Technical Report No. 2009-555
An Exploration of Semantic Formalisms - Part II: A Brief Survey of Other Semantic Formalisms[*]

Craig Thomas
School of Computing, Queen's University
Kingston, Ontario, Canada K7L 3N6

July 6, 2009

Abstract

A previous paper examined Montague's Intensional Logic, the First Order Predicate Calculus and Jackendoff's Conceptual Structures. The purpose of this paper is to examine other semantic formalisms, and to explore whether or not any of these formalisms has a degree of user-friendliness and expressiveness that would make it appropriate for a generalized semantic tool that would be applicable to any linguistic domain.

Contents

1 Introduction

In a previous technical report (Thomas, 2009), we explored three different semantic formalisms, and classified them according to the type of semantic theory they were based upon. The distinction between reference or meaning theories provides a rough guideline as to the set of properties one may expect from a given formalism, and provides insight as to

[*]This research was conducted with funding assistance from the Natural Sciences and Engineering Research Council of Canada.

what expressive limitations may occur (Davis and Gillon, 2004). For example, Intensional Logic developed by Montague (1974) and the First Order Predicate Calculus share roughly the same sets of expressive properties, as each relies on model and truth theory. While at first glance one appears to be unrelated to the other, this dissimilarity is ultimately explainable by differences in the underlying models for each formalism. Conceptual Structures developed by Jackendoff (1990) on the other hand, exhibits a different set of properties and expressive limitations. Based on a cognitive model of meaning, Conceptual Structures succeed in expressing knowledge using structures and primitives theoretically universal and innate to every human mind (Jackendoff, 1983; Wierzbicka, 1996). Unfortunately, none of the formalisms mentioned are capable of the coverage, precision and user-friendliness necessary to be considered a universal semantic tool. While each is being actively researched and expanded, it is obvious that the complexity associated with each system makes it difficult for the average computational linguist to use.

1.1 Other Semantic Systems

In addition to these, there is an abundance of computational systems that contain some form of semantic representation. One list of natural language processing systems compiled by Zock and Adorni (1996) has continued to grow and is now in its fourth incarnation, providing descriptions and links to over 200 systems (Bateman and Zock, 2008). While not every system on the list makes use of a semantic formalism, an impressive number of systems contain some form of semantic representation, be it TALE-SPIN's text planning component that makes use of Conceptual Dependency (Meehan, 1977), or SLANG's use of semantic preselections in a systemic grammar (Patten, 1988).

In general, these systems may be broken down into two categories as described by Levison and Lessard (1992): systems developed to fulfill a particular set of needs, and systems developed to realize a particular linguistic theory. While systems designed for the former purpose may not have wide application outside of their specific linguistic domain, systems developed to realize a particular linguistic theory may be, by their very definition, more universal in nature.

1.2 Goal of the Paper

The purpose of this paper is to examine semantic formalisms that differ from First Order Predicate Calculus, Intensional Logic and Conceptual Structures. We will consider the formalisms of Case Grammar, Conceptual Dependency, Discourse Representation Theory, Lexical Functional Grammar, Semantic Networks, Systemic Grammar, the Generative Lexicon and Rhetorical Structure Theory. This paper will examine whether or not any of these formalisms has a degree of user-friendliness and expressiveness that would make it appropriate for a universal semantic tool that would be applicable to any linguistic domain. The basic machinery of each formalism will be demonstrated through a series of simple linguistic examples. More complex examples will be used to explain how each formalism either lacks coverage or contains undue complexity that would diminish its user-friendliness.

2 Other Formalisms

2.1 Case Grammar

One of the first and earliest attempts at characterizing the the semantic content of a given sentence was developed by Fillmore (1968)[2]. Fillmore recognized that a semantic type of role labelling was needed, since surface structure *grammatical function* labels do not provide a clear semantic account of the participants involved in a given proposition[3]. For example, consider the sentence "Brutus stabbed Caesar". Here the subject of the verb is "Brutus", and the direct object of the verb is "Caesar". The terms *subject* and *direct object* are used to label the grammatical function of syntactic constituents, and have no semantic impact. A problem arises if the sentence were to be transformed to the passive: the grammatical function of each constituent may change. For example, in the passive "Caesar was stabbed by Brutus", the subject of the verb is now "Caesar" and the direct object of the verb is now "Brutus". Fillmore recognized from a semantic standpoint that the underlying proposition and its parameters remain the same, regardless of whether the surface form is stated in the active or passive voice. In both "Brutus stabbed Caesar" and "Caesar was stabbed by Brutus", the proposition "stab" requires an entity that will perform the stabbing, and an entity that will be the victim of the action. Fillmore's goal was to create a label system that would account for these semantic roles, and would allow for a mapping between the semantic participants and the surface structure grammatical functions.

From a generative point of view, Fillmore theorized that the deep structure formation of an utterance in any given language starts with a modality M and a proposition P. Of these two objects, Fillmore was mostly concerned with the expansion of propositions, which yield a verb and a set of cases. For example, as depicted by Fillmore (1968), the proposition P expansion is written as:

$$\mathrm{P} \quad \rightarrow \quad \mathrm{P} + \mathrm{V} + \mathrm{C}_1 + \ldots + \mathrm{C}_n$$

The rule above states that a proposition may recursively give rise to another proposition P, a verb V, and a number of *cases* denoted $\mathrm{C}_1 \ldots \mathrm{C}_n$ where each C_i represents a unique noun that fulfills a particular θ-role. There are six original θ-roles that Fillmore discussed: agentive, dative, instrumental, factitive, locative and objective. The *agentive* A indicates an animate actor who carries out a particular action. The *dative* D indicates an animate object that is affected by the action. The *instrumental* I indicates the use of a non-animate object that is the cause of the action. The *factitive* F indicates the creation or resulting object of the action. The *locative* L indicates the location of the action, and the *objective* O is a general "catch all" case that indicates an object that is involved in the action. Over time, different cases have emerged as new semantic roles are either discovered or further explored. As Winograd (1983) points out, Fillmore himself refined the set of base cases resulting in: *agent*, *counter-agent*, *object*, *result*, *instrument*, *source*, *goal* and *experiencer*.

Coming back to propositions, one valid expansion of P may be V + D + A. The environment created by this particular expansion is called the *case frame*. The purpose of the case frame is to dictate the lexical selection of verbs and nouns based upon the features and θ-roles that are available to be filled. For example, this particular expression says that a

[2]Fillmore (1968) uses the term "deep cases" to refer to what we will call θ-roles. Different papers use slightly different terms to refer to the general concept of the θ-role. We will use θ-role to mean the same as thematic role, thematic function, semantic role, deep case or theta-role.

[3]Grammatical functions simply describe the syntactic features of a sentence (Jurafsky and Martin, 2000).

verb, indicated by V, is involved in the sentence along with a *dative* noun indicated by D, and an *agentive* noun indicated by A. In order to choose appropriate nouns and verbs during the generative process, the lexicon is consulted. For verbs, each lexical entry includes a set of *frame features* that indicate the case for each of its formal parameters. For example, the verb "stab" accepts two formal parameters, and may have a frame feature that looks like the following:

+ [_____ D + A]

The frame feature above states that the verb "stab" may fit a case frame that requires *dative* and *agentive* nouns. The blank line indicates the position where the verb would be inserted. In the example of V + D + A, the verb "stab" is a candidate, since its frame features allow it to be placed in the environment. For nouns, each lexical entry has an associated set of *semantic features* that are used to indicate various properties of the word. For example, the noun "Brutus" may have the feature [+animate], which would make it a valid selection for any case that requires an animate object (for example, the *dative* or *agentive* cases). Thus, one possible selection of nouns and verbs for the expansion of V + D + A would result in the sentence "Brutus stabbed Caesar", and would be represented by:

$$\text{stab} + \text{Brutus}_D + \text{Caesar}_A$$

Fillmore (1968) developed a notation for frame features to describe several possibilities that may arise in special circumstances. One such circumstance is when a single verb may have optional parameters. For example, the verb "stab" in "Brutus stabbed Caesar" and "Brutus stabbed Caesar with a knife" may fulfill the two case frames: V + D + A and V + D + A + I. This is made possible if the instrumental case is marked optional. The resulting frame feature of the verb "stab" would become:

+ [_____ D (I) A]

The parenthesis in the frame feature above indicates optional cases. If a choice between two optional cases is required, where either the first, second or both cases may be selected, the parenthesis would overlap. For example, Fillmore (1968) points out the verb "kill" would need to fill the dative case, along with an agentive and / or instrumental case:

+ [_____ D (I◊A)]

Unfortunately, one of the major drawbacks to using case grammar has to do with the descriptive power of the set of θ-roles available. Pustejovsky (1995) has argued that the granularity of the θ-role simply does not provide an adequate level of description at the semantic level. This statement was prompted by the research of Levin and Rappaport (1986) who demonstrated that the notion of the θ-role was only useful in the most general sense when mapping semantic objects to grammatical functions. Simply put, while θ-roles provide a nice way of determining what the various semantic participants are and how they are manifested in the surface structure, there are other mechanisms involved during generation which suggest that the process responsible for assigning θ-roles to participants is ultimately more informative than the θ-role label itself. Winograd (1983) also demonstrates a limitation of case grammar, noting that there is simply not enough information to determine why certain verbs restrict various syntactic structures from forming. An example by Winograd involves the comparison of the sentences "Cinderella broke the glass" and "Cinderella polished the glass". Both "break" and "polish" have the same frame features, namely:

$$+ [\ \_\_\_\_\_\ A + O\ ]$$

However, "break" may be reformulated into an ergative, while "polish" cannot. The end result is that the sentence "the glass broke" is grammatical, while "the glass polished" is not. Case grammar alone cannot account for these differences.

Despite these limitations, case grammar has had a significant impact on computational systems. Some systems that use case grammar include: ERMA (Clippinger, 1975), PLANES (Waltz and Goodman, 1977), GIST (Swartout, 1982) and GENNY (Maybury, 1989).

2.2 Semantic Networks

Semantic networks have existed since the late nineteenth century, one of the earliest forms being that of *existential graphs* developed by Charles Peirce (Lehmann, 1992). Existential graphs were meant to depict relationships in the first order predicate calculus. Lines were used to represent semantic individuals and nodes were drawn to represent relationships[4]. The use of these types of existential graphs proliferated in the twentieth century, and they were further expanded to encompass work in psychology and linguistics. Eventually they became an important system of semantic representation for artificial intelligence, machine translation and question answering projects (Sowa, 1987).

As noted by Sowa, the particulars regarding the style of each graph may vary, but the underlying principles remain the same with present day graphs: nodes depict semantic individuals or concepts, and a series of arcs are drawn between them to represent relationships. The *relational graph* is the simplest type of network, and is so named since the arcs that connect conceptual nodes are understood to represent specific relationships (Sowa, 1987). Usually, the relationships used to link semantic concepts together are θ-roles, and are formally called *case links* (Lehmann, 1992). An example relational graph for the sentence "Brutus stabs Caesar viciously" is provided in figure 1. The semantic network given in this example represents a *verb-centered* relational graph, since the verb is shown as the focal point for all of the semantic relationships (Sowa, 1987).

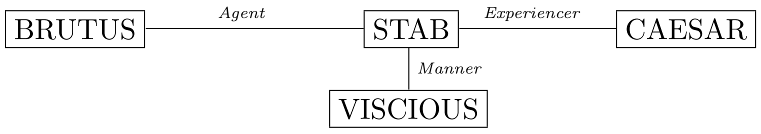


Figure 1: An example relational graph representing the sentence "Brutus stabs Caesar viciously".

While relational graphs appear to be straightforward and easy to understand, there are various problems. Since semantic networks were first conceived to convey formulas in the first order predicate calculus, they are subject to various expressive limitations. For example, quantification is restricted to the use of the existential quantifier $\exists$, and the only type of logical operation supported is conjunction (Sowa, 1987). In addition to these difficulties, the scopes of various relationships pose problems for verb-centered graphs. For example, consider the sentences "Brutus sang strangely" and "strangely, Brutus sang". In a verb-centered relational graph, only the first sentence may be expressed, as shown in figure 2. Representing the latter sentence is impossible without a way of referring to the sentence as a whole.

[4]As Lehmann (1992) observes, this notation is "just the opposite of modern practice".

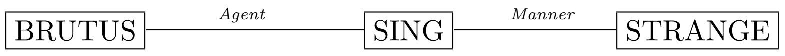


Figure 2: An example relational graph representing the sentence “Brutus sang strangely”.

A different type of semantic network known as a *propositional network* was developed in order to deal with shortcomings in verb-centered relational graphs (Sowa, 1987). The main difference between the two types of network is that propositional networks allow nodes to represent entire propositions, allowing for the expression of sentences that required fully formed expressions to become nested arguments to other propositions (Sowa, 1987). For example, as stated above, simple verb-centered graphs could not properly handle the scope requirements of modifiers in sentences such as “strangely, Brutus sang”, where the manner $\boxed{\text{STRANGE}}$ needs to refer to the entire sentence and not just the $\boxed{\text{SING}}$ action that occurred. Since propositional networks can relate fully formed propositions to one another, this particular sentence can be trivially expressed as in figure 3.

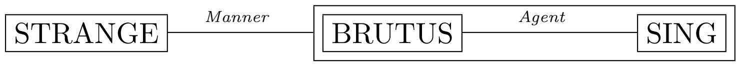


Figure 3: An example propositional network representing the sentence “strangely, Brutus sang”.

These semantic networks are examples of *assertion networks*, where the relationships between nodes are effectively designed to express true propositions (Lehmann, 1992; Marinov and Zheliazkova, 2005). Other forms of semantic network include *type hierarchies*, where objects are arranged based upon their types (Sowa, 1987). These types of networks are known as *inheritance networks*, since subtypes may inherit properties from their parent nodes[5] (Daelemans *et al.*, 1992; Lehmann, 1992). These types of semantic networks are used to assert properties between sets of objects, or to indicate class membership. This type of information is of a *higher order* than assertion networks, since it describes relationships between types, and not just information relating to individuals (Sowa, 1987). For this reason, assertion networks and inheritance networks are usually kept as separate entities, since they attempt to convey different types of information (Lehmann, 1992).

