded predicate declaration would be has character object0, and the ungrounded rule declaration would be

$$\text{has C O} \to \text{sell C O} \odot \text{at C market}$$

We previously expressed this relation by means of an assumed function type from constant term declarations to type variables. In the Ceptre predicate sell jack cow, “jack” and “cow” are constants that must have been previously declared to be of some given types i.e. character : type. and object : type. Then, TYPE(jack) = character and TYPE(cow) = object and we get the pred declaration has character object. Types are retrievable from f-structures; due to the “HUMAN” attribute which allows character types to be distinguished from non-character types if the lexicon recognizes the lexical item as a proper name. Terms typed as location are distinguishable by recognizing the presence of directive/locative prepositions in the f-structure as given by the PSEM attribute. Other types are typed as “objectn”, thus “cow” and “bean”
are defined to be of type \texttt{object0} as in the example since they both take the place of the second argument in the predicates \texttt{have jack beans} and \texttt{have jack cow}.

The output .cep file produced by the natural language interface requires the user to input an initial context as demanded by Ceptre in order to run the file. We note that a different initial context leads to a different proof and thus represents a different narrative, and thus have left it to the user to specify the narrative state they wish to begin with. This design choice allows users to analyze how different initial narrative states may lead to different narrative endings under the same set of user defined rules, which is simply \textit{narrative variability}. Suppose we start with the initial context

$$\Delta_0 = \{\text{have jack cow, at jack home}\}$$

then a possible narrative is as below

$$\Delta_0 \otimes \text{rule/0} \rightarrow \text{sell jack cow} \otimes \text{go jack market} \otimes \text{at jack home}$$

$$\otimes \text{rule/1} \rightarrow \text{at jack market} \otimes \text{sell jack cow}$$

$$\otimes \text{rule/2} \rightarrow \text{have jack bean}$$

$$\Delta_1 = \{\text{have jack bean}\}$$

We note that we end at the state where \texttt{have jack bean}. However, if we were to give an initial context of $$\Delta_0 = \{\text{have jack bean, at jack home}\}$$, none of the rules can be applied to this context and thus we do not transition from this state, representing the narrative where Jack has beans and stays at home.

Ceptre represents this proof alternatively by assigning variable $$x_i$$ to each element in the initial context as in Figure 4 where

\begin{align*}
\text{let } [x4, x3] &= \text{rule/0 } [x1, []] \\
\end{align*}

means that $$x1$$ was consumed by \texttt{rule/0} to generate

$$x3 = \text{sell jack cow}$$ and $$x4 = \text{go jack market}$$
let [x1] = have jack cow
let [x2] = at jack home
let [x4, x3] = rule/0 [x1, []];
let [x5] = rule/1 [x4, [x2, []]];
let [x6] = rule/2 [x5, [x3, []]];

Figure 4. Alternative representation of proof

Narrative generativity of the output file can be increased by rewriting certain predicates contained in rules as ungrounded predicates which take types rather than terms. This allows different terms to be applied to the predicates thus changing the way the narrative progresses. For instance, in Figure 5, a possible narrative trace is given below, with initial context

\[
\Delta_0 = \text{need abby candy, at james home, hungry james}
\]

\[
\Delta_0 \otimes \text{rule/3} \rightarrow \text{need james food} \otimes \text{need abby candy}
\]

\[
\otimes \text{rule/0} \rightarrow \text{go abby store} \otimes \text{need james food}
\]

\[
\otimes \text{rule/1} \rightarrow \text{at abby store} \otimes \text{need james food}
\]

\[
\Delta_1 = \{\text{at abby store, need james food}\}
\]

We see that if we rewrite rule/0 as

\[
\text{need C 0} \rightarrow \text{go C store}
\]

where C stands for character : type. and O stands for object0 : type., then the state need james food is able to transition to go james store, even though the rule “IF James needs food THEN James goes to the store” is not explicitly stated in the input file. If we rewrite rule/1 in the same manner as go character location \( \rightarrow \text{at character location} \) we get the new set of rules, which allows the narrative to transition to a different final state as shown below.
6. A NATURAL LANGUAGE INTERFACE FOR CEPTRE

rule/0' : need C O -o go C store
rule/1' : go C L -o at C L

$$\Delta_0 = \{\text{need abby candy, at james home, hungry james}\}$$

$$\Delta_0 \otimes \text{rule/0'} \rightarrow \text{go abby store} \otimes \text{at james home} \otimes \text{hungry james}$$
$$\otimes \text{rule/1'} \rightarrow \text{at abby store} \otimes \text{at james home} \otimes \text{hungry james}$$
$$\otimes \text{rule/3} \rightarrow \text{need james food} \otimes \text{at abby store}$$
$$\otimes \text{rule/0'} \rightarrow \text{go james store} \otimes \text{at abby store}$$
$$\otimes \text{rule/1'} \rightarrow \text{at james store} \otimes \text{at abby store}$$
$$\otimes \text{rule/2} \rightarrow \text{like abby james}$$

$$\Delta_1 = \{\text{like abby james}\}$$

The natural language interface does not attempt to infer any such generalizations. We have left it up to the user to manipulate the output to their needs. The interface does not try to model such variability - it only provides a model of the narrative as given to establish the most accurate correspondence between it and the input natural language expressions as parsed from relevant f-structures.