Inheritance networks typically use IS-A relationships for types (Marinov and Zheliazkova, 2005; Sowa, 1987). For example, a simple inheritance network used for a Shakespearean setting is given in figure 4. With this inheritance network, it is possible to reason about class membership, and vary the syntactic form of a proposition accordingly. For example, in the above network, “Caesar” is assigned the types “Roman” and “dictator”, two distinctions which could contribute to the realization of “Brutus stabbed a Roman dictator” from the first order predicate calculus formula STAB(BRUTUS, CAESAR).

While inheritance networks are used to convey a great deal of type information, they are subject to a number of limitations. First, with many tasks it becomes necessary to group higher order properties into various ontologies, and to use only a subset of those ontologies within a semantic network. The proper selection of ontologies is important when performing

[5]Inheritance networks are also known as *taxonomic networks* and *definition networks* (Sowa, 1987; Marinov and Zheliazkova, 2005).

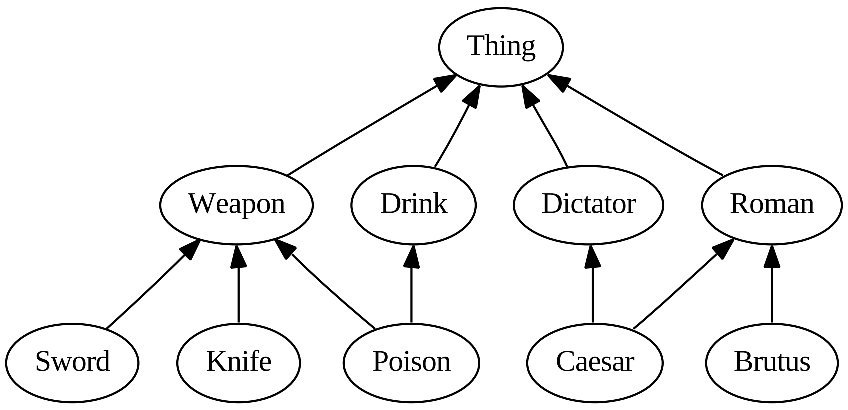


Figure 4: An example type hierarchy for a simple Shakespearean world.

semantic tasks, since it directly impacts the quality of semantic information available[6]. For example, depending on the timeperiod in a Shakespearean setting, it may be important to indicate that CAESAR IS-A SENATOR, but not CAESAR IS-A DICTATOR. Additionally, since many types are not directly comparable to one another, a *partial order* results[7] (Sowa, 1987). This may cause some confusion if the partial ordering is not clearly indicated (Sowa, 1987). For example, a RACE-CAR may be a subtype of CAR, but not usually a subtype of RACE.

The type of semantic network given in figure 4 is an example of a *multiple inheritance* network (Lehmann, 1992). As is obvious from the example, child nodes may inherit information from more than one parent. This type of inheritance network can create problems when multiple levels of inheritance lead to conflicting sets of properties (Lehmann, 1992). One solution to the multiple inheritance problem is to simply require that each node in an inheritance network have at most one parent, thereby eliminating the problem altogether (Lehmann, 1992). However, it has been argued that such *single inheritance* networks do not provide an accurate representation of the complexities found in the real world, where a single object may belong to a number of different parent concepts (Daelemans *et al.*, 1992).

An alternate solution to this problem is to incorporate exceptions in a semantic network through the process of *defeasible inheritance* (Lehmann, 1992). Lehmann's example is that of Tweety bird, who is a penguin, and its relationship to the flyer and non-flyer properties depicted in figure 5. In this instance, a penguin inherits properties relating to the bird type with the exception of the flyer property, which is cancelled out by the IS-NOT-A link from penguin to non-flyer[8].

[6]An ontology is a collection of properties that belong to a particular category (Lehmann, 1992).

[7]Two types that may not be directly compared are RACE and CAR. As a result, it is not possible to say that RACE $<$ CAR or CAR $<$ RACE (Sowa, 1987).

[8]Notice that this problem could be solved by breaking the Bird type into two distinct sets: Non-Flying-Bird which does not inherit the Flyer type, and Flying-Bird which does. The Penguin type would then be a member of Non-Flying-Bird, thus avoiding complications with inheritance.

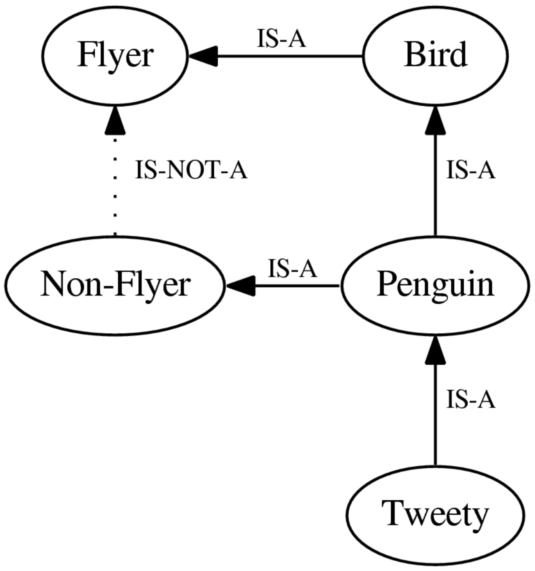


Figure 5: The Tweety bird inheritance network, reproduced from Lehmann (1992). The IS-NOT-A inheritance link between flyer and non-flyer has the effect of cancelling out the flyer property that Tweety would inherit through being a member of the bird type.

While cancellation solves the problem surrounding the need to include exceptions for some classes, researchers such as Brachman (1985) criticize this approach on the basis that overzealous use of cancellation leads to any truth about any type. For example, Brachman states that "[it leaves] us with possibilities like 'a rock is an elephant, except that it has no trunk, it isn't alive, it has no legs ...,' and even more outrageous semantic anomalies that we dare not imagine here". An alternative to cancellation is to incorporate orderings along parent links. In such *prioritized networks*, the parent with the lowest ordering that points to an overlapping property is "the winner"; any other links which lead to a conflicting property that have a higher ordering are ignored (Daelemans *et al.*, 1992). A third alternative is an *orthogonal network* scheme. In order to avoid inheritance conflicts, this solution assumes there are several different single inheritance networks stacked on top of each other, and that each node is divided into a number of different partitions. The mechanism works by enforcing the rule that no child node may inherit more than one property from the same layer. In other words, each partition of the child node must inherit properties from different networks (Daelemans *et al.*, 1992).

Additional issues with semantic networks deal with their inability to provide treatment for extensional phenomena (Johnson-Laird *et al.*, 1984). The example used by Johnson-Laird *et al.* is made using inference. Semantic inheritance networks are able to express the fact that "Fido is a dog" directly from Fido IS-A Poodle and Poodle IS-A Dog. This type of inferencing works correctly, since it only requires information relating to the concepts and their relationship to one another. In other words, this example deals entirely with intensional information. Unfortunately, the same cannot be said when extensional information is needed to make valid inference. The example used by Johnson-Laird *et al.* is that of three people, A, B and C. Consider the sentences "A is on B's right" and "B is on C's right". The inference "A is on C's right" requires knowledge about the real world, not just about concepts. In other words, we need to know extensional information. To show the limitation involved, imagine the instance where all three participants are seated around a circular table, as opposed to a rectangular table. While it may be true that both “A is on B’s right” and “B is on C’s right” are valid, the overall inference “A is on C’s right” is no longer true.

Finally, researchers such as Busa *et al.* (2001) have discussed the problem of IS-A overloading. For example, consider the word “product”. This lexical item has a wide variety of meanings that change with context. As pointed out by Busa *et al.*, “product” may be a piece of software, an abstract entity, or it may even refer to an event. These three different meanings are all based on IS-A relationships. Unfortunately, lexical items may have a potentially large number of meanings that all depend on context. The only way of generating the correct sense of each and every word would be to express it along every single IS-A dimension possible. This exhaustive approach, as Busa *et al.* point out, of using IS-A links to represent a hierarchy of types may never be sufficient, since it would be extremely difficult to capture every single contextual use of any given word. Clearly this represents a shortcoming of semantic networks, as well as many other semantic theories.

Despite these criticisms, there is continued research into the use of semantic networks. Some computational systems that make use of semantic networks include: ACT (Anderson and Kline, 1977), KL-ONE (Brachman and Schmolze, 1985), ANALOG (Ali and Shapiro, 1993) and KALOS (Cline and Nutter, 1994).

2.3 Conceptual Dependency

Created by Schank (1972), Conceptual Dependency (CD) is a form of representation that captures the human thought behind an utterance in a natural language. The approach taken with CD and with many other systems is that language is *stratified* in the sense that it can be broken up into distinct layers or *strata* (Winograd, 1983). Each of these layers has its own formal structure and units that describe an utterance[9]. Schank recognized the fact that units at one level are distinct and separable from units at another level, and theorized that there is a language independent level of structure called a *conceptual base* that captures the meaning of an utterance. Primitives in the conceptual base are *concepts*, which carry the underlying meaning of a word or set of words[10]. Utterances in a natural language are formed by mapping a set of concepts onto a set of syntactic structures, a process which humans innately perform when they wish to communicate. Different languages are characterized by different mapping rules from concept to syntax. The underlying principle is that two utterances in a natural language are equivalent if they share the same conceptual base.

Formally, meaning in CD is carried by concepts and their relationships to one another in a *conceptualization*, which is drawn as a graph called a *conceptual dependency network* or C-diagram (Schank, 1972). Schank theorized that there were three different types of conceptual primitives: nominals, actions and modifiers. By their very nature, *nominals* are concepts that can be understood by themselves, and usually relate to nouns. Schank called these objects *picture producers* (PP), since he noticed they tend “to produce a picture of that real world item in the mind of the hearer.” As their title suggests, conceptual *actions* (ACT) express what a PP is doing, and usually relate to verbs. Conceptual *modifiers* relate to properties, and rely on the existence of a PP or ACT to be correctly interpreted. Schank identified two types of modifiers: *picture aiders* (PA), and *action aiders* (AA). Additional primitives are also available for *locations* (LOC), and for *times* (T) (Schank, 1975).

[9]In general, the units for each stratum are *sememes* at the semantic level, *lexemes* and *morphemes* at the syntactic level, and *phonemes* at the phonological level (Winograd, 1983).

[10]The notion of the concept is similar to the one used by Jackendoff (1983).

To understand how the conceptual primitives work, consider the sentence "Brutus stabs Caesar". Both "Brutus" and "Caesar" belong to the PP category, as they are both objects that can be understood by themselves. The verb "stab" is an action and thus belongs to the ACT category; however, it would more likely be represented using a combination of some of the eleven primitive ACTs defined by Schank (1975) (this is discussed in more detail later). If we were to modify the sentence to "yesterday, Brutus stabbed Caesar viciously", the term "yesterday" would be classified as belonging to the T category and "viciously" would be classified as an AA.

Another key ingredient in CD is the notion of the *dependency.* In simple terms, dependencies are relationships that exist between concepts (Schank, 1972). As Schank explains, a dependency consists of a governor and a dependent. Governors are PPs and ACTs, while dependents are PAs and AAs. Dependents are so named, since they have no meaning on their own; they serve only to modify other concepts. For example, consider the sentence "Caesar is dead" represented in figure 6[11].

$$\text{Caesar} \quad \leftrightarrows \quad \text{HEALTH(-10)}$$

Figure 6: An example of the conceptual dependency diagram for the sentence "Caesar is dead".

Here, *Caesar* and *HEALTH* are members of a two-way attributive dependency, indicated by $\leftrightarrows$ (Schank, 1972). This dependency shows that the PP Caesar is dependent on the PA Health in a predication. The dependence relationship is two-way since both *Caesar* and *HEALTH* must be present for the attributive predication to exist. Schank (1975) describes the state HEALTH by a numerical scale, which ranges from -10 to +10, with each number on the scale representing a state of health. Here a HEALTH of -10 indicates the state of death. Examples of other numerical rankings for HEALTH include: -9 sick, -2 under the weather, 0 all right, +7 tip top and +10 perfect health. Other states are also represented using numerical scales, with examples of states being FEAR, ANGER, MENTAL STATE, PHYSICAL STATE, etc. Schank indicates that a wide variety of states can be expressed by combining many of these 'primitive' states together into one complex structure.

The formation of valid C-diagrams is governed by a fixed set of *conceptual syntax* rules (Schank, 1975). In figure 6, the rule stating that PP and PA are dependents was invoked. For a C-diagram involving more complex rules, consider the sentence "Caesar walked home from the senate", drawn in figure 7. With this example, several primitive concepts are combined to provide the final C-diagram. The concepts *home* and *senate* are involved in an ACT *PTRANS* that has a two-part dependency between objects designated *from* and *to*. This portion of the structure expresses the *directive case*, represented by the *D* on the diagram (conceptual case is discussed in more detail below) (Schank, 1972). The ACT PTRANS represents the physical change of location that an object undergoes, and it is one of the eleven primitive ACTs defined by Schank (1975). Other primitive ACTs that Schank defines are: SPEAK, PROPEL, MOVE, INGEST, EXPEL, GRASP, ATTEND, MTRANS, MBUILD and ATRANS. Many of these ACTs are fairly intuitive. However, a few such as ATRANS are not. ATRANS is the act of changing an abstract relationship, ATTEND is the act of sensing a stimulus, MTRANS is the act of transferring information,

[11]The Conceptual Dependency examples provided in this section have been modified from the examples provided by Schank (1975), to keep with the Shakespearean theme.

and MBUILD is the act of creating a new thought out of others. The ⇂ connecting the objects *Caesar* and *home* is qualified with POSS, indicating that one object possesses the other. In this case, we are referring to "Caesar's home". The ↔ connecting *Caesar* and *go* is similar to the dependency in figure 6, but in this instance is used to show an action predication dependency, namely that *Caesar* is performing the ACT *go*. The *p* over the ↔ is a temporal marker used to indicate that the action occurred at an unspecified point in the past. All of the items combine to form the conceptualization of "Caesar walked home from the senate".

$$\begin{tikzcd}
 & & & & \text{Caesar} \arrow[d, "\text{POSS}"] \\
\text{Caesar} \arrow[r, leftrightarrow, "p"] & \text{PTRANS} & \text{Caesar} \arrow[l, "o"'] & \text{D} \arrow[l] \arrow[r, "\text{to}"] \arrow[dr, "\text{from}"', tail reversed] & \text{home} \\
 & & & & \text{senate}
\end{tikzcd}$$

Figure 7: An example of the conceptual dependency diagram for the sentence "Caesar walked home from the senate".

The previous example introduced a new structure that contained two dependents: the *from* and *to* of a particular action. Schank (1972) describes this requirement as an example of a *conceptual case*, which is similar to the type of case Fillmore (1968) introduced. A conceptual case specifies the dependents that are required for a given ACT, or in other words, it defines the valency requirements of semantic concepts. Schank demonstrates that there are four different conceptual cases: objective, directive, recipient and instrumental. In the conceptualization of "Caesar walked home from the senate", a directive case was used. The recipient case may be observed in the sentence "Brutus took a knife from Caesar" illustrated in figure 8.

$$\begin{tikzcd}
\text{Brutus} \arrow[r, leftrightarrow, "p"] & \text{PTRANS} & \text{knife} \arrow[l, "o"'] & \text{R} \arrow[l] \arrow[r, "\text{to}"] \arrow[dr, "\text{from}"', tail reversed] & \text{Brutus} \\
 & & & & \text{Caesar}
\end{tikzcd}$$

Figure 8: An example of the conceptual dependency diagram for the sentence "Brutus took a knife from Caesar".

In figure 8, the same structure is used as in figure 7, except that this new use reflects the *recipient case* as marked by the *R*. The recipient case is used to show that *Brutus* initiated a transfer *PTRANS* that involves a *knife* moving from the donor *Caesar* to the recipient *Brutus*. Schank (1972) indicates that a trans mechanism is involved in many different verb senses such as 'steal', 'give', 'take', 'sell', etc. The same trans can be used in all of these complex conceptualizations, as Schank argues that it is the placement of the *from* and *to* objects and the initiator object of the ACT that account for the differences between verbs such as 'buy' and 'sell'. The *o* above the ← indicates that the recipient case involving the knife is the *objective* of the PTRANS action. All of these items make up the conceptualization of "Brutus took the knife from Caesar".

In addition to defining various forms of ACT, Schank (1975) discusses the fact that concepts may rely on other concepts, and that underspecified actions may result in a generic function such as DO. This can be demonstrated through *causality*. For example, consider the sentence "Brutus killed Caesar", represented in figure 9. Here the cause of Caesar's death is some action that is initiated by Brutus. Since details of the action are not available, the ACT *DO* is recorded to demonstrate that some action was done by Brutus, but that the details of the action are not available. The rest of the structure is straightforward, with the movement of PHYS.ST. from >-10 to -10 indicating that Caesar was once at a state above death, but came to be at the physical state associated with death. The $\upuparrows$ indicates that the movement of Caesar's physical state is caused by the action DO that is dependent on Brutus. Had a knife or any other physical object or detail been involved, that detail would have been captured in the diagram, instead of the generic DO action.

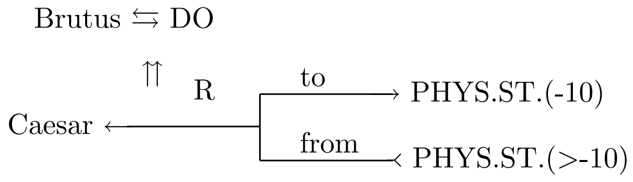


Figure 9: An example of the conceptual dependency diagram for the sentence "Brutus killed Caesar".

While CD appears to provide a rich set of mechanisms for representing meaning, there are several published accounts that suggest that CD theory is inadequate as a universal semantic formalism. One account comes from Riesbeck (1975), whose work revealed that CD has theoretical problems with its representation of PPs. As described by Schank (1975), a PP is meant to be a pointer to a series of attributes that describe a given object. What Riesbeck points out is that there is no well defined theory of how PPs relate to one another, or how the internal structure of a PP's attributes would be organized. For example, Riesbeck asks questions such as "how does the concept of a chair relate to that of a table, what does it mean to use a cup for a hammer, and so on", all of which lack answers in CD. While Riesbeck does not see this as a reason to halt research into CD theory, Dunlop (1990) draws attention to this issue in a footnote, and notes that historical attempts to successfully define internal structure for objects like PPs has proven to be difficult.

Dunlop (1990) goes on to present several arguments against CD theory from a psychological perspective, one of which deals with the issue of descriptive insufficiency. Dunlop observes that the eleven primitive ACTs are unlikely to adequately capture information relating to various real world actions. This observation is based on the stipulation made by Schank (1975) that all primitive ACTs may infer information that is not explicitly carried by the C-diagram. For example, if an object is PTRANSed, it is inferred that the object is now LOCated at the destination. Dunlop notes that while these types of inferences are useful, any action involving the description of a social institution or social convention cannot be properly expressed, since many of the intended inferences are simply not conveyed by the primitives at hand. Schank admits that there are limitations surrounding representations such as "Brutus kissed Claudia". Here, the realization in CD results in a structure that is best paraphrased by "Brutus moved his lips towards Claudia". Clearly this does not capture the essence of what a kiss actually is; in this case, the expression of an intimate feeling.

One final observation which prevents Conceptual Dependency from being a universal semantic formalism is the issue of complexity. The underlying information carried in a C-diagram are meant to be fully formed and *complete* conceptualizations. Unfortunately, one requirement by Schank (1975) states that ACTs *always* require the instrumental case. As Dunlop (1990) explains, this requirement causes problems, since the instrumental case requires fully formed concepts to carry meaning. Consider the sentence "Brutus stabbed Caesar with a knife", represented in figure 10.

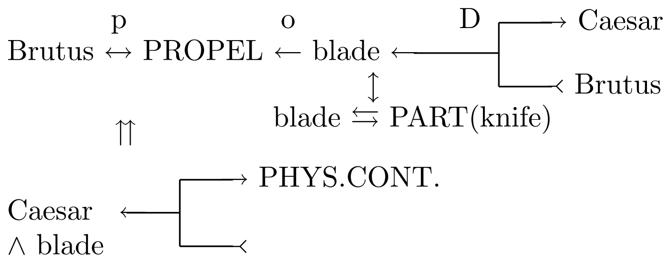


Figure 10: An example of the conceptual dependency diagram for the sentence "Brutus stabbed Caesar with a knife".

While the conceptualization appears to be complete, it is in fact insufficient. In particular, an instrumental case is needed to describe the fact that it was Brutus' arm that moved the knife towards Caesar. In fact, Dunlop states that in order to explain the instrumental case of a given ACT, there exists the presupposition that there are an *infinite number* of ACTs that make up the required conceptualization, making it impossible to express. For example, the movement of Brutus' arm requires another instrumental case, namely that of Brutus' arm muscles causing the arm to move. That instrumental case requires another instrumental case, one that explains that impulses from the brain caused the muscles in Brutus' arm to actually move, and so on. Obviously the C-diagrams in question must be of finite size, as it is both impractical and impossible to infinitely specify all of the instrumental cases required for even the simplest of concepts. While Schank himself noted and dismissed this problem as merely an inconvenience, Dunlop hints that the problem is more serious, remarking that "if the performance of any given one ACT presupposes an infinity of ACTs, no ACTs will be possible at all". We are forced to conclude that using conceptual dependency to express simple statements may appear clear, but trying to cleanly and precisely represent some complex concepts such as "Cassius bet Brutus that Caesar would not survive the assassination attempt later that afternoon" becomes much more difficult. This particular issue coupled with descriptive insufficiency makes it unlikely that CD theory will become a universal semantic formalism without additional research and expansion.

Several computational systems make use of Conceptual Dependency including: BABEL (Goldman, 1975), TALE-SPIN (Meehan, 1977), KAFKA (Mauldin, 1984) and ViRbot (Savage *et al.*, 2009).

2.4 Discourse Representation Theory

Developed by Kamp (1981), Discourse Representation Theory grew up as an extension to first order predicate logic. Kamp was interested in anaphoric relationships, and how they posed problems to semantic analysis and representation where multi-sentence utterances were concerned. The classic examples involve well known "donkey anaphora", where pronouns are involved in binding relationships across sentences. For example, consider the following textual fragment: "Pedro owns a donkey. It is grey." This multi-sentence utterance poses a problem for standard first order predicate logic, as each sentence in isolation only represents a fragment of the full meaning. In particular, the problem of associating the pronoun "it" with the indefinite "a donkey" is at the heart of the issue. Standard first order predicate calculus offers no mechanisms to accomplish this. In order to tackle this problem, Kamp (1981) introduced a new type of structure called a Discourse Representation Structure (DRS) that is used to keep track of discourse information over time. Figure 11 contains a simple DRS for the sentence "Brutus stabbed Caesar".

$$\boxed{\begin{array}{c}\text{x y}\\[4pt] \text{BRUTUS(x)}\\ \text{CAESAR(y)}\\ \text{STAB(x, y)}\end{array}}$$

Figure 11: An example of a Discourse Representation Structure for the sentence "Brutus stabs Caesar".

This example demonstrates some of the main features of every DRS. Simply put, each DRS contains a set of *discourse referents* which are essentially a set of variables that correspond to some real world referent, and a set of *conditions* that describe the identities of each discourse referent (Kamp, 1981, 1988; Kamp and Reyle, 1993). In this example, variables x and y are used as referents to "Brutus" and "Caesar" respectively. Notice also that identities are described by conditions that are expressed using familiar first order predicate calculus notation[12].

When new sentences are encountered, the set of discourse referents is carried forward, and the scope of the previously introduced variables is extended to cover the new sentence (Kamp, 1981). Kamp and Reyle (1993) state that there is a construction algorithm that is responsible for assembling the set of DRSs from the syntax of the sentence. For example, consider the sentences: "Brutus stabs Caesar. He dies". The process of construction for the example is indicated below[13]:

[12]Kamp (1981, 1988) makes use of a different notation for expressing conditions. However, the examples here utilize the first order predicate calculus notation as employed by Geurts and Beaver (Winter 2008).

[13]The formal expression of each construction rule and how it relates to the syntax of the original sentence is not stated here. Rather, it is our hope to convey the general idea of how these rules form appropriate DRSs based upon simple linguistic examples. More detailed information may be found in Kamp (1981) and Kamp and Reyle (1993), which explains the original single step version of DRT. The construction example shown here follows the newer two step version of DRT employed by Geurts and Beaver (Winter 2008).

$$\underset{\text{(a)}}{\boxed{\begin{array}{c} \text{x y} \\ \\ \text{BRUTUS(x)} \\ \text{CAESAR(y)} \\ \text{STAB(x, y)} \end{array}}} + \underset{\text{(b)}}{\boxed{\begin{array}{c} \underline{\text{u}} \\ \\ \text{DIE(u)} \end{array}}} \rightarrow \underset{\text{(c)}}{\boxed{\begin{array}{c} \text{x y u} \\ \\ \text{BRUTUS(x)} \\ \text{CAESAR(y)} \\ \text{STAB(x, y)} \\ \text{u = y} \\ \text{DIE(u)} \end{array}}} \rightarrow \underset{\text{(d)}}{\boxed{\begin{array}{c} \text{x y} \\ \\ \text{BRUTUS(x)} \\ \text{CAESAR(y)} \\ \text{STAB(x, y)} \\ \text{DIE(y)} \end{array}}}$$

Figure 12: An example of the DRS construction process for the sentences "Brutus stabs Caesar. He dies".

Figure 12 demonstrates the progression used to create the DRS of the overall sentence. In (a), the original DRS representing "Brutus stabs Caesar" is used with the addition of a new DRS in (b) representing "he dies". The variable *u* in (b) is used to represent the pronoun "he", and is underlined to signal the fact that it requires resolution in the encompassing context (Geurts and Beaver, Winter 2008). In order to build a proper DRS, the two previous DRSs are merged to create (c). At this point in the construction, the DRS is examined to see which of the variables may be equivalent to previously known information. Through the use of identity (the = operator), the discourse referent *u* is identified as being Caesar. Once combined, the DRS is simplified by eliminating extraneous variables, the result of which is shown in (d). The simplification is possible since *u* and *y* are equivalent statements. The variable *u* is removed from the DRS entirely, and *y* is substituted as needed.

What is left unsaid with the previous example is how it came to be known that *he* and *Caesar* were the same semantic individual. In other words, the process that decides which variable "he" belongs to has not been formally defined. Kamp (1981) stipulates that such processes could operate by a variety of mechanisms, and provides examples where gender could provide clues as to how the association may work. While Kamp theorizes that the basic mechanism for anaphoric reference resolution should be based upon a speaker's understanding, as of yet there is no explicit knowledge of how speakers operate in regards to this issue. Kamp deliberately leaves this item open for future researchers to resolve when linguistic knowledge is sufficient to describe this process.

Discourse Representation Structures can implement the same types of logical statements that are available to the first order predicate calculus. For example, logical implication, negation, conjunction, disjunction and quantification are possible (Kamp and Reyle, 1993). For example, consider the sentence "Brutus does not stab a dictator." This example is represented in figure 13.

$$\boxed{\begin{array}{c} \text{x} \\ \\ \text{BRUTUS(x)} \\ \neg\ \boxed{\begin{array}{c} \text{y} \\ \\ \text{DICTATOR(y)} \\ \text{STAB(x, y)} \end{array}} \end{array}}$$

Figure 13: A DRS representing the sentence "Brutus does not stab a dictator."