2. Limitations of our Natural Language Interface

Our interface uses the correspondence between f-structure PRED attributes and grounded Ceptre predicates to build a Ceptre program given a list of rules given in natural language. These rules in natural language describe causal relationships between narrative states. The natural language expressions we use to describe narrative states are simple verb phrases which contain a single main verb. Thus, the state where ”Jack goes to meet Gina” cannot be parsed to Ceptre predicate since the infinitive “to meet”, which can be broken down into a predicate meet jack mother, acts as the object of the verb “goes”. A possible extension to the natural language interface would then consist of handing verb phrases with multiple main verbs, but the current version is only able to
handle rules consisting of predicates expressed in natural language with only one main verb.

The implementation of such an extension would consist of the addition of `action : type` to classify infinitive verb phrases. We show the c- and f-structure parse by XLE in Figure 6 in which we see that the node labeled 1328 states that the c-structure rooted at that node is an infinitive verb phrase. This maps to the XCOMP attribute of the corresponding f-structure, whose value is the f-structure with PRED value “meet<Jack, Gina>”, thus the Ceptre predicate `meet jack gina` would be of type `action`, and the Ceptre predicate corresponding to the complete f-structure would be `go jack (meet jack gina)`, declared as `go character action : pred`. The XCOMP attribute denotes an open predicative complement whose subject is determined by an external argument, in this case “Jack”, which is the subject of the main verb. This approach would be able to handle sentences such as “Jack wants to steal the coins”, also sentences with a closed complement argument\(^2\) such as “Jack thinks the giant is sleeping”.

Extensions can be made to this interface for Ceptre by providing support syntactic sugar of the type $A \rightarrow B$ which is equivalent to $A \rightarrow A \otimes B$. Consider `rule/0' : need C O -o go C store`. We note that a more accurate representation of the narrative action should account for the preservation of `need C O` on the right hand side since it could be argued that the character still needs the object O when they go to the store (since they don’t yet own the item). Then, the new formulation of `rule/0'` should be written as

\[
\text{rule/0'} : \text{\$need C O -o go C store}
\]

to represent that the updated global state after this narrative action still includes the state where the character needs a certain object.

\(^2\)A clausal argument with its own subject
Figure 1
IF Jack has a cow THEN Jack sells the cow AND Jack goes to the market.

IF Jack is at home AND Jack goes to the market THEN Jack is at the market.

WHEN Jack is at the market AND Jack sells the cow THEN Jack has beans

**Figure 2**

```plaintext
% jack_out.cep
class character : type.
class location : type.
object0 : type.

cow : object0.
bean : object0.
jack : character.
market : location.
home : location.

go character location : pred.
at character location : pred.
have character object0 : pred.
sell character object0 : pred.

stage begin = {
  rule/1
  : go jack market * at jack home -o at jack market.
  rule/2
  : at jack market * sell jack cow -o have jack bean.
  rule/0
  : have jack cow -o sell jack cow * go jack market.
}

#trace _ begin init.
```

**Figure 3**
IF Abby needs candy THEN Abby goes to the store.

WHEN Abby goes to the store THEN Abby is at the store.

WHEN Abby is at the store AND James is at the store THEN Abby likes James.

WHEN James is at home AND James is hungry THEN James needs food.

```plaintext
% abby_out.cep
character : type.
location : type.
object0 : type.

abby : character.
james : character.
store : location.
home : location.
candy : object0.
food : object0.

go character location : pred.
at character location : pred.
like character character : pred.
hungry character : pred.
need character object0 : pred.

stage begin = {
rule/0
: need abby candy -o go abby store.
rule/1
: go abby store -o at abby store.
rule/2
: at james store * at abby store -o like abby james.
rule/3
: at james home * hungry james -o need james food.
}
#trace _ begin init.
```

Figure 5
"Jack goes to meet Gina."

**Figure 6**

A natural language interface for Ceptre 60.
CHAPTER 7

Conclusion

In this thesis, we have approached the problem of designing a natural language interface from a linguistically motivated perspective - by means of formal linguistic theory. The science of language, from its modern roots in Saussure’s Course in General Linguistics to Bloomfieldian Structuralism led to the development of the Distributional Hypothesis, as illustrated in the beginning chapters, which provides a legitimate foundation for statistical NLP. We reject this methodology, returning to Chomskyian Generative Grammar which allows a more concrete understanding of a syntax-semantics interface. This interface is further emphasized in the development of Lexical Functional Grammar, where the PRED attribute of the functional structure guides semantic assembly. The use of linear logic in glue semantics mirrors the use of linear logic in narrative formalization, with the resource-conscious logic acting as the mechanism for assembly in both cases. The ability of a natural language expression to be represented in terms of a linear logic formula in the LFG framework motivates our use of it in our natural language interface, which depends on the XLE LFG parser to generate an f-structure through the correspondence architecture in order to map from the f-structure to a ζ-structure, another representation of the semantic information encapsulated by the syntax-semantics interface of LFG. An input file consisting of natural language sentences corresponding to Cepvre rules can then be transformed into a partial Cepvre program requiring user specification of initial context. This interface thus allows non-programmers to utilize the Cepvre specification language to design and analyze narratives. In this way, we show that the use of a formal linguistic theory can simplify the process of designing a natural language interface.
Bibliography


IEEE.