In figure 13, several phenomena are covered. First, the existential nature of the indefinite "a dictator" is captured by the introduction of the y variable associated with the condition predicate DICTATOR. In essence, the introduction of the y variable results in an existential quantification, the same mechanism responsible for indefiniteness in the first order predicate calculus. Second, negation is introduced with the $\neg$ operator. The value of the negated expression is based upon the truth of the embedded DRS, namely that of DICTATOR(y) and STAB(x, y). Negation of these sub-expressions occurs in the normal way, namely if both conditions are true, then the overall value of the embedded DRS is false. The entire sub-expression is read as "it is not the case such that there is a dictator y, and x stabs y."

The same disjunction operator is available with DRT that was available to the first order predicate calculus, with the same truth conditional evaluation attached to it. Consider the sentence "Cassius or Brutus stabbed Caesar", which is represented by the DRS in figure 14. Similar treatment is also provided for conjunction in the sentence "Cassius and Brutus stabbed Caesar" in figure 15, and for material implication in the sentence "if Brutus has a knife, he will stab Caesar" in figure 16.

$$\boxed{\begin{array}{c} \text{x y z} \\ \\ \text{CASSIUS(x)} \\ \text{BRUTUS(y)} \\ \text{CAESAR(z)} \\ \boxed{\text{STAB(x, z)}} \quad \vee \quad \boxed{\text{STAB(y, z)}} \end{array}}$$

Figure 14: A partially formed DRS for the sentence "Cassius or Brutus stabbed Caesar."

$$\boxed{\begin{array}{c} \text{x y z} \\ \\ \text{CASSIUS(x)} \\ \text{BRUTUS(y)} \\ \text{CAESAR(z)} \\ \boxed{\text{STAB(x, z)}} \quad \wedge \quad \boxed{\text{STAB(y, z)}} \end{array}}$$

Figure 15: A partially formed DRS for the sentence "Cassius and Brutus stabbed Caesar."

$$\boxed{\begin{array}{c} \text{x} \\ \\ \text{BRUTUS(x)} \\ \boxed{\begin{array}{c} \text{y} \\ \\ \text{KNIFE(y)} \\ \text{HAVE(x, y)} \end{array}} \quad \rightarrow \quad \boxed{\begin{array}{c} \text{z u} \\ \\ \text{CAESAR(z)} \\ \text{u = x} \\ \text{STAB(x, z)} \end{array}} \end{array}}$$

Figure 16: A partially formed DRS for the sentence "if Brutus has a knife, he will stab Caesar."

Care must be taken when forming sentences to ensure that the correct predicates are being formed, and that the required referents and conditions are *accessible* to the contexts that require them. For example, if we were to extend the previous example to "Brutus does not stab a dictator. He dies", then the DRS in figure 17 would be created. With this example, the DRS is constructed as previously stated, however, the addition of the "he dies" portion of the discourse results in an interesting problem. The discourse referent variable u is introduced to stand in for "he", but the identity of u is unknown at this point. If we meant for u to be equivalent to "a dictator", then we are faced with an *accessibility* problem. Simply put, the outer context of the DRS cannot access any of the discourse referents inside of the negated DRS. Thus, we cannot simply say that u and y are equivalent, since the outer context has no access to the inner context. By contrast, the embedded DRS in the negated statement has access to the outer DRS discourse referents, allowing us to make statements such as STAB(x, y), where x refers to the greater set of discourse referents. The only possible equality of u may be to "Brutus", represented by variable x.

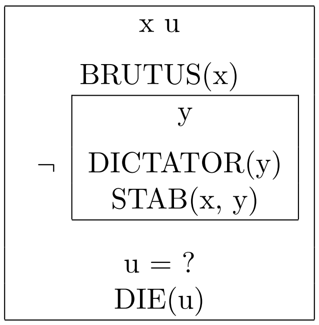


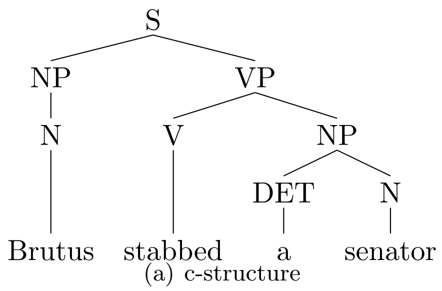
Figure 17: A partially formed DRS representing the sentence "Brutus does not stab a dictator. He dies."

While the notion of the DRS provides a powerful way to construct multi-sentence discourse, it is not without its limitations. As mentioned by Kamp (1981) above, the resolution of anaphoric references requires investigation, as the mechanism of resolution is not yet fully realized, nor is there any linguistic theory immediately available that would suggest a solution to this particular problem. Additionally, the limitations surrounding expressiveness also pose problems, since Kamp and Reyle (1993) remark that DRT is only a notational variant on the first order predicate calculus. As we have already observed in Thomas (2009), the first order predicate calculus has limitations on expressiveness that ultimately makes it a less desireable form of representation for semantic tasks that target a wide variety of linguistic phenomena. However, Kamp (1988) has defended the theory on the basis that DRT offers a transparent method of knowledge representation, and also provides an ability to restate the formal semantics of a sentence in a tractable way.

Some computational systems that make use of DRT include: ACORD (Kohl *et al.*, 1990), PRETEXTE (Gagnon and Lapalme, 1996), FOGS (Endriss and Klabunde, 2000) and HYPERBUG (Klarner, 2004).

2.5 Lexical Functional Grammar

Conceived by Kaplan and Bresnan (1982), Lexical Functional Grammar (LFG) is a type of unification grammar that expresses the grammatical relationships of surface structure constituents to their deep structure arguments in a unified and language independent way. While not originally developed to deal purely with semantics, LFG does contain references to semantic content in its 'Predicate' relationship, which is discussed further below. The formalism makes use of two different types of structures: *c-structures* which are tree representations of the sentence structure, and *f-structures* which describe grammatical functions (Kaplan and Bresnan, 1982). An example of each type of structure is represented in figure 18.



(a) c-structure

$$\begin{bmatrix} \text{Subject} & = & \begin{bmatrix} \text{Definiteness} & = & \text{Definite} \\ \text{Number} & = & \text{Singular} \\ \text{Predicate} & = & \text{`Brutus'} \end{bmatrix} \\ \text{Tense} & = & \text{Past} \\ \text{Predicate} & = & \text{`Stab}\langle(\text{Subject}), (\text{Object})\rangle\text{'} \\ \text{Object} & = & \begin{bmatrix} \text{Definiteness} & = & \text{Indefinite} \\ \text{Number} & = & \text{Singular} \\ \text{Predicate} & = & \text{`Senator'} \end{bmatrix} \end{bmatrix}$$

(b) f-structure

Figure 18: The c-structure and f-structure relating to the sentence "Brutus stabbed a senator".

While c-structures should be familiar to most computational linguists, f-structures require more explanation. The notion of the f-structure in the context of LFG stands for *functional structure* (Kaplan and Bresnan, 1982). Functional structures are used to provide a description that applies to a particular linguistic object. Each functional structure contains a set of ordered pairs, where each pair consists of an *attribute* and a *value*. Attributes are labels that are used to indicate the type of function being described. Values come in one of three forms: symbols, semantic forms or nested functional structures. For example, the functional structure for "a senator" is given in figure 19. In this example, the attribute 'Definiteness' describes whether the object is definite or indefinite (as implied by the label), 'Number' indicates the grammatical number used for morphological agreement, and 'Predicate' provides the semantic contribution of the functional structure.

$$\begin{bmatrix} \text{Definiteness} & = & \text{Indefinite} \\ \text{Number} & = & \text{Singular} \\ \text{Predicate} & = & \text{`Senator'} \end{bmatrix}$$

Figure 19: An example of a functional structure for the fragment "a senator".

The path from a c-structure to an f-structure requires several steps and several constructs. C-structures are generated using a slightly modified context-free phrase structure grammar (Kaplan and Bresnan, 1982). The context-free phrase structure rules and lexical items are modified to contain templates called statement *schemata*, which are used to build *functional specifications*. An example set of context-free phrase structure rules and their associated schemata is provided in figure 20.

$$\begin{aligned}
\text{S} &\rightarrow \underset{(\uparrow \text{Subject})=\downarrow}{\text{NP}} \quad \underset{\uparrow=\downarrow}{\text{VP}} \\
\text{VP} &\rightarrow \text{V} \quad \underset{(\uparrow \text{Object})=\downarrow}{\text{NP}} \\
\text{NP} &\rightarrow \text{DET N} \mid \text{N} \\
\text{DET} &\rightarrow \textit{a} \\
\text{N} &\rightarrow \textit{senator} \mid \textit{Brutus} \\
\text{V} &\rightarrow \textit{stabbed}
\end{aligned}$$

Figure 20: Context-free phrase structure rules with associated schemata.

To understand how a schema works, consider the right hand side of the S rule in figure 20. Located underneath the non-terminal NP is the schema (↑ Subject)=↓. The ↑ and ↓ symbols are known as *metavariables* (Kaplan and Bresnan, 1982). The metavariables determine the attributes and values that will be inherited from various locations of the c-structure. For example, the ↓ indicates that the NP will inherit a functional specification from its child node. Similarly, the (↑ Subject) indicates that the NP will pass a functional specification for a 'Subject' attribute up to the S node. In other words, the S node will have an attribute called 'Subject', which will contain the f-structure formed by the NP node.

The terminals or *lexical items* of the language are contained within the lexicon and, as mentioned above, may contain schemata as well (Kaplan and Bresnan, 1982). Each lexical definition contains the terminal symbol, its syntactic category, and any relevant schemata (Winograd, 1983). For example, table 1 contains a set of lexical items and their associated schemata appropriate for the sentence "Brutus stabbed a senator".

Terminal Symbol	Category	Schemata
a	DET	(↑ Definiteness) = Indefinite
Brutus	N	(↑ Number) = Singular (↑ Definiteness) = Definite (↑ Predicate) = 'Brutus'
senator	N	(↑ Number) = Singular (↑ Predicate) = 'Senator'
stabbed	V	(↑ Tense) = Past (↑ Predicate) = 'Stab⟨(↑ Subject), (↑ Object)⟩'

Table 1: Lexical entries relating to the sentence "Brutus stabbed a senator".

Once the tree structure has been created, the schemata associated with non-terminals are placed on the tree by their non-terminal symbols, while schemata associated with lexical items are placed on the tree at the location of their syntactic category (Kaplan and Bresnan, 1982). Figure 21 provides an example of the c-structure for the sentence "Brutus stabbed a senator" after the application of all appropriate schemata.

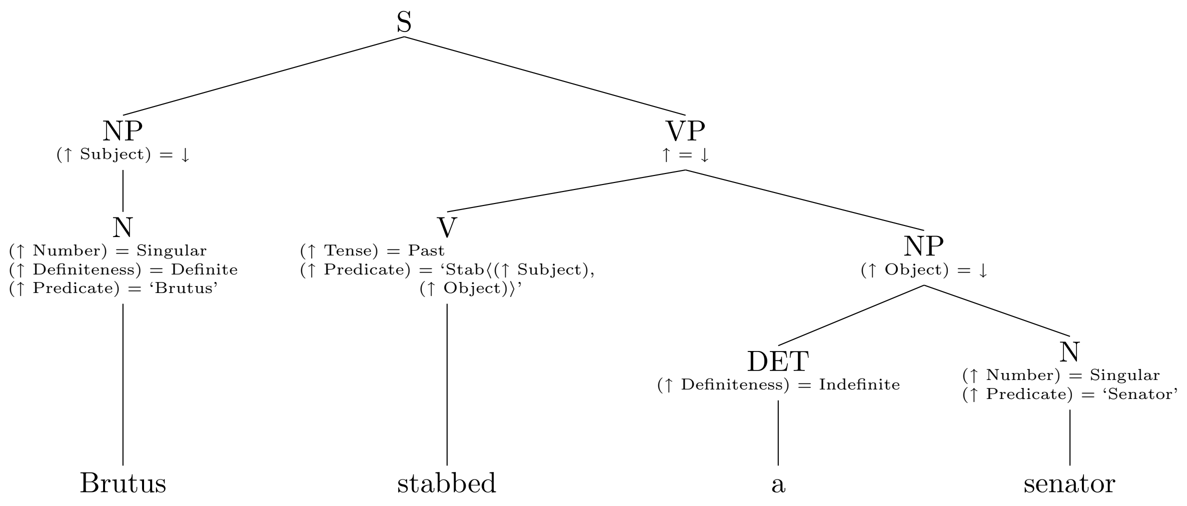


Figure 21: A c-structure representation of "Brutus stabbed a senator" with associated schemata.

Once the all the appropriate schemata have been applied, non-terminals are labelled with *variables* of the form x_n (Kaplan and Bresnan, 1982). Only non-terminals with a ↓ metavariable are labelled, as well as the S non-terminal. Figure 22 contains a c-structure labelled with variables.

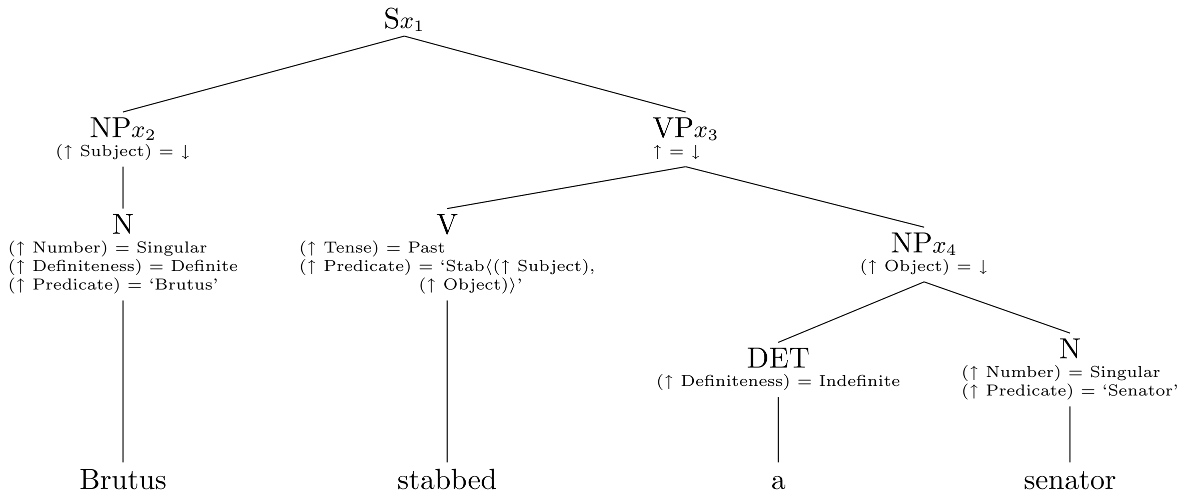


Figure 22: A c-structure representation of "Brutus stabbed a senator" after the application of variables.

The variables are used to generate functional descriptions or *f-descriptions* (Kaplan and Bresnan, 1982). Functional descriptions are simply a collection of properties that describe some portion of an f-structure. Figure 23 presents the set of functional descriptions that result from the c-structure in figure 22.

In order for the overall sentence to be considered grammatical, there must be at least one valid solution to the set of functional descriptions. In other words, the functional descriptions must simplify to a valid f-structure (Kaplan and Bresnan, 1982). While it

$$
\begin{aligned}
x_1 = x_3 = & \begin{bmatrix} \text{Subject} & = x_2 \\ \text{Object} & = x_4 \\ \text{Tense} & = \text{Past} \\ \text{Predicate} & = \text{`Stab}\langle(\text{Subject}),\ (\text{Object})\rangle\text{'} \end{bmatrix} \\
x_2 = & \begin{bmatrix} \text{Number} & = \text{Singular} \\ \text{Definiteness} & = \text{Definite} \\ \text{Predicate} & = \text{`Brutus'} \end{bmatrix} \\
x_4 = & \begin{bmatrix} \text{Definiteness} & = \text{Indefinite} \\ \text{Number} & = \text{Singular} \\ \text{Predicate} & = \text{`Senator'} \end{bmatrix}
\end{aligned}
$$

Figure 23: The set of functional descriptions for the sentence "Brutus stabbed a senator". The descriptions result from the c-structure seen in figure 22.

is possible that the c-structure is a valid production based upon the context-free rules, failure to find at least one satisfactory solution by unifying the set of functional descriptions indicates that the sentence is, in actuality, ungrammatical in nature[14]. Figure 24 presents the final f-structure that is obtained by unifying the set of functional descriptions in figure 23.

$$
\begin{bmatrix}
\text{Subject} & = & \begin{bmatrix} \text{Definiteness} & = & \text{Definite} \\ \text{Number} & = & \text{Singular} \\ \text{Predicate} & = & \text{`Brutus'} \end{bmatrix} \\
\text{Object} & = & \begin{bmatrix} \text{Definiteness} & = & \text{Indefinite} \\ \text{Number} & = & \text{Singular} \\ \text{Predicate} & = & \text{`Senator'} \end{bmatrix} \\
\text{Tense} & = & \text{Past} \\
\text{Predicate} & = & \text{`Stab}\langle(\text{Subject}),\ (\text{Object})\rangle\text{'}
\end{bmatrix}
$$

Figure 24: The final f-structure solution for the functional descriptions seen in figure 23.

Turning to the semantic content, contained within each f-structure is an attribute labelled 'Predicate' that yields semantic values for the sentence (Kaplan and Bresnan, 1982). The 'Predicate' attribute contains what is known as the *predicate argument structure*, which dictates how grammatical functions are mapped onto the "logical" structure or semantic meaning of the sentence (Bresnan, 1982). In the scope of our example in figure 24, the 'Predicate' attribute for the overall f-structure has the following configuration:

$$
\begin{array}{ccccc}
 & \text{Brutus} & & \text{Senator} & \\
 & \updownarrow & & \updownarrow & \\
\text{Stab}\langle & (\text{Subject}) & , & (\text{Object}) & \rangle
\end{array}
$$

The first argument place to the predicate argument structure for the verb 'Stab' is associated with the attribute 'Subject'. In our example case, we consult the 'Subject'

[14]Kaplan and Bresnan (1982) describe in detail the method by which f-descriptions may be unified. This algorithm is not presented here for reasons of brevity.

attribute in figure 24, and find that it is an f-structure. We consult the value of 'Predicate' in the f-structure, and find that 'Brutus' is the semantic value to use as the first argument to 'Stab'. Similarly, 'Senator' is the semantic value for 'Object', and becomes the second argument to 'Stab'. Overall, the sentence predicate becomes 'Stab⟨ Brutus, Senator ⟩'. In this way, LFG is able to provide a mapping of how syntactic constituents are involved in an action at the semantic level.

While the predicate argument structure appears to adequately describe semantic relationships, it is insufficient for use as a universal semantic formalism. As Bresnan (1982) states, all of the arguments contained within a predicate argument structure must be filled by grammatical functions. Arguments which are implicit or *internal* to the semantic function are lost. Bresnan's example is the sentence "Fred homered in the seventh". A reasonable assumption for the predicate argument structure would be:

$$\begin{array}{ccccc} & \text{Fred} & & & \\ & \updownarrow & & & \\ \text{Hit}\langle & \text{(Subject)} & , & \text{Home Run} & \rangle \end{array}$$

As Bresnan points out, the second argument place to 'Hit' contains the semantic value 'Home Run', which is understood to be a semantic constant. However, predicate argument structure slots may only be filled with grammatical functions. That is to say that the second argument place of 'Hit' may only be filled with some value obtained from a 'Predicate' attribute. Unfortunately, the c-structure and f-structure representations of the sentence do not yield a 'Predicate' value that corresponds to the 'Home Run' which appears in the second argument place in 'Hit'. It is for this reason alone that we are forced to conclude that the action of "homering" must be intransitive in nature. Thus, as Bresnan states, the only valid predicate argument structure representation for "Fred homered in the seventh" would be:

$$\begin{array}{ccc} & \text{Fred} & \\ & \updownarrow & \\ \text{Homer}\langle & \text{(Subject)} & \rangle \end{array}$$

Here, the semantic notion of a 'Home Run' is embedded in the knowledge of the function 'Homer'. In other words, the semantic information relating to the 'Home Run' became an internal semantic argument, and is no longer expressed in a transparent manner.

Unfortunately, this lack of transparency hides vital information that is essential to the proper semantic understanding of the function 'Homer'. In order to overcome this problem, Halvorsen (1983) suggests that additional layers of structure are required to represent the pure semantics of a sentence, resulting in an Intensional Logic description of the sentence[15]. Thus the pure semantic description of "Fred homered in the seventh" may have a description of:

$$\exists \text{x} \exists \text{y}(\text{x=FRED} \land \text{HOMERUN(y)} \land \text{HIT(x, y)})$$

Notice now the fact that Fred is represented by variable x, and that x hits an object y that is identified as a home run. It is only at the level of Intensional Logic that we now see the essence of what a 'Homer' actually entails. While Halvorsen (1983) provides a mechanism

[15]These layers consist of the semantic structure, Intensional Logic layer, and finally a model-theoretic interpretation (Halvorsen, 1983).

whereby an f-structure can be transformed into an model-theoretic construct expressed using Intensional logic, we observed in Thomas (2009) that Intensional logic ultimately has issues surrounding complexity and expressiveness. An example of a computational system that makes use of LFG is ITEX by Kranzdorf and Griefahn (1993).

2.6 Systemic Grammar

As stated by Patten (1988), Systemic Grammar was originally developed by Halliday (1961) as a way of describing the social and functional role of language in context (Winograd, 1983; Jurafsky and Martin, 2000). Like Conceptual Dependency, Systemic Grammar relies on the theory of stratification, in that language is broken up into numerous layers and mappings occur from one layer to the next (Winograd, 1983). However, unlike many formalisms that go to great lengths to specify constituency structure at the syntactic level, Systemic Grammar instead specifies the *function* of the elements contributing to an overall sentence, without specifying the exact syntactic form (Patten, 1988; Winograd, 1983).

Central to Systemic Grammar is the use of functional descriptions to describe the syntactic elements that occur in a sentence. There are three main sets of functions, referred to as *meta-functions* (Winograd, 1983; Patten, 1988; Jurafsky and Martin, 2000). The first meta-function called *mood* is known as the *interpersonal meta-function* as it describes the relationship between the reader and the writer and informs the reader of the overall sentence structure: command, question, statement or some combination of these three. The second meta-function called *transitivity* is known as the *ideational meta-function*, as it describes the contents of various expressions by informing the reader of various processes and their participants. The third meta-function called *theme* is known as the *textual meta-function*, as it describes what information is new to the reader, and how that information fits into the overall discourse. All of these meta-functions overlap to provide a full description of a particular sentence (Winograd, 1983). An example of the functional descriptions for the sentence “Brutus will stab Caesar” are represented in figure 25.

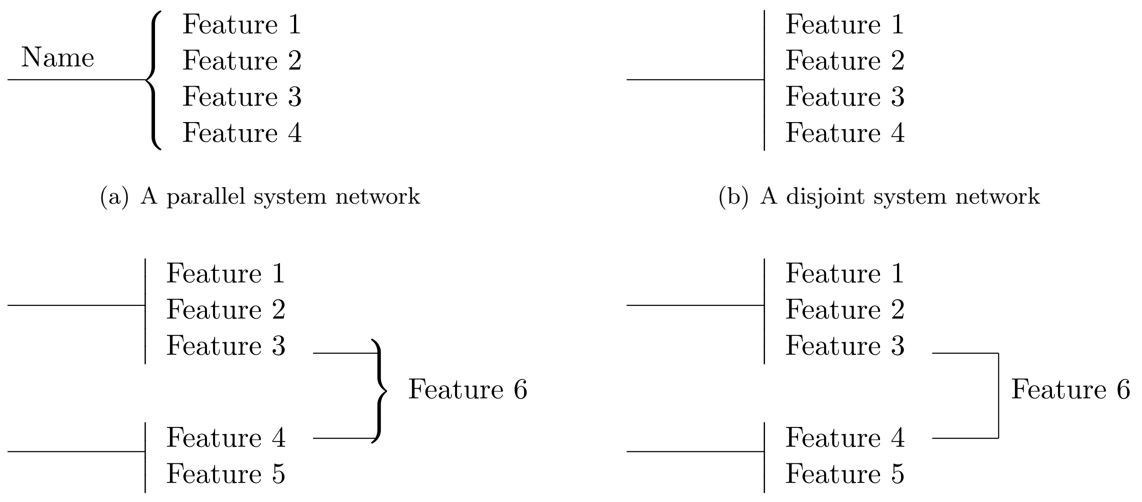
	Brutus	will	stab	Caesar
Mood	subject	finite	predicator	object
Transitivity	actor	process		goal
Theme	theme	rheme		

Figure 25: An example systemic analysis for the sentence “Brutus will stab Caesar”.

In order to generate sentences using a systemic grammar, a series of *system networks* are utilized[16] (Jurafsky and Martin, 2000). System networks reflect what are essentially a set of disjoint or parallel choices of features that are available when realizing a sentence (Patten, 1988). For example, consider the semantic choices Brutus would have when deciding how to carry out an assassination attempt. Brutus must semantically choose, amongst other things, the weapon (e.g. sword, knife, poison, etc.), time, place, and victim for the attempt. Figure 26 demonstrates the notation for various system networks. Figure 26(a) contains a simple system network named “Name” that contains a parallel choice of features, indicated by the curly brace. The parallel choice means that each of “Feature 1” through “Feature 4” must be selected. In contrast, the system network in figure 26(b) contains a disjoint

[16]Winograd (1983) uses the term *choice system*, which means the same as *system network*.

set of choices, indicated by the vertical bar. In this case, only one of “Feature 1” through “Feature 4” may be selected. In figure 26(c), the curly brace is used to indicate a parallel entry condition to the sub-system. Simply put, the parallel entry condition means that *both* “Feature 3” and “Feature 4” must be chosen before “Feature 6” may be selected. In contrast, figure 26(d) demonstrates the form of the disjoint entry condition, where either one, or both of “Feature 3” and “Feature 4” may be chosen before “Feature 6” may be selected.



(a) A parallel system network

(b) A disjoint system network

(c) A parallel entry condition

(d) A disjoint entry condition

Figure 26: Examples of simple system networks.

System networks have been used extensively at the syntactic level to demonstrate what conditions are necessary to form various syntactic constructs (Winograd, 1983). For example, figure 27 provides a simplified version of a system network that contains the choices involved in the selection of English personal pronouns. When selecting an appropriate personal pronoun to use, a human must choose between the disjuncts Question, Personal or Demonstrative. If the human were to select the feature Personal, then they must choose in parallel features for the sub-system networks Case, Number and Person. As Winograd (1983) states, the pronoun “I” is obtained by selecting the features: Personal, Case: Subjective, Person: First, and Number: Singular. The pronoun “she” is obtained by selecting the features: Personal, Case: Subjective, Person: Third, and Number: Singular. Since both Number: Singular and Person: Third have been selected, then the Gender must also be chosen, which in the case of the pronoun “she” would be Feminine.

System networks alone are not sufficient to generate a true syntactic form. The generation of an utterance relies on the use of *realization rules* in combination with system networks (Winograd, 1983; Patten, 1988; Jurafsky and Martin, 2000). Each realization rule acts as a constraint on how the final functional form of the sentence may be realized. The realization rules are attached to the system network at the location of features where they become valid. For example, figure 28 contains a portion of the system network that describes mood choices for the English clause. Realization rules are displayed in italics underneath the feature where they apply.

There are a number of realization rules that are involved in the generation of a sentence.

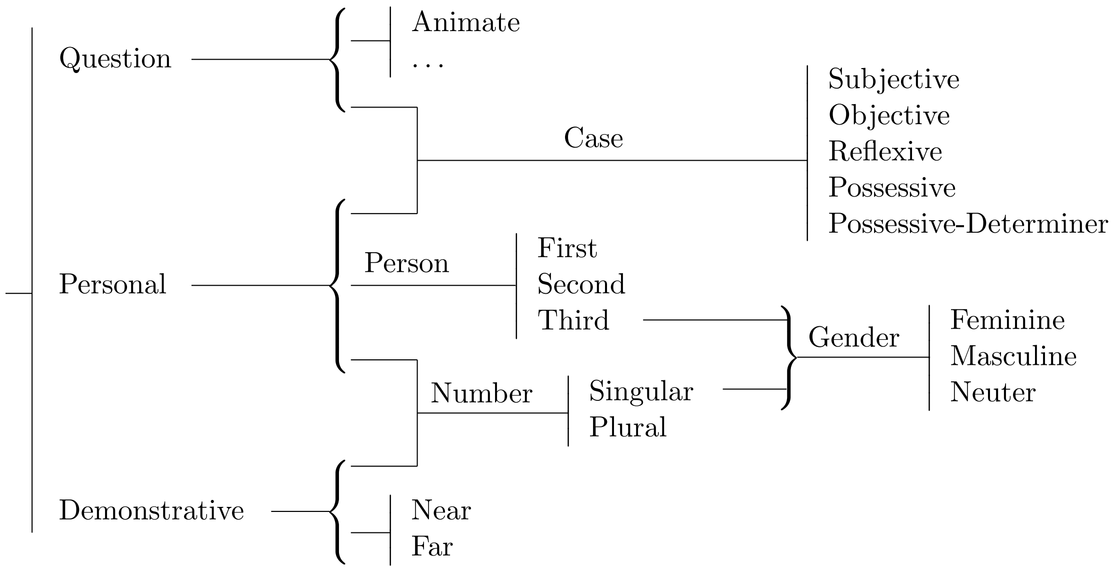


Figure 27: A system network relating to the choice of English pronouns, reproduced from Winograd (1983).

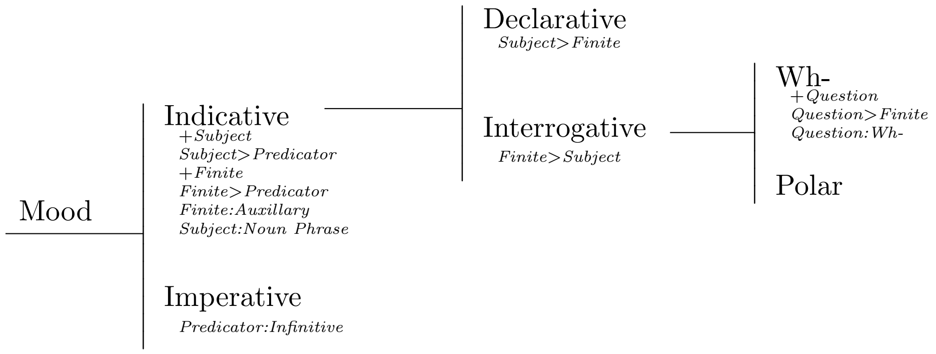


Figure 28: A system network relating to the choices involved with mood for the English clause, reproduced from Jurafsky and Martin (2000).

Unfortunately, the exact syntax and actual number of rules varies from source to source (Patten, 1988). The following summarizes some of the realization rules described in Jurafsky and Martin (2000), Winograd (1983) and Patten (1988). The insertion rule $+X$ states that function X should be inserted into the functional description. For example, +*Subject* states that the function *Subject* should inserted. The conflation rule X/Y states that functions X and Y may occur on the same constituent. For example, "Brutus" may be described by the mood function *Subject*, the transitive function *Actor* and the theme function *Theme*. Thus, *Subject/Actor* states that the constituent may be described by the mood function *Subject* and the transitive function *Actor*. The ordering rule $X>Y$ states that the function X should come before function Y. For example, *Subject>Predicator* states that the function *Subject* should come before the function *Predicator*. The adjacency rule $X\hat{}Y$ states that function X is adjacent to function Y. For example, *Subjectˆ Finite* states that functions *Subject* and *Finite* must be beside each other. Variants on the adjacency rule are $\#\hat{}Y$, which states that Y is the first function in the entire description, and $\%\hat{}Y$, which states that Y is the first function in some expanded function. The preselection rule *X:Y*, states that the function X should take on the form of Y. This particular rule allows the system network to classify a function based upon a feature that exists on a different stratum (Patten, 1988). For example, the preselection *Subject:Noun Phrase* states that the function *Subject* is to be a form of *Noun Phrase*, a function which exists at the syntactic level. The preselection rule allows functions on various levels to interact with one another. Finally, the lexify rule $X=Y$ states that the function X should have the lexical form Y. For example *Subject=Brutus* means that the value of the function *Subject* should be *Brutus*. In this way, Y specifies the exact syntactic form to use.

To demonstrate system networks and realization rules together, consider the system network that was presented in figure 28. The progression represented in figure 29 demonstrates how the functional description of a sentence would change after the realization rules are applied from the selection of each feature. The example ultimately results in the mood description of a simple interrogative sentence.

Mood	Subject	Finite	Predicator

(a) After selection of *Indicative* feature

Mood	Finite	Subject	Predicator

(b) After selection of *Interrogative* feature

Mood	Question	Finite	Subject	Predicator

(c) After selection of *Wh-* feature

Figure 29: The generation of a functional sentence structure following selection of features along the Mood system network from figure 28.

As stated previously, systemic grammar relies on the notion of stratification. That is to say that elements of the language can be broken up into different layers or strata: the semantic stratum, the syntactic stratum (which includes both the lexicon and grammar) and the phonological stratum (Winograd, 1983). So far, our examples have concentrated on the syntactic stratum. It is possible however, to use system networks to represent the choices that a human would have on the level of semantics. These available choices rely on the exact context, which is referred to as the "register" (Halliday, 1978; Patten, 1988).

For example, consider the system network that would be available to Brutus when deciding how to murder Caesar. Figure 30 provides a simplified system network corresponding to the choices that Brutus must make.

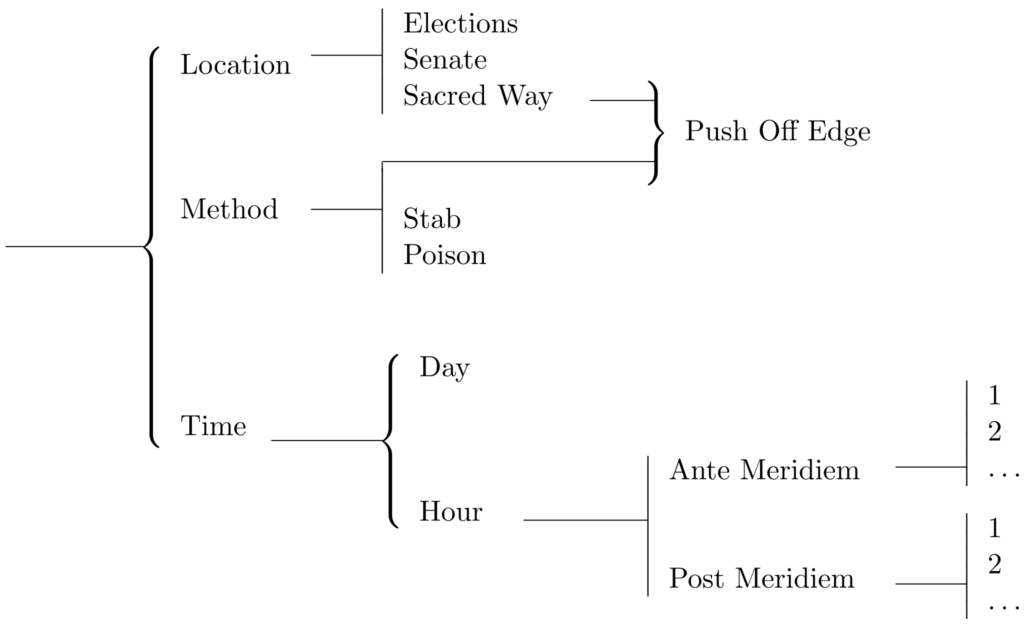


Figure 30: A system network relating to the semantic choices that Brutus would have to make when deciding how to assassinate Caesar.

One may imagine that the semantic system network would have realization rules attached to each feature, and those realization rules would interact with the syntactic stratum to provide constraints on how semantic information may be realized. In this way, semantic choices would have a direct impact on the syntactic formation of an utterance. However, questions of scalability and practicality come to mind with such semantic system networks, especially when one considers the number of semantic choices that exist for any given context. As Patten (1988) points out, Halliday (1978) is aware that further study of the semantic strata is needed, and that the actual set of structures needed to represent meaning may be drastically different than system networks. Researchers such as Mann and Matthiessen (1985) have ventured into this area, and have extended systemic grammar by adding an inquiry mechanism and a chooser mechanism. The inquiry mechanism allows the grammar to ask questions about the external environment in a controlled fashion, while the chooser then decides which system to follow based upon the answers. While this approach is useful, there are some limitations surrounding the applicability of the chooser-inquiry mechanism to tasks such as parsing (O'Donnell, 1994).

To date, there are many computational implementations which make use of systemic grammar, both for syntactic and semantic purposes. Some of these systems are: NIGEL (Mann and Matthiessen, 1985), SLANG (Patten, 1988), GENESYS (Fawcett, 1990), HORACE (Cross, 1992), IMAGENE (Linden *et al.*, 1992), WAG (O'Donnell, 1995), PRETEXTE (Gagnon and Lapalme, 1996), SURGE (Elhadad and Robin, 1996), KPML (Bateman, 1997) and ILEX (O'Donnell *et al.*, 2001).

2.7 The Generative Lexicon

The Generative Lexicon is a semantic theory developed by Pustejovsky (1991) in response to the research by Levin and Rappaport (1986) that criticized the granularity of the θ-role in Case Grammar, as well as to what Pustejovsky saw as descriptive inadequacies in other semantic theories. In the Generative Lexicon, Pustejovsky (1995) is concerned with *lexical semantics* - the meaning of individual words - and how current semantic theories are unable to provide an adequate level of description necessary to create a well defined semantic lexicon. Pustejovsky states that the semantics of a sentence is intricately tied to the syntactic form of a sentence, and that the combination of various lexical choices with syntactic patterns results in new meaning not captured directly by primitives in the semantic lexicon. Thus, a new way of looking at how to encode lexical meaning is needed.

Pustejovsky (1995) uses various examples to demonstrate the problems that occur with traditional *sense enumeration lexicons.* For example, he shows that there must be at least two lexical entries necessary to account for the different uses of "run" in the sentences "Mary ran to the store yesterday" and "Mary ran yesterday". In the first instance, "run" is used to indicate that Mary got somewhere in particular by running, while in the second sense, it is used to indicate that Mary was simply moving by a particular means. While these two meanings of the verb are related, current semantic theories offer no choice but to create separate entries in the lexicon to capture the meaning of the different verb senses. Unfortunately, this particular solution results in a large number of entries in the lexicon, and does not provide information as to whether or not one element is related to another. This problem becomes more pronounced with the adjective "good". As Pustejovsky points out, the definition of "good" may change for each item to which it is applied. For example, "good children", "good food", and "good weather" each invoke a different sense definition for "good". Given the large number of unique uses for "good", it becomes difficult to ensure that the lexicon has an entry for every sense possible. This particular shortcoming has already been demonstrated with semantic networks, where it is apparent that creating a definition for every single use of a word will never be satisfactory since different and new contexts may change the semantic meaning of any given word (Busa *et al.*, 2001).

As noted above, the Generative Lexicon attempts to overcome the problems that occur in various semantic systems. In particular, Pustejovsky (1995) states that traditional exhaustive approaches should instead be transformed to work in a *generative* fashion. Simply put, rather than exhaustively enumerate every possible sense of a word and list their primitive compositions, Pustejovsky instead assumes that there are a set number of devices that are responsible for generating semantic meaning in context. Using this approach, the semantics of related senses of a word can be captured by a single lexical entry.

Pustejovsky (1995) states that every lexical item has a *qualia structure* that plays an important role when determining the semantics of a sentence. Traditional semantic analyses are focused around the verb, and thus the lexicon usually contains a multitude of verbs that each reflect a different sense of the word. With qualia structures, the verb instead has a minimal structure that combines with the qualia structure of nouns and noun phrases to create new meaning. This is best seen with the examples that Pustejovsky provides: "John baked a potato" and "John baked a cake". In the former, the use of the verb "bake" reflects a change of state, while the latter reflects a creation event. As Pustejovsky points out, the shift of meaning of "bake" in the two sentences does not come from two different lexical entries for bake, rather it comes from the two different combinations of qualia structure. In the first case, "bake" will combine with information in "potato" which says that potatoes are naturally occurring objects that may be “heated up”. In the second case, “bake” will combine with information in “cake” which says that cakes are derived artifacts that can only be created through some creation process. In this way, the noun is responsible for determining how “bake” is interpreted.

To achieve this level of representation, Pustejovsky (1995) describes the qualia structure in great detail. In simple terms, the qualia structure is responsible for identifying the roles that a word may play. In a qualia structure, there are four different *qualia* that may occur: constitutive, formal, telic and agentive. The contribution of these qualia to the overall semantic meaning of a word carry both a semantic and grammatical impact. For example, the *formal* quale describes the physical characteristics of the item, and usually has a *type* associated with a formal argument. The *telic* quale provides information relating to the purpose of the entity. For example, Pustejovsky provides the description of “knife” in figure 31.

$$\begin{bmatrix} \textbf{knife} & & \\ \text{ARGSTR} & = & \begin{bmatrix} \text{ARG1} & = & \text{x:tool} \end{bmatrix} \\ \text{QUALIA} & = & \begin{bmatrix} \text{FORMAL} & = & \text{x} \\ \text{TELIC} & = & \text{cut(e, x, y)} \end{bmatrix} \end{bmatrix}$$

Figure 31: The qualia structure for the noun “knife”, reproduced from Pustejovsky (1995).

In figure 31, the ARGSTR represents the formal arguments of the lexical item, and is discussed in more detail below[17]. With this example, the lexical item “knife” has a single argument that belongs to the type “tool”. The FORMAL quale simply identifies an individual item from the set of tools. The TELIC quale provides information relating to the purpose of the entity, in this case, the purpose of the knife is to cut something. In the function ‘cut’, the variable ‘e’ represents the type of event, and ‘y’ represents the object that undergoes cutting. Not yet mentioned is the *constitutive* quale, which provides the material composition of the entity. For example, Pustejovsky (1995) states that “hand” would be a part of a body. Finally, the *agentive* quale identifies how an item is brought about. An example from Pustejovsky is that of a “book”, which would have to be written. While these examples explore nouns, the same qualia may be applied to verbs. For example, consider the qualia structure for “kill”, which is given in figure 32.

$$\begin{bmatrix} \textbf{kill} & & \\ \text{QUALIA} & = & \begin{bmatrix} \text{FORMAL} & = & \text{dead}(e_2, \boxed{2}) \\ \text{AGENTIVE} & = & \text{kill\_act}(e_1, \boxed{1}, \boxed{2}) \end{bmatrix} \end{bmatrix}$$

Figure 32: The qualia structure for the verb “kill”, reproduced from Pustejovsky (1995).

Within the qualia structure presented in figure 32, only the formal and agentive qualia are provided. This structure reflects two senses of the verb kill. The first verb sense reflects the state of being dead, as is indicated in the FORMAL quale. Here, there are two arguments to the dead function. The argument e_2 is inherited from the event structure, where it provides information that the semantic function in question reflects a state of being. The argument $\boxed{2}$ is inherited from the argument structure, which states that the argument is a physical entity. Read together, dead(e_2, $\boxed{2}$) states that the entity $\boxed{2}$ is in the state of

[17]Pustejovsky (1995) notes that some lexical items may lack one or more qualia.

being dead. The second verb sense reflects how the state of killing is brought about, and is indicated by the AGENTIVE quale. The argument e_1 states that the function 'kill_act' is a process, while $\boxed{1}$ denotes the entity performing the action, and $\boxed{2}$ denotes the entity undergoing the action. The objects e_1, e_2, $\boxed{1}$ and $\boxed{2}$ are all imported from other levels of semantic structure, where they are more formally defined.

As mentioned above, qualia structure does not act alone. Instead, Pustejovsky (1995) states that qualia structure is one of four levels that describe the semantics of a given utterance. Each level contains its own structure and provides different sorts of information to the overall semantic meaning. The four levels are: argument structure, event structure, qualia structure and inheritance structure. The *argument structure* reflects the valency requirements of the word, and describes the nature of the arguments in more detail. An example of the argument structure and qualia structure for "kill" is given in figure 33.

$$\left[\begin{array}{lll} \textbf{kill} & & \\ \text{ARGSTR} & = & \left[\begin{array}{lll} \text{ARG1} & = & \boxed{1}\left[\begin{array}{l}\textbf{ind}\\ \text{FORMAL} = \text{phys\_obj}\end{array}\right] \\ \text{ARG2} & = & \boxed{2}\left[\begin{array}{l}\textbf{animate\_ind}\\ \text{FORMAL} = \text{phys\_obj}\end{array}\right]\end{array}\right] \\ \text{QUALIASTR} & = & \left[\begin{array}{lll}\text{FORMAL} & = & \text{dead}(<\text{state}>, \boxed{2}) \\ \text{AGENTIVE} & = & \text{kill\_act}(<\text{process}>, \boxed{1}, \boxed{2})\end{array}\right]\end{array}\right]$$

Figure 33: The argument and qualia structure for the verb "kill", reproduced from Pustejovsky (1995).

As can be seen from the example, "kill" takes two arguments: argument $\boxed{1}$, which must be a physical object, and argument $\boxed{2}$, which must also be a physical object. These argument specifications act as constraints on the types of objects that can be passed to the various functions[18]. For example, in the 'dead' function in the qualia structure, argument $\boxed{2}$ must be a physical object. Therefore, only lexical items which describe physical objects may be combined with "kill" to form valid phrases.

The event structure identifies the word as a state, process, or transition, and also serves to provide information about how the event is internally structured (Pustejovsky, 1995). For example, the verb "kill" starts with a process and results in a state of being. In order to capture this information, Pustejovsky defines a notation for event ordering, and provides an attribute within the event structure to express this information. The lexical entry for "kill" with the event, argument and qualia structures is provided in figure 34. With this example, the RESTR attribute within the event structure provides information as to how the event is structured. The symbol $<_{\propto}$ states that the overall event starts at e_1 and ends at e_2. Thus, the lexical item kill is composed of a 'kill_act' followed by a 'dead' state. The notation developed by Pustejovsky (1995) allows for many types of event structures. For example, the $\circ_{\propto}$ symbol states that events would occur in parallel. This is useful for verbs such as "accompany".

To see how multiple meanings can be achieved in context, consider the statement "Brutus

[18]The two arguments seen so far are examples of *true arguments*, since both arguments must be expressed in the syntax in order to understand the meaning of the sentence (Pustejovsky, 1995). Pustejovsky defines other types of arguments such as default arguments, shadow arguments and true adjuncts, which allow for information to be optionally omitted from the surface structure by specifying explicitly in the argument structure what they are.

$$\begin{bmatrix} \textbf{kill} & & \\ \text{EVENTSTR} & = & \begin{bmatrix} \text{E1} & = & \text{e}_1\text{:state} \\ \text{E2} & = & \text{e}_2\text{:process} \\ \text{RESTR} & = & <_\propto \end{bmatrix} \\ \text{ARGSTR} & = & \begin{bmatrix} \text{ARG1} & = & \boxed{1} \begin{bmatrix} \textbf{ind} \\ \text{FORMAL} = \text{phys\_obj} \end{bmatrix} \\ \text{ARG2} & = & \boxed{2} \begin{bmatrix} \textbf{animate\_ind} \\ \text{FORMAL} = \text{phys\_obj} \end{bmatrix} \end{bmatrix} \\ \text{QUALIASTR} & = & \begin{bmatrix} \text{FORMAL} & = & \text{dead}(\text{e}_1, \boxed{2}) \\ \text{AGENTIVE} & = & \text{kill\_act}(\text{e}_2, \boxed{1}, \boxed{2}) \end{bmatrix} \end{bmatrix}$$

Figure 34: The event and qualia structures for the verb “kill”, as produced by Pustejovsky (1995).

used the knife”. In figure 31, the TELIC attribute within the qualia structure provided the purpose of the “knife”. The definition of the verb “use” would consult the TELIC attribute of the argument “knife”, and find that its purpose is “to cut”. Thus “Brutus used the knife” would be explicitly identified as involving a cutting action. By using the TELIC attribute for other nouns, “use” can be combined successfully with “gun” or “brush” to provide the correct senses of “use” that each of these nouns suggest.

There have been several criticisms of the Generative Lexicon. According to Fodor and Lepore (1998), the notion of the semantic lexicon as imagined by Pustejovsky (1995) is an unattainable goal. One argument stemming from Fodor and Lepore is that Pustejovsky’s framework cannot explain how an item may be put to a use that does not reflect its TELIC role. For example, consider the fact that someone may use a screwdriver to pound a nail into a wall. Fodor and Lepore summarize the heart of this issue with the question: “what happens if a verb makes a demand on an argument that the lexical entry of the argument doesn’t satisfy?” Other criticisms focus on the ontological characteristics of the lexicon itself. As posited by Fodor and Lepore, why should the lexical entries for the two senses of the word “bake” (see above) be combined into a single entry at all? More importantly, what is the methodology used to determine their relatedness? While surely they share some properties, one is a creation process and the other reflects a change of state. Fodor and Lepore conclude that the problem of distinguishing word sense is instead recast as a problem regarding the enumeration of all possible processes that occur in natural language.

2.8 Rhetorical Structure Theory

Rhetorical Structure Theory (RST) developed by Mann (1984) attempts to describe how a text is organized, and works by stating the nature of the relationships that occur over portions of non-overlapping text. RST is meant to comprehensively describe full texts, as opposed to the single sentence descriptions that many other formalisms are limited to. Mann identified several factors which contributed to the development of RST. In essence, he required a compressive theory that could scale easily, was formal in nature, provided informative descriptions, could be used in a generative setting, and was not limited to a single type of text. The result was the development of RST. While Mann stated that its initial purpose was to provide a descriptive mechanism for text, one of his goals was to augment the theory and use it to construct text.

Within the framework of RST, Mann and Thompson (1988) state that there are a series of relationships that are involved in various structural configurations, which are used to describe the text. The *relations* possible in RST are between two portions of text, and consist of the originating sentence termed the *nucleus*, and the related sentence known as the *satellite*. Constraints may be placed on either or both parts of the relation. Each relation also has an *effect* that describes the overall impact of the relationship. For example, consider the sentences "$_1$[Caesar must be assassinated.] $_2$[He is a dictator who is drawn to power,] $_3$[and is unwilling to listen to other members of the senate.]" Portion 1 is the nucleus, while portions 2 and 3 form the satellite. The relationship between the sentences is known as *justification*, the overall effect is that portions 2 and 3 (the satellite) increase the listener's willingness to accept the argument put forward in portion 1 (the nucleus). Mann and Thompson (1988) outline 23 different relations that may occur in RST; however, they explicitly state that the set of possible relations remains open since different genres and cultural styles are constantly extending the framework.

The relationships are depicted using a *schema*, which is responsible for describing the constituency structure of the text (Mann, 1984; Mann and Thompson, 1988). For example, figure 35 contains the schema for the example sentences above. The curve represents the relationship that holds between the various portions of the text, and points from the satellite to the nucleus. The horizontal lines denote the portions of the text that are affected, and are marked with the passage numbers to indicate their scope. In all, there are five different schemas that are possible within RST, and several types of schemas are used to relate other patterns of organization. A *schema application* simply refers to the schemas and relations as they appear in context.

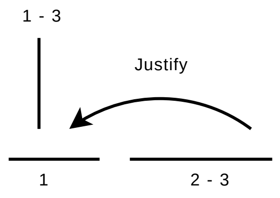


Figure 35: An example of the justify relation in a schema.

RST *structures* are simply portions of text that are marked as "units" (Mann and Thompson, 1988). The exact definition of a unit is rather informally stated by Mann (1984), but is generally considered to be a clause. Regardless of the character of the unit, the consequent structure of the schema is required to obey several constraints. The schema structure should be complete in the sense that the schema applications should cover the entire text. The resulting schema structure should be fully connected so that no portions of the text fall outside of the hierarchy of schema applications. The connected schemas should be unique (no overlaps). This results in a tree structure.

Bringing all portions of RST theory together, the diagram in figure 36 represents the structure of the preceding paragraph. Each sentence represents one unit in the RST tree. The units put together form the RST analysis of the tree. Notice that all of the schema applications cover the entire text, and that the tree is fully connected.

The nature of RST makes it possible to perform analysis over structured texts. However, the theory does not lend itself to examine any given sentence in great depth. It is left up to other linguistic theories to provide more information. While this particular formalism provides insights into how various texts are structured, on its own, RST is not capable of

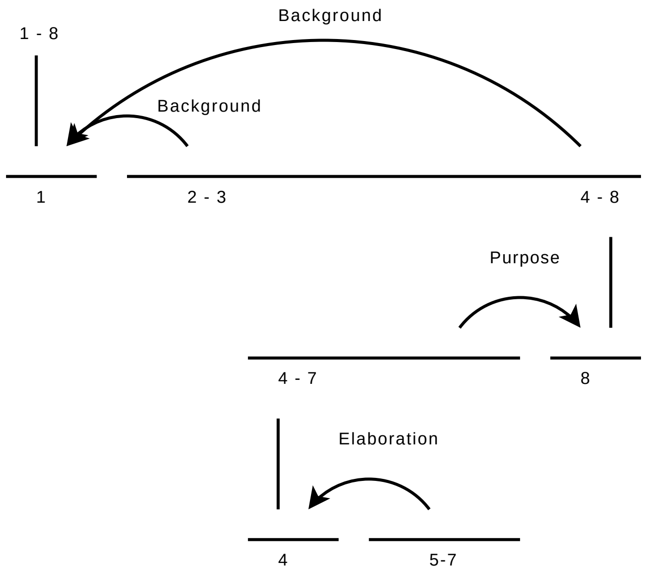


Figure 36: An example of RST of the preceding paragraph.

performing the fine-grained analysis that is needed in a universal semantic tool.

Some computational systems that make use of RST include: PEA (Moore and Swartout, 1989), IMAGENE (Linden *et al.*, 1992), PPP (Gulla, 1996), OPADE (de Rosis *et al.*, 1999), DArt$_{bio}$ (Bateman *et al.*, 2001) and ILEX (O'Donnell *et al.*, 2001).

3 Summary

All of the semantic systems briefly explained in this paper are of interest to computational linguists seeking to represent knowledge in a formal manner. Case grammar by Fillmore (1968) was the first example of a system that had a primary goal of describing the semantic contribution of various grammatical objects. This particular formalism became influential in the development of many systems, despite its limited granularity and descriptive power. Conceptual Dependency by Schank (1972) was developed using a semantic network as a primary mode of expression. Using simple primitives, this powerful formalism dealt directly with concepts, but ultimately failed to capture the greater meaning behind instrumental cases, and could not explain key concepts such as social institutions and inferences. Turning to the greater scope of semantic networks, many of the network formalisms developed offered different ways of expressing semantic information. Powerful type hierarchies and propositional networks make it possible to express a wide variety of phenomena, and make them suitable for inferencing tasks. Unfortunately, their inability to deal with extensional information make them problematic at times (Johnson-Laird *et al.*, 1984).

Discourse Representation Theory by Kamp (1981) was the first formalism dedicated to the expression of multi-sentence discourse and anaphoric relationships. While the formalism was limited to being a notational variant of the first order predicate calculus, it offered a new and systematic way of building up knowledge representations across a large body of text. Systemic grammar by Halliday (1994) offers a break from the traditional constituency based approach of other grammars, through the use of powerful system networks and multiple layers of representation. However, the scalability of these networks in light of the greater set of semantic possibilities remains to be seen. Lexical Functional Grammar by Kaplan and Bresnan (1982) offers a way to relate grammatical functions and semantic themes using by utilizing a portion of unification grammar. Unfortunately, the underlying semantic structure in LFG utilizes Intensional Logic, which ultimately has issues with user-friendliness.

Finally, the notion of the Generative Lexicon by Pustejovsky (1995) offered a new way of approaching semantics. Instead of decomposing lexical entries into a set of primitives, or otherwise enumerating every possible sense of a word, the theory of the Generative Lexicon offered a way to build semantic meaning in context. Unfortunately, the underlying basis of the Generative Lexicon has been questioned, and it remains to be seen whether this new approach towards building a semantic lexicon will continue to be embraced. Rhetorical Structure Theory by Mann (1984) provides useful tools for analyzing large spans of text, but is not meant to be a complete replacement for other forms of semantic specification. While useful in combination with other forms of representation, it alone does not have the level of detail necessary to capture the pure semantics of a natural language.

The limitations of these system may not be final, as active research and extension of many of them continues. However, it must be noted that none of the systems explored are currently capable of the type of scalability and practicality that would make them a truly user-friendly and universal semantic tool.

References

Ali, S.S. and Shapiro, S.C. (1993). Natural language processing using a propositional semantic network with structured variables. *Minds and Machines*, 3(4):421–451.

Anderson, J.R. and Kline, P. (1977). Design of a production system for cognitive modelling. In *Proceedings of the Workshop on Pattern-Directed Inference Systems*, pages 60–65.

Bateman, J., Kleinz, J., Kamps, T., and Reichenberger, K. (2001). Towards constructive text, diagram, and layout generation for information presentation. *Computational Linguistics*, 27(3):409–449.

Bateman, J. and Zock, M. (August 2008). John Bateman and Michael Zock's list of natural language generation systems. `http://www.fb10.uni-bremen.de/anglistik/langpro/NLG-table/NLG-table-root.htm`.

Bateman, J.A. (1997). Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3(01):15–55.

Brachman, R.J. (1985). I lied about the trees, or, defaults and definitions in knowledge representation systems. *AI Magazine*, 6(3):80–95.

Brachman, R.J. and Schmolze, J.G. (1985). An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216.

Bresnan, J. (1982). Polyadicity. In J. Bresnan (Editor), *The Mental Representation of Grammatical Relations*, pages 149–172. The MIT Press.

Busa, F., Calzolari, N., Lenci, A., and Pustejovsky, J. (2001). Building a semantic lexicon: Structuring and generating concepts. In H. Bunt, R. Muskens, and E. Thijsse (Editors), *Computing Meaning*, volume 77 of *Studies in Linguistics and Philosophy*, pages 29–51. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Cline, B.E. and Nutter, J.T. (1994). Kalos—a system for natural language generation with revision. In *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, pages 767–772. American Association for Artificial Intelligence, Menlo Park, CA.

Clippinger, J.H. (1975). Speaking with many tongues: some problems in modeling speakers of actual discourse. In *TINLAP '75: Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, pages 68–73. Association for Computational Linguistics, Morristown, NJ.

Cross, M. (1992). Choice in lexis: Computer generation of lexis as most delicate grammar. *Language Sciences*, 14(4):579 – 605.

Daelemans, W., Smedt, K.D., and Gazdar, G. (1992). Inheritance in natural language processing. *Computational Linguistics*, 18(2):205–218.

Davis, S. and Gillon, B.S. (2004). Introduction. In S. Davis and B.S. Gillon (Editors), *Semantics: A Reader*, pages 1–111. Oxford University Press, Oxford, UK.

de Rosis, F., Grasso, F., and Berry, D.C. (1999). Refining instructional text generation after evaluation. *Artificial Intelligence in Medicine*, 17(1):1–36.

Dunlop, C.E. (1990). Conceptual dependency as the language of thought. *Synthese*, 82(2):274–296.

Elhadad, M. and Robin, J. (1996). An overview of SURGE: a reusable comprehensive syntactic realization component. Technical Report 96-03, Department of Mathematics and Computer Science, Ben Gurion University, Beer Sheva, Israel.

Endriss, C. and Klabunde, R. (2000). Planning word-order dependent focus assignments. In *INLG '00: Proceedings of the first international conference on Natural language generation*, pages 156–162. Association for Computational Linguistics, Morristown, NJ.

Fawcett, R.P. (1990). The computer generation of speech with discoursally and semantically motivated intonation. In *Proceedings of the Fifth International Workshop on Natural Language Generation*, pages 164–173a. Dawson, PA.

Fillmore, C.J. (1968). The case for case. In E. Bach and R.T. Harms (Editors), *Universals in Linguistic Theory*, pages 1–90. Holt, Rinehart and Winston, New York, NY.

Fodor, J.A. and Lepore, E. (1998). The emptiness of the lexicon: Reflections on James Pustejovsky's "The Generative Lexicon". *Linguistic Inquiry*, 29(2):269–288.

Gagnon, M. and Lapalme, G. (1996). From conceptual time to linguistic time. *Computational Linguistics*, 22(1):91–127.

Geurts, B. and Beaver, D.I. (Winter 2008). Discourse representation theory. In E.N. Zalta (Editor), *The Stanford Encyclopedia of Philosophy*. `http://plato.stanford.edu/archives/win2008/entries/discourse-representation-theory/`.

Goldman, N.M. (1975). Sentence paraphrasing from a conceptual base. *Communications of the ACM*, 18(2):96–106.

Gulla, J.A. (1996). A general explanation component for conceptual modeling in CASE environments. *ACM Transactions on Information Systems*, 14(3):297–329.

Halliday, M. (1961). Categories of the theory of grammar. *Word*, 17(3):241–292.

Halliday, M. (1978). *Language as social semiotic.* Edward Arnold (Publishers) Ltd, London, UK.

Halliday, M. (1994). *Introduction to Functional Grammar.* Edward Arnold, New York, NY, second edition.

Halvorsen, P.K. (1983). Semantics for lexical-functional grammar. *Linguistic Inquiry*, 14(4):567–615.

Jackendoff, R. (1983). *Semantics and Cognition.* Current Studies in Linguistics Series. The MIT Press, Cambridge, MA.

Jackendoff, R. (1990). *Semantic Structures.* Current Studies in Linguistics. The MIT Press, Cambridge, MA.

Johnson-Laird, P.N., Herrmann, D.J., and Chaffin, R. (1984). Only connections: A critique of semantic networks. *Psychological Bulletin*, 96(2):292–315.

Jurafsky, D. and Martin, J.H. (2000). *Speech and Language Processing.* Prentice Hall Series in Artificial Intelligence. Prentice Hall, New Jersey.

Kamp, H. (1981). A theory of truth and semantic representation. In J. Groenendijk, T. Janssen, and M. Stokhof (Editors), *Formal Methods in the Study of Language*, volume 135 of *Mathematical Centre Tracts*, pages 277–322. Mathematisch Centrum.

Kamp, H. (1988). Discourse representation theory: What it is and where it ought to go. In *Natural Language at the Computer*, volume 320 of *Lecture Notes in Computer Science*, pages 84–111. Springer Berlin.

Kamp, H. and Reyle, U. (1993). *From Discourse to Logic: Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and Discourse Representation Theory*, volume 42 of *Studies in Linguistics and Philosophy.* Kluwer Academic Publishers, Dordrecht, The Netherlands.

Kaplan, R.M. and Bresnan, J. (1982). Lexical-functional grammar: A formal system for grammatical representation. In J. Bresnan (Editor), *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press.

Klarner, M. (2004). Hybrid NLG in a generic dialog system. In A. Belz, R. Evans, and P. Piwek (Editors), *Natural Language Generation*, volume 3123 of *Lecture Notes in Artificial Intelligence*, pages 205–211. Springer Berlin.

Kohl, D., Plainfossé, A., and Gardent, C. (1990). The general architecture of generation in ACORD. In *COLING*, pages 388–390.

Kranzdorf, H. and Griefahn, U. (1993). Text planning in ITEX: A hybrid approach. In H.J. Ohlbach (Editor), *GWAI-92: Advances in Artificial Intelligence*, volume 671 of *Lecture Notes in Computer Science*, pages 235–246. Springer Berlin.

Lehmann, F. (1992). Semantic networks. *Computers & Mathematics with Applications*, 23(2-5):1–50.

Levin, B. and Rappaport, M. (1986). The formation of adjectival passives. *Linguistic Inquiry*, 17(4):623–661.

Levison, M. and Lessard, G. (1992). A system for natural language sentence generation. *Computers and the Humanities*, 26(1):43–58.

Linden, K.V., Cumming, S., and Martin, J. (1992). Using system networks to build rhetorical structures. In R. Dale, E. Hovy, D. Rosner, and O. Stock (Editors), *Aspects of Automated Natural Language Generation*, volume 587 of *Lecture Notes in Computer Science*, pages 183–198. Springer Berlin.

Mann, W.C. (1984). Discourse structures for text generation. In *ACL-22: Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, pages 367–375. Association for Computational Linguistics, Morristown, NJ.

Mann, W.C. and Matthiessen, C.M. (1985). Demonstration of the nigel text generation computer program. In J.D. Benson and W.S. Greaves (Editors), *Systemic Perspectives on Discourse*, volume 1, pages 50–83. Ablex, Norwood, NJ.

Mann, W.C. and Thompson, S.A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281.

Marinov, M. and Zheliazkova, I. (2005). An interactive tool based on priority semantic networks. *Knowledge-Based Systems*, 18(2-3):71–77.

Mauldin, M.L. (1984). Semantic rule based text generation. In *Proceedings of the 10th international conference on Computational linguistics*, pages 376–380. Association for Computational Linguistics, Morristown, NJ.

Maybury, M.T. (1989). GENNY: a knowledge-based text generation system. *Information Processing and Management*, 25(2):137–150.

Meehan, J.R. (1977). TALE-SPIN, an interactive program that writes stories. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 91–98. Cambridge, Massachusetts.

Montague, R. (1974). The proper treatment of quantification in ordinary english. In R. Thomason (Editor), *Formal Philosophy*, pages 247–270. Yale University Press.

Moore, J.D. and Swartout, W.R. (1989). A reactive approach to explanation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, volume 2, pages 1504–1510.

O'Donnell, M., Mellish, C., Oberlander, J., and Knott, A. (2001). ILEX: an architecture for a dynamic hypertext generation system. *Natural Language Engineering*, 7(3):225–250.

O'Donnell, M. (1994). *Sentence Analysis and Generation - a Systemic Perspective.* Ph.D. thesis, Linguistics Department, University of Sydney.

O'Donnell, M. (1995). Sentence generation using the systemic workbench. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 235–238. Leiden, The Netherlands.

Patten, T. (1988). *Systemic Text Generation as Problem Solving.* Studies in Natural Language Processing. Cambridge University Press, New York, NY.

Pustejovsky, J. (1991). The generative lexicon. *Computational Linguistics*, 17(4):409–441.

Pustejovsky, J. (1995). *The Generative Lexicon.* The MIT Press, Cambridge, Massachusetts.

Riesbeck, C. (1975). Conceptual analysis. In *Conceptual Information Processing*, volume 3 of *Fundamental Studies in Computer Science*, pages 83–156. North-Holland Publishing Company, Amsterdam, The Netherlands.

Savage, J., Weitzenfeld, A., Ayala, F., and Cuellar, S. (2009). The use of scripts based on conceptual dependency primitives for the operation of service mobile robots. In L. Iocchi, H. Matsubara, A. Weitzenfeld, and C. Zhou (Editors), *RoboCup 2008: Robot Soccer World Cup XII*, volume 5399 of *Lecture Notes in Computer Science*, pages 284–295. Springer Berlin.

Schank, R.C. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, 3(4):532–631.

Schank, R.C. (1975). Conceptual dependency theory. In *Conceptual Information Processing*, volume 3 of *Fundamental Studies in Computer Science*, pages 22–82. North-Holland Publishing Company, Amsterdam, The Netherlands.

Sowa, J. (1987). Semantic networks. In S.C. Shapiro, D. Eckroth, and G.A. Vallasi (Editors), *Encyclopedia of Artificial Intelligence*, volume 2, pages 1011–1024. John Wiley & Sons, New York, NY.

Swartout, B. (1982). GIST English generator. In *Proceedings of the Second International Conference on Artificial Intelligence*, pages 404–409. The AAAI Press.

Thomas, C. (2009). An exploration of semantic formalisms - part I: A comparison of First Order Predicate Calculus, Intensional Logic and Conceptual Structures. Technical Report No. 554, School of Computing, Queen's University.

Waltz, D. and Goodman, B. (1977). Planes: a data base question-answering system. *ACM SIGART Bulletin*, (61):24–24.

Wierzbicka, A. (1996). *Semantics: Primes and Universals.* Oxford University Press, New York, NY.

Winograd, T. (1983). *Language As a Cognitive Process: Syntax.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA.

Zock, M. and Adorni, G. (1996). Introduction. In A. Giovanni and M. Zock (Editors), *Trends in Natural Language Generation - An Artificial Intelligence Perspective*, volume 1036 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin.