The Double R Theory of Language Comprehension

Summary Paper

Jerry T. Ball, PhD
www.DoubleRTheory.com
Jerry@DoubleRTheory.com

Abstract

Double R Grammar is a linguistic theory of the grammatical encoding and integration of referential and relational meaning in English. Double R Grammar is fundamentally a Cognitive Linguistic theory (Langacker, 1987, 1991; Lakoff, 1988; Talmy, 2003) and the use of the term "grammar" encompasses both meaning and structure as it does in Cognitive Grammar (also known as Cognitive Semantics). Double R Process is a psycholinguistic theory of the processing of English text into Double R Grammar based representations. Double R Process is a highly interactive, bottom-up, lexically driven theory of language processing that is highly compatible with Kintsch's Construction-Integration Theory (Kintsch, 1998). Double R Theory is used to refer generally to Double R Grammar and Double R Process. Double R Model is a computational psycholinguistic implementation of Double R Theory using the Atomic Components of Thought – Rational (ACT-R) cognitive architecture and modeling environment (Anderson & LeBiere, 1998; Anderson et al., 2002). Double R Model is intended to validate the representation and processing commitments of Double R Theory and to form the basis for the development of large-scale, functional language processing systems.

Introduction

Double R Grammar is a linguistic theory of the grammatical encoding and integration of referential and relational meaning in English. Double R Grammar is fundamentally a Cognitive Linguistic theory (Langacker, 1987, 1991; Lakoff, 1988; Talmy, 2003) and the use of the term "grammar" encompasses both meaning and structure as it does in Cognitive Grammar (also known as Cognitive Semantics). Double R Process is a psycholinguistic theory of the processing of English text into Double R Grammar based representations. Double R Process is a highly interactive, bottom-up, lexically driven theory of language processing that is highly compatible with Kintsch's Construction-Integration Theory (Kintsch, 1998). Double R Theory is used to refer generally to Double R Grammar and Double R Process. Double R Model is a computational psycholinguistic implementation of Double R Theory using the Atomic Components of Thought – Rational (ACT-R) cognitive architecture and modeling environment (Anderson & LeBiere, 1998; Anderson et al., 2002). Double R Model is intended to validate the representation and processing commitments of Double R Theory and to form the basis for the development of large-scale, functional language processing systems.

Double R Theory improves on an earlier theory called PM, Propositional Model (Ball 1991, 1992). PM was focused on the representation of relational meaning and on the processing of text into propositional representations of relational meaning. The transition from PM to Double R Theory is based on the realization that referential meaning is a key dimension of meaning—largely implicit in PM—that should be made explicit and that should be accorded equal importance with relational meaning.

This paper begins with a description of Double R Grammar and focuses on the grammatical encoding of referential and relational meaning. It continues with a comparison of Double R Grammar to X-Bar Theory (Chomsky, 1970; Jackendoff, 1977). It is argued that Double R Grammar can provide a meaningful basis for the important grammatical generalization captured by X-Bar Theory, and in so doing avoid some of the pitfalls that have beset various versions of X-Bar Theory. This section continues with an overview of Langacker's Cognitive Linguistic description of the conceptual content of nominals and clauses (Langacker, 1991). The relationship between Double R Grammar and Langacker's conceptual schema for nominals and clauses is explored in some detail.

The paper continues with a description of Double R Process, highlighting the interactive, bottom-up, and lexically driven nature of the language processing system and its reliance on referential and relational schemas to determine the structure of texts being processed. The activation, selection, and integration phases of Double R Process are discussed. Double R Process is then compared to Kintsch's Construction-Integration Theory (Kintsch, 1998) and the two are shown to be highly compatible. The section concludes with a description of the ACT-R architecture (Anderson & LeBiere, 1998; Anderson et al., 2002) and a discussion of the appropriateness of using the ACT-R architecture and modeling environment for the development of language comprehension systems.

The third section of the paper discusses the Double R Model implementation of the theory using the ACT-R 5.0 modeling environment. The range of coverage of the model is explored. Examples of particular techniques used in the model (e.g. the creation of default rules and the use

of inheritance) are presented. Some suggested modifications to ACT-R which may be needed to support a more complete implementation of Double R Model are discussed.

The paper concludes with a consideration of the contribution of Double R Theory to the study of language comprehension. It is argued that the integrated representation and processing system emerging out of this research and the implementation of a testable model represent a significant achievement.

Double R Theory was originally called ЯR Theory. The ЯR symbol is an integration of the Cyrillic Я and the Roman R. It symbolizes the integration of referential and relational meaning which is the cornerstone of Double R Theory. Unfortunately, negative feedback and formatting problems led to the abandonment of this icon to symbolize the theory.

Double R Grammar: The Linguistic Theory

Double R Grammar is fundamentally a Cognitive Linguistic theory (Langacker, 1987, 1991; Lakoff, 1988; Talmy, 2003). Grammar encodes meaning (Langacker, 1987, 1991). All grammatical elements have a semantic basis, including parts of speech, grammatical markers, phrases and clauses (Langacker, 1987, 1991; Taylor, 1992). Our understanding of language is embodied and based on experience in the world (Lakoff, 1987; Lakoff and Johnson, 1980). Categorization is a key element of linguistic knowledge, and categories are seldom absolute—exhibiting, instead, effects of prototypicality, basic level categories (Rosch, 1978), family resemblance (Wittgenstein, 1953), fuzzy boundaries, radial structure and the like (Lakoff, 1987; Taylor, 1992). Our linguistic capabilities derive from basic cognitive capabilities—there is no autonomous syntactic component (Chomsky, 1957, 1965; Frazier & Fodor, 1978) separate from the rest of cognition. Knowledge of language is for the most part learned and not innate (Langacker, 1987; Tomasello, 1999). Abstract linguistic categories (e.g. noun, verb, nominal, clause) are learned on the basis of experience with multiple instances of words and expressions which are members of these categories, with the categories being abstracted and generalized from experience. Also learned are schemas which abstract away from the relationships between linguistic categories. Over the course of a lifetime, humans acquire a large stock of lexical items as well as schemas at multiple levels of abstraction and generalization, representing knowledge of language and supporting language comprehension. Whereas knowledge of individual lexical items is the basis of **lexical semantics**, schematic knowledge of the associations between lexical items constitutes what might be called **grammatical semantics**, although schemas are, for the most part, associated with specific lexical items.

Grammar is the symbolization of meaning. Discussion of grammar goes hand in hand with discussion of the meaningful consequences of grammatical variation. The linear stream of English text encodes multiple dimensions of meaning. The encoding of multiple dimensions of meaning in a single linear dimension results in trade-offs in encoding across dimensions and variation within a given dimension across different grammatical contexts.

Two key dimensions of meaning that get grammatically encoded are referential meaning and relational meaning. The grammatical encoding of referential and relational meaning is described in terms of the four basic functional categories: **specifier**, **head**, **modifier**, and **complement**. These terms are borrowed from X-Bar Theory (Chomsky, 1970), but are given a semantic basis. The functional category of complement is further classified in terms of the functional roles that complements fulfill within a clause: **subject**, (**direct**) **object**, **indirect object**, and **complement** (e.g. clausal complement or prepositional phrase complement). Double R Grammar gives these

functional categories definitions based on their use in representing referential and relational meaning. A specifier is a word or expression (or morphological marker) that establishes the referential type of the expression it specifies. The term "specifier" as used in Double R Grammar is compatible with Chomsky's (1970) suggestion that determiners and auxiliaries are typical specifiers (ibid, p. 210), a position he later rejected (e.g. Chomsky, 1995). A head is the semantically most significant element of an expression and determines the relational type—where relational type is used generally to encompass non-relational objects—of that expression (called a **Type Specification** in Langacker, 1991). A modifier is a linguistic element that provides supporting information about the head it modifies. A modifier may constrain the relational type of the head, perhaps further classifying that type into a subtype, but the modified expression retains the overall type of the head. A complement is a referring expression that functions as an argument to a relational element. Clauses are headed by relational elements and the complements which occur in a clause are the arguments of the relational head.

The joint consideration of referential and relational meaning leads to integrated representations of referential and relational meaning of two basic types: **situation referring expressions** which typically correspond to clauses and **object referring expressions** which typically correspond to nominals. **Predicate referring expressions** are situation referring expressions without their associated complements, however, predicate referring expressions seldom occur in isolation without their complements.

A predicate referring expression is headed by a word or expression describing a relation. The relational head is typically a verb, adjective or preposition and occasionally an adverb—the four basic relational parts of speech (cf. Langacker, 1991). A specifier, providing information about tense and/or modality (and negation), combines with a relational head to form a predicate referring expression. The predicate referring expression takes on the functional role of a **predicate** within a situation referring expression. The predicate typically combines with one or more complements (or arguments) in forming a situation referring expression. The terms **complement** and **argument** are used synonymously in Double R Grammar, and they are referring expressions. **Schemas** are associated with relational words and expressions and reflect the number, typical position, and functional role of the complements of the relational element. For example, the verb "hit" is associated with the schema **|subj hit obj|** reflecting the occurrence of a subject complement preceding the verb and an object complement following the verb. Within this schema, **hit** is itself a schema for the possible instantiations of a predicate headed by the verb "hit" (e.g. "hit", "is hitting", "could have hit", "did not hit") and **subj** and **obj** are complement schemas. Non-relational words and expressions (e.g. nouns, pronouns, proper nouns, nominals) do not typically evoke such schemas, although the possibility of schemas containing unexpressed relationships is not precluded (e.g. a **|noun noun|** schema). In addition to relational schemas, referential schemas like **|specifier head|** reflect the referential structure of the referring expressions in a text.

An object referring expression is headed by a word or expression describing a type of object, or by a word or expression describing a type of relation or situation that is viewed objectively. A specifier combines with the head to form an object referring expression, although certain types of heads (e.g. pronoun, proper noun, plural common noun, mass noun) are inherently or morphologically specified and do not require a separate specifier. Neither the head of an object referring expression nor the object referring expression itself takes any complements (since the head is viewed as an object and not a relation). However, both the head and the overall object referring expression may combine with one or more modifiers. Further, it is possible for a

relational expression containing complements to be objectified and treated as the head of an object referring expression, giving the appearance that the head takes complements. For example, in the expression "this giving money to strangers" (a **poss-ing** construction), the entire sub-expression "giving money to strangers" (e.g. Pullum, 1991) is objectified and functions as the head. That is, the complements "money" and "to strangers" combine with "giving" before the expression as a whole is objectified. The typical object referring expression is headed by a noun, specified by a determiner, and has the form of a noun phrase. However, the head of an object referring expression need not be a noun and the object specifying function need not be provided by a determiner.

In addition to situation referring expressions, predicate referring expressions and object referring expressions, there is a collection of less basic types of referring expressions: **location referring expression**, **direction referring expression**, **time referring expression**, **path referring expression**, **manner referring expression**, **reason referring expression**, **and measure referring expression** (and perhaps a few others). These less basic referring expressions typically function as modifiers, although they may occasionally function as complements (e.g. "on the table" in "he put the ball on the table").

The full range of referring expressions corresponds to the wh-words in English as follows:

Who – (human) object referring expression
What – object referring expression; predicate referring expression; situation referring expression
Where – location referring expression; direction referring expression; path referring expression
When – time referring expression
Why – reason referring expression
How – manner referring expression
How much/many – measure referring expression

Jackendoff (1983, 2002) presents a similar ontology of conceptual categories (based in part on the wh-words in English—1983, p. 53) without committing to their having a referential basis, although his **referentiality principle** that "unless there is a linguistic marking to the contrary, all phrases that express conceptual constituents are referential" (1983, p. 70) leads to a similar result. However, Double R Grammar takes a different position in arguing that specifiers mark referentiality, but allows that some words and expressions may be inherently or morphologically specified (e.g. wh-words, pronouns, proper nouns, plural common nouns, mass nouns, deictic words and expressions, present tense and past tense verbs).

The relationship between a specifier and a head is primarily referential, with the specifier dominating the head in determining the referential status of the overall expression. For example, in the expression "the kick" the specifier "the" determines the expression to be an object referring expression even though the head "kick" describes a type of action. Specifiers are classified in terms of the types of referring expressions they specify. The two basic types of specifiers are **predicate specifiers** which specify predicate referring expressions and **object specifiers** which specify object referring expressions. Clausal complements may be specified by a **complementizer** (e.g. "that" in "I believe that he likes you"), especially when the complementizer indicates that the clausal complement is being viewed objectively (i.e. an **objectified situation referring expression**). The less basic types of referring expressions are

typically specified by prepositions or adverbs which also function as the heads of such expressions (e.g. “on” in “on the table”).

The relationship between a head and a modifier is primarily relational (where relational again encompasses non-relational objects) with the head determining the relational type of the overall expression and the modifier constraining that relational type. For example, in the expression “red ball”, the modifier “red” combines with the head “ball” to constrain the relational type of “ball”, but the overall expression still describes a type of ball. The term **endocentric** (Bloomfield, 1933) is used to describe expressions in which the head of the expression determines the type of the expression as a whole. The head-modifier relationship preserves endocentricity.

Whereas the head-modifier relationship preserves endocentricity, the specifier-head relationship need not. For example, in the expression “the running (of the bulls)” the verb participle “running” functions as the head of an object referring expression despite its verbal status. It is the specifier “the” which determines the referential type of the overall expression and not the head “running”. In general, referential type is more closely aligned with the notion of syntactic type (e.g. noun phrase, verb group) than is relational type—at least where they differ—and the specifier-head relationship is **exocentric** in this case.

To retain the notion of endocentricity for expressions like “the running (of the bulls)”, “running” would need to be reanalyzed as a noun in this expression and nouns and noun phrases (or nominals) would have to be considered the same (syntactic) type (as in X-Bar Theory), but Ball (2003a) argues that such an approach is circular since words of most (if not all) parts of speech and many different types of expression may function as the head of a noun phrase and would need to be classified as nouns—and the only basis for doing so is because they occur as the head of a noun phrase. Consider

> The *running* (present participle) of the bulls
> The *injured* (past participle) were taken to hospital
> The *sad* (adjective) are in need of cheering up
> The *cheering up* (verb participle + particle) of the sad
> The *buy out* (verb + particle) of the corporation
> The *up and down* (conjoined prepositions) of the elevator
> The *Fillmores* (proper noun)
> The *ayes* (adverb) have it
> The man took a *walk* (verb) around the park (Dixon, 1991)

In some of these examples (e.g “ayes”, “Fillmores”), the head of the object referring expression is pluralized and it may appear that these words are indeed nouns since the ability to pluralize is often cited as a feature of nouns. However, if pluralization is a feature of the head of an object referring expression and not of nouns per se, then pluralization cannot be used as a definitive test of noun hood. The fact that nouns are the prototypical heads of object referring expressions explains their ability to pluralize, but does not preclude the pluralization of words of other parts of speech when they head object referring expressions. The head of an object referring expression is viewed objectively whether it is a noun, verb, adjective, adverb or preposition. This objective viewpoint allows the head to be pluralized when multiple instances of the relation expressed by a verb, adjective, adverb or preposition are conceptualized.

Besides the requirement to treat all heads of noun phrases as nouns, endocentricity requires that a noun phrase be the same type as the head noun it contains. But there are clear

distributional differences between noun phrases and nouns which suggest that they are not the same grammatical type. Singular count nouns cannot substitute for full-fledged nominals (e.g. "The book is long" vs. "Book is long") because they lack the specification that creates a full-fledged nominal (i.e. an object referring expression) from a head noun. The possessive takes a noun, not a full nominal (e.g. John's book vs. John's the book) because the possessive provides the specification that creates a full-fledged nominal and this conflicts with the function of the determiner "the".

The notion of head used in Double R Grammar is more closely aligned with the notion of **stem** (or **root**) in morphology than is true of most other grammatical approaches. It is the distinction between the head-modifier and specifier-head relationship which makes this closer alignment possible. The head or stem is the semantically most significant element that determines the semantic **profile** (Langacker, 1987, 1991) of the composite expression. In a word like "blackbird", "bird" is the stem which determines the semantic type of the composite lexical item with "black" constraining that semantic type. In this example of derivational morphology, the correspondence is to the modifier-head relationship and "blackbird" and "bird" are both nouns. On the other hand, the word "easily" combines the adjective "easy" with the adverb marker "-ly" with "easy" determining the semantic type of the word, but with "-ly" determining the part of the speech of "easily". In this latter case of derivational morphology, the correspondence is to the specifier-head relationship. In essence, there are two basic mechanisms of combination, one which retains endocentricity (i.e. the head or stem determines the type of the overall composite), and one which does not (i.e. the specifier or affix determines the type). Syntactic approaches which adopt a strong notion of endocentricity (e.g. X-Bar Theory, Dependency Grammar, HPSG) are unable to align themselves with morphology where exocentricity is a commonly attested pattern (e.g. the creation of adverbs from adjectives via the addition of "-ly"; the creation of nouns from verbs via the addition of "-er").

The relationship between a head and its complements is primarily relational in nature. Only relational heads take complements and those complements are full referring expressions. The relational nature of a word or expression is a linguistic conventionalization. That is, the number and type of complements that a relational word or expression takes is a conventionalization of the situation that the word or expression is used to describe. That conventionalization provides a particular perspective on the situation (Fillmore, 1977) that may vary from language to language (and within a language), but is nonetheless meaningfully motivated. Such conventionalization may be driven by psychological limits on the number of separate chunks of information that can be separately entertained at one time (i.e. short-term working memory limitations).

The relationship between a specifier and a relational head and between a relational head and its complement(s) is largely, though not entirely, orthogonal. The specifier-head relationship is primarily referential, whereas, the head-complement relationship is primarily relational. Grammatically, there is often a closer affinity between a specifier and a relational head than there is between a relational head and its complements. For example, a predicate specifier combines with a relational head to form a predicate referring expression and a predicate referring expression combines with its complements to form a situation referring expression. The closer affinity of a predicate specifier and relational head is suggested by the fact that the infinitive marker "to" which specifies a non-finite predicate referring expression, also sanctions the non-occurrence of the subject complement as in the infinitive phrase "to go" in the clause "I want to go" (an example of the interdependence of referential and relational meaning). Although, the infinitive marker sanctions the non-occurrence of the subject complement, that complement is

typically retrievable in the context of use of the expression. On the other hand, in the expression "the book is on the table" the relationship between the predicate specifier "is" and the relational head "on" may be subordinated to the relationship between "on" and its object complement "the table". If so, "on" first combines with "the table" to form a higher order relation before combining with the predicate specifier "is" (and the subject complement).

Complements can be contrasted with modifiers. Complements are sanctioned by the relational elements they combine with to form composite expressions and they are typically obligatory elements of the expressions they occur in—although a relational element may license alternative schemas in which the complement does not occur. Modifiers are not sanctioned by a relational element and are always optional. Complements are referring expressions. Modifiers may or may not be referring expressions. The typical complement is a non-relational object referring expression (e.g. nominal). The typical modifier is a relational expression (e.g. attributive adjective, prepositional phrase or adverb). A complement profiles itself, although the profile of the complement does not project to the composite expression. It is the relational head with which the complement combines that projects the profile and type of the composite expression. A modifier profiles the head it modifies and does not project the type of the composite expression (although it may constrain that type into a subtype). Thus, neither a complement nor a modifier projects the type of the composite expression. Although the distinction between a prototypical complement and a prototypical modifier is clear cut, the distinction is somewhat fuzzy at the boundaries. The expression "on the table" functions as a complement in "he put the ball on the table" and a modifier in "the ball on the table". The complement status of "on the table" in the first expression stems from the fact that the verb "put" sanctions a locative complement, whereas the nominal "the ball" does not. In the expression "he gave Mary the ball", "Mary" is a complement of "gave", whereas in "he gave the ball to Mary", "to Mary" may or may not be a complement. The treatment of "to Mary" depends on which schema is used to construct a representation. Assuming a schema for "gave" like |**subj give obj ppcomp**| which includes a (prepositional phrase) complement (functioning as a direction referring expression) following the object, "to Mary" can be instantiated as that complement. However, there may also be a more specialized schema like |**subj give obj to obj**| which explicitly represents "to" and only abstracts from the object of "to". In this schema, "to" participates as part of the predicate "give…to"—similar to the way verb particles contribute to predicates—but "to" also sanctions an additional object (unlike verb particles). This more specialized schema is likely to be preferred over the more abstract schema which treats "to Mary" as a complement. Finally, in the expression "he threw the ball to Mary", "to Mary" is likely functioning as a directional modifier of the predicate "threw" (assuming the lack of a schema for "throw" which incorporates a prepositional phrase complement like "to Mary" or a more specialized schema like |**subj throw obj to obj**|).

The subject complement is the only complement of an **intransitive predicate** and the first complement of a **transitive predicate**. The direct object is the second complement of a transitive predicate or relational modifier (e.g. prepositional phrase modifier). Langacker (1991) defines the subject as the **primary figure** or **trajector** in a profiled relation and the direct object as the **secondary figure** or **landmark**. The trajector is the participant in a relation from which the relation "flows" and this flow is towards the landmark. That is, the trajector is viewed as the source of the relation and the landmark is viewed as the target of the relation. Two additional complements are possible, the indirect object and an additional complement (i.e. in addition to the direct and indirect object). The indirect object occurs with relations like "give" as is the case

for “Mary” in “he gave Mary the book”. Langacker treats the indirect object as the secondary figure or primary landmark, demoting the direct object in such expressions to the status of secondary landmark. In double object constructions, the relation flows from the trajector thru the primary landmark (i.e. indirect object) to the secondary landmark (i.e. direct object). Thus, “Mary” has the status of primary landmark in the example above and “to Mary” has the status of secondary landmark in “he gave the book to Mary” with “the book” being the primary landmark. Clausal complements can function as trajectors (e.g. “that he likes you” in “that he like you is nice”), primary landmarks (e.g. “he likes you” in “I believe he likes you”), secondary landmarks (e.g. “he likes you” in “he told me he likes you”) and even tertiary landmarks (e.g. “he likes you” in “I bet you $50 he likes you”) (the “bet” example is adapted from Steedman, 2000). In the last example, the clausal complement constitutes a distinct functional category.

Many relational expressions participate in schemas with a reduced number of complements as in the expression “he bet me” which is missing the second object and clausal complement. In such schemas, the participants which are explicitly encoded are profiled to the exclusion of the missing participants. It is even possible for all participants in a relation to be excluded, in which case only the relation itself is profiled. This is what happens in object referring expressions with relational heads where all complements are left unexpressed as in “the bet was late” (where “bet” refers to the act of betting and not the amount of the bet).

The basic parts of speech in English are noun, verb, adjective, adverb and preposition. Of these, all are relational except for nouns (Langacker, 1987; 1991 p. 5). Nouns typically function as the heads of object referring expressions, however, they may also function as modifiers. Verbs typically function as the heads of predicate referring expressions, however, they may also function as the heads of object referring expressions in which case they are viewed objectively. It is interesting to note that, of the relational parts of speech, it is action verbs which most readily function as heads of object referring expressions. The short duration of action verbs makes them highly suitable for objectification as in “the kick”, “the punch”, “the throw”, etc., even though action verbs are the most typical verbs. Verbs (especially verb participles) may also function as modifiers in object referring expressions. Adjectives typically function as the heads of predicate referring expressions (i.e. predicate adjectives) and also as modifiers in object referring expressions (i.e. attributive adjectives). Adverbs typically function as modifiers of relations and situations and occasionally as predicates. Prepositions in combination with their objects typically function as modifiers of object referring expressions, modifiers of the heads of predicate referring expressions (i.e. predicate modification) and modifiers of situation referring expressions (i.e. situational modification). Prepositions may also function without an object (i.e. verb particles) as modifiers of the heads of predicate referring expressions.

As an example of a Double R Grammar based representation, consider the representation of the expression “he gave me the book”

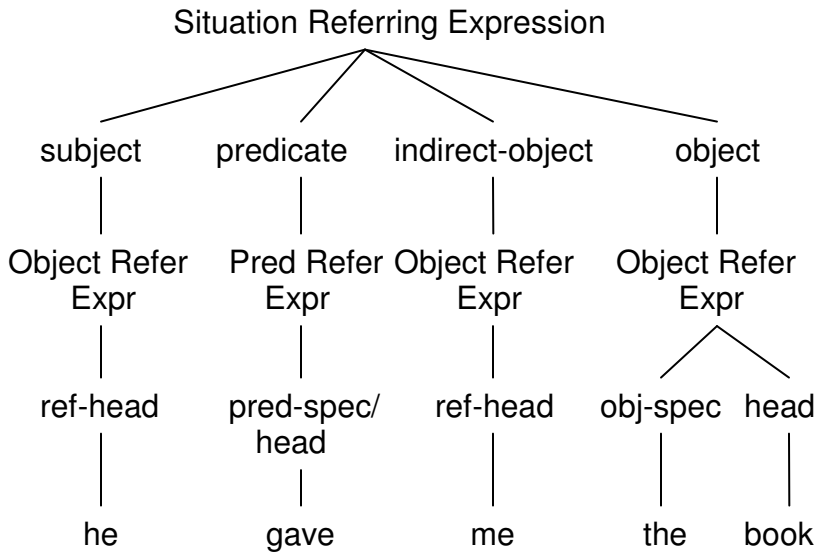

Figure 1: a situation referring expression with multiple complements

In this example, the pronouns “he” and “me” are inherently referring object referring expressions that function as the subject and indirect object, respectively (ref-head = referring head). The determiner “the” provides the specification for the object referring expression “the book” (obj-spec = object specifier) which functions as the (direct) object. The past tense verb “gave” is a morphologically specified (i.e. past tense) predicate referring expression that functions as the main predicate (pred-spec = predicate specifier). In this example, the integration of referential and relational meaning is relatively straightforward with the object referring expressions “he”, “me” and “the book” corresponding to the arguments of the predicate “gave”. In general, the integration of referential and relational meaning will not be so straightforward and trade-offs in the encoding will result. For example, consider the expression “the book on the table” which is represented by

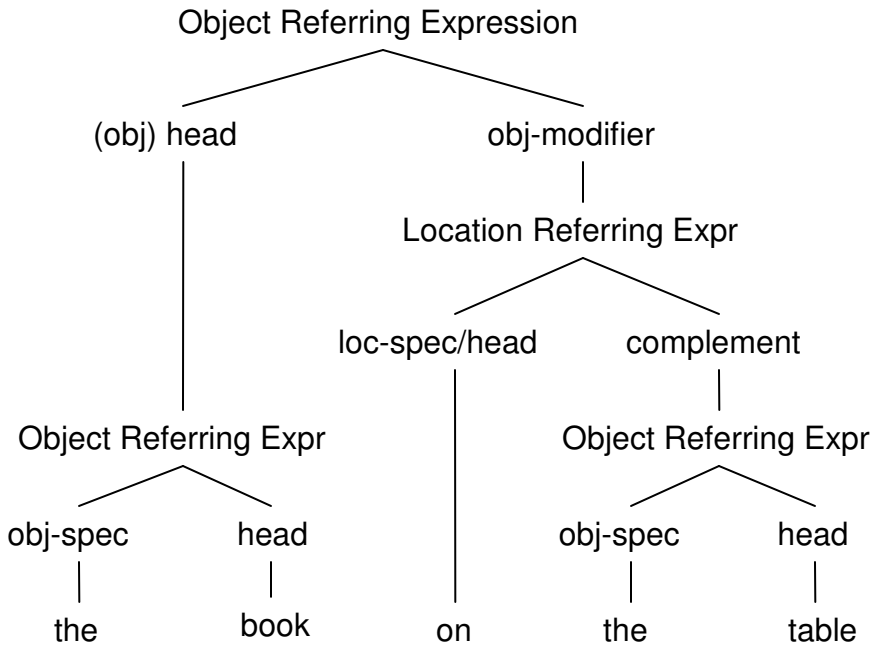

Figure 2: an object referring expression with a post-head modifier

In this representation, the relational meaning of "on" with respect to the complement "the book" is only implicit in the modifying function of "on the table" (obj-modifier = object modifier). It is also of interest to note that the preposition "on" functions to both specify a location referring expression and to head that expression (loc-spec = location specifier). In general, words that function as specifiers come from closed class parts of speech like determiner, auxiliary verb, preposition, and pronoun and tend to be short, frequently occurring words. That specifiers tend to be short, frequently occurring words from closed class parts of speech is a reflection of their importance for identifying grammatical structure. Their membership in closed class parts of speech makes them easier to learn. Their small size makes them quicker to perceive (once they are learned). Gomez and Gerken (2000) suggest that words from closed class parts of speech (e.g. "the") and morphological markers (e.g. the plural marker "-s") provide perceptual cues that, once learned, support the learning of open class categories like noun and verb. Specifiers make the complexity of adult grammatical competence possible. Closed-class parts of speech exist because there are only highly restrictive processes for creating new members of these classes. Instead, they are the basis (along with morphological markers) for the productive processes that create new members of the open-class parts of speech.

Note also that in the example the prepositional phrase "on the table" modifies the object referring expression "the book" and not just the head of that expression "book". This representational commitment is motivated by the fact that relations take referring expressions as complements with "the book" functioning as an implicit complement of "on". Thus, "the book" is the head with respect to the modifier "on the table", but it is a complement with respect to the relation "on". Typically, post-modification in object referring expressions is of an object referring expression and not just a head, whereas pre-modification is of the head as in "the very old man" which is represented by

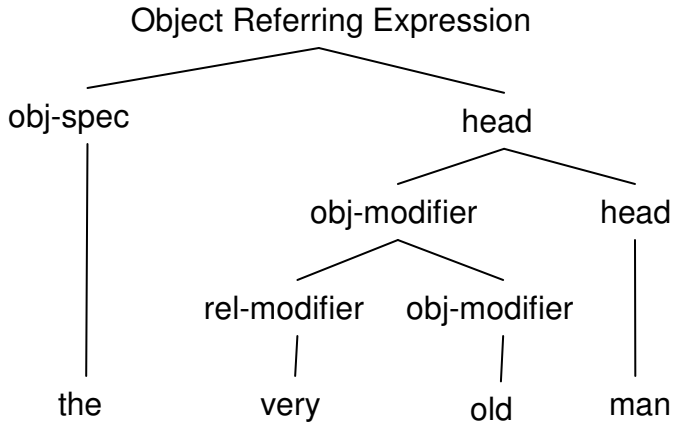

Figure 3: an object referring expression with a pre-head modifier

Note that the adverb "very" functions as a relation modifier modifying the relation "old" (with "old" functioning as an object modifier). "Very old" in turn modifies the head "man". In pre-modification, the modifier-head relation dominates and "man" is not a complement of "old" (since "man" is not, in and of itself, a referring expression).

The specifier function need not be filled by a single lexical item. Predicate specification may include a modal auxiliary, up to three other auxiliaries, a negative, and even an embedded adverbial modifier (e.g. "clearly" in "he is clearly not happy"). The specifier function may also be morphologically based (e.g. the past-tense marker). Consider the expression "the boy could not have been hitting the ball" which is represented by

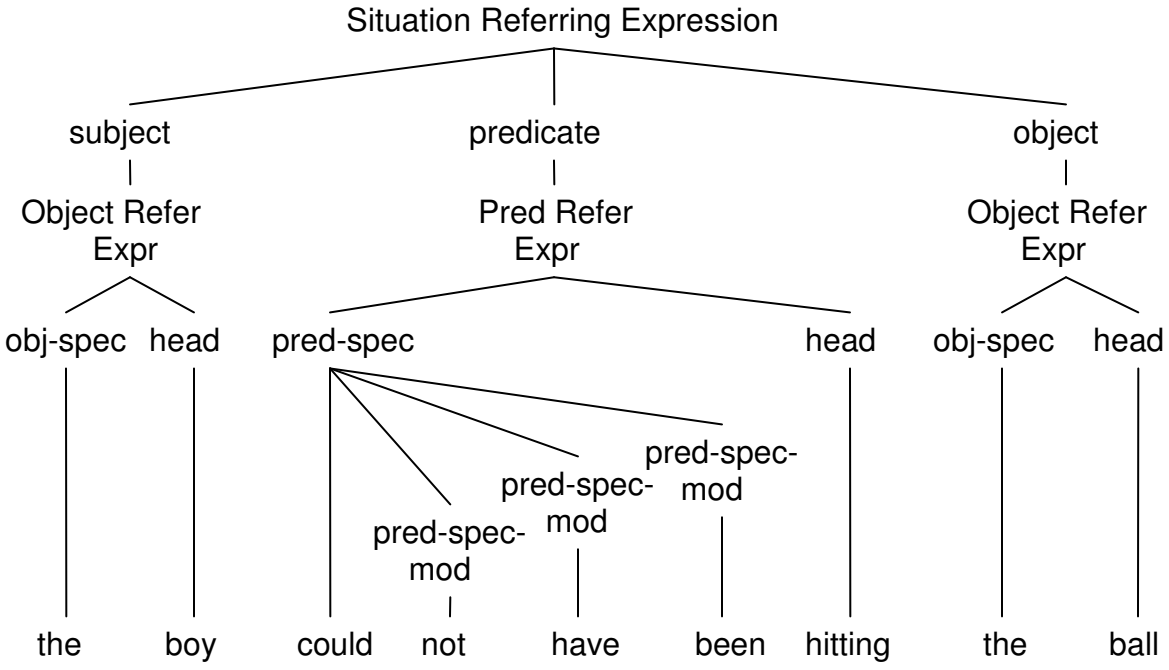

Figure 4: a complex predicate specifier

In this example, the predicate specifier consists of the modal auxiliary "could", the negative "not", and the auxiliary verbs "have" and "been". The predicate referring expression refers to a possibly negative past state of hitting and combines with the object referring expressions "the boy" and "the ball" to form a situation referring expression.

Object specification may also consist of more than one lexical item or morphological marker. Consider the expression "the first two books". Using grammatical features (where grammatical is understood as being meaningful) to represent different types of object specification, this can be represented by

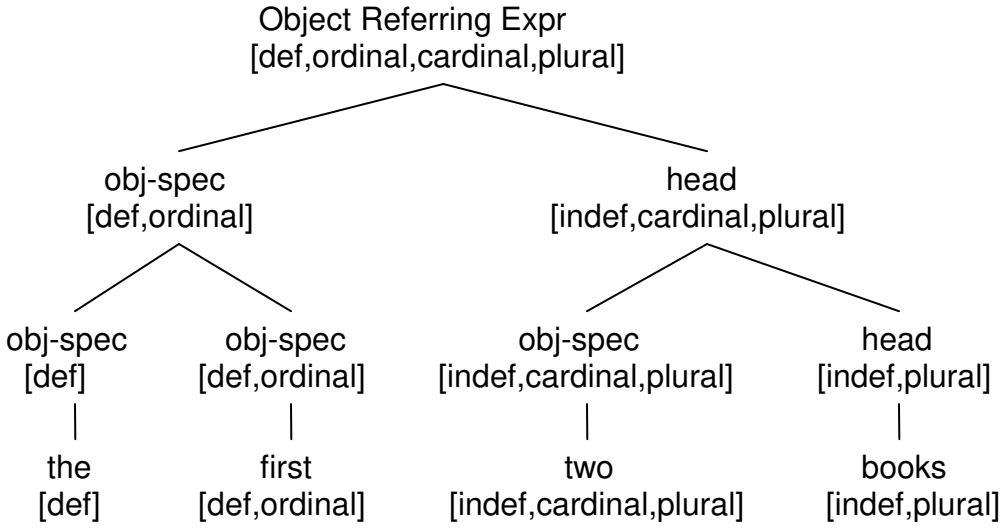

Figure 5: adding grammatical features to the representation

Note that the overall object referring expression is definite despite the indefinite feature on "books" and the indefinite specification of two instances provided by "two". "Two" combines with "books" and the combination may function as a fully specified (indefinite) object referring expression, although in the example further specification occurs. "First" could combine with "two books" forming "first two books", but the result is not a fully specified object referring expression unlike "two books" and this combination is dispreferred for this reason (i.e. it is assumed that a specifier does not combine with a object referring expression to form an expression that cannot also function as an object referring expression). Instead, "the" combines with "first" to form a combined definite object specifier, which combines with "two books" to form a definite object referring expression. An alternative representation could have "the first" combine with "two" to form a definite object specifier "the first two" which then combines with "books" to form an object referring expression. This alternative representation could even be preferred with appropriate emphasis (e.g. "*the first two*…books"). The specifier and the head can be viewed as providing two **poles**—a referential pole and a relational pole (which encompasses non-relational objects)—to which other lexical items are attracted. Lexical items of particular parts of speech are preferentially attracted to one pole or the other (based on their meaning), although such preferences can be overridden with appropriate emphasis or context. That ordinal and cardinal numbers are preferentially specifiers is suggested by the fact that they occur before modifiers (e.g. "the first two blue books"). In addition, they contribute to the determination of the referent of the overall expression. However, an expression like "the blue two books" is not all

that awkward, and in such a case “two” may be functioning as a modifier (or “blue” may be modifying the fully specified object referring expression “two books”). Multiple specification is presumed to be the norm in Double R Grammar, and modification (and further specification) may occur external to specification, subject to certain limitations. For example, a definite object referring expression cannot be further specified by an indefinite specifier and in most cases addition of a second definite object specifier is redundant (or would be inconsistent) (e.g. “these all books”). It is for this reason, that further specification of definite object referring expressions is extremely limited, although not entirely non-existent (e.g. “all” in “all these books”). Despite the possibility of multiple specification, the distinction between a specified head that can function as a referring expression and an unspecified head that cannot, is still meaningful and important.

Cardinal and ordinal numbers are subtypes of quantifier. Quantifiers, more generally, are considered to be part of the specification in object referring expressions. Consider the expression “many books” which is represented by

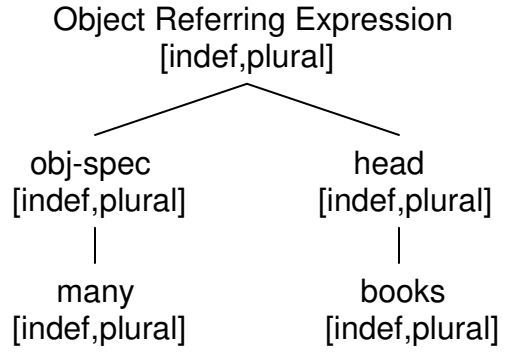

Figure 6: overlapping grammatical features

The boundary between specification and modification is somewhat fuzzy. Determiners, and quantifiers typically function as object specifiers. Adjectives and verbal participles typically function as object modifiers. Modals, auxiliaries and negatives typically function as predicate specifiers. Adverbs typically function as predicate modifiers. The requirement for “do support” in negation (e.g. “he did not go”) and the contracted forms of negation (e.g. “couldn’t”, “shouldn’t”, “won’t”) suggest the preferred treatment of the negative as part of the predicate specification. However, consider the expression “he could not go” which is ambiguous between a preferred reading in which “not” combines with “could” to form a predicate specifier (i.e. the preferred or unmarked reading) and a dispreferred reading in which “not” functions as a predicate modifier modifying “go” as in “he could NOT GO” (i.e. “he could (decide) not (to) go”) with “could” providing the predicate specification and “not go” functioning as the modified head. Further, quantifiers may function as heads of object referring expressions as well as object specifiers as is the case for “some” in “some of the books”. This combined object specifier/head functionality is typical of pronouns, and quantifiers functioning in this manner are typically considered to be pronouns (although an alternative is to view them as quantifiers functioning as full object referring expressions, or what Langacker calls “pronominals”). Likewise, auxiliary verbs may function as the heads of predicate referring expressions as in “he did it” in which case they combine the predicate specifier/relational head functionality just like other present and past

tense verbs. Or consider "he didn't do it" in which the first occurrence of do (i.e. "did") is functioning as a predicate specifier and the second instance is functioning as a relational head or predicate.

The encoding of referential meaning is a key element of Double R Grammar. Of course, referring expressions must ultimately refer to something. Reference in Double R Grammar is to objects and situations in a **situation model** (adapting Kintsch's, 1998, terminology) which is a mental representation of the real and imagined world. The situation model is comparable to Johnson-Laird's Mental Models (1983) and to Jackendoff's (1983) projected world. To the extent that the evolving situation model corresponds to the broader (and less dynamic) mental model of the real world that humans maintain, referring expressions refer to objects and situations in the real world (as represented in the projected world). However, referring expressions may refer to objects or situations in the situation model which do not correspond to objects or situations in the broader mental model of the real world. This makes it possible to treat situation referring expressions like "the boy could not have been hitting the ball" and object referring expressions like "no one" in "no one is here" as referential. In the situation model corresponding to "the boy could not have been hitting the ball" a possibly negative state of hitting involving two objects is referred to which does not correspond to any situation in the broader mental model of the real world, and in "no one is here" a person and a location are referred to, however, the person referred to in the situation model is understood not to correspond to anyone in the broader mental model of the real world (although the location may refer to a real world location). The creation of a situation model corresponding to a text and its relationship to the broader mental model of the real world is an active area of research (Ball, 2003b). Key influences on that research include Johnson-Laird (1983), Kintsch (1998), Langacker (1987, 1991), and Jackendoff (1983, 2002). Logical and formal approaches to the handling of reference (and quantification) are largely rejected, although considerable insight may be gleaned from that research (e.g. Thomason and Stalnaker, 1973; Barwise and Perry, 1983; Kamp and Reyle, 1993).

Double R Grammar makes use of schemas (using Langacker's terminology) for the representation of knowledge of language. In Double R Grammar there is an abstract schema of the form **|subj pred obj|** (**subj** is short for object referring expression filling the subject role, **pred** is short for predicate referring expression filling the predicate role, and **obj** is short for object referring expression filling the direct object role) which represents knowledge about the linear encoding, number and type of arguments which are associated with transitive predicates. There is also likely to be a more concrete schema of the form **|subj *hit* obj|**, reflecting specific knowledge about the transitive verb "hit." And even more concrete schemas like **|subj *hit the nail on the head*|** are possible. Thus, Double R Grammar assumes the existence of schemas at multiple levels of abstraction and generalization. Figure 7 lists some possible schemas for the verb "hit".

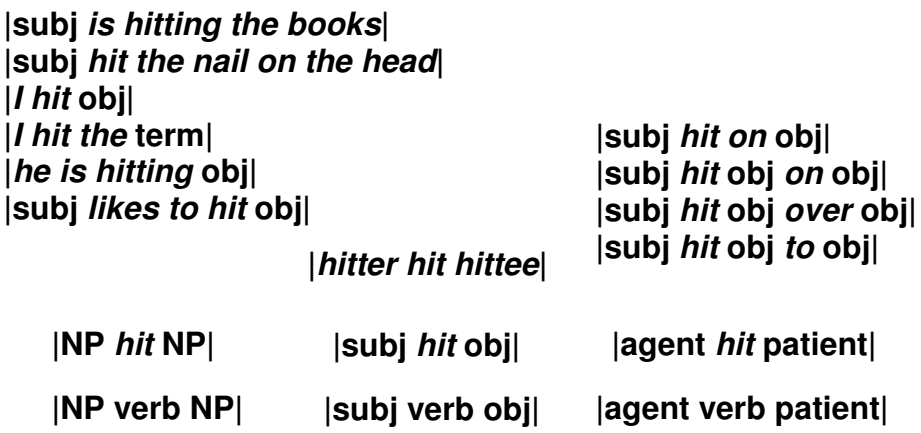

Figure 7: Some Possible Schemas for the Verb *Hit*

Of interest to note is that a schema which contains specific lexical items might be said to be part of the lexicon (assuming the schema is addressable via the lexical item it contains), whereas a schema which does not contain any specific lexical item might be said to be part of the grammar (Jackendoff, 2002, makes a similar point). But if abstract schemas like **|subj pred obj|** are directly associated with specific lexical items, this distinction loses its force.

Although all these schemas (and more) are possible, Double R Grammar assumes that more concrete schemas like **|subj *is hitting the books*|** take precedence over more abstract schemas like **|subj pred obj|** in language comprehension since they will be more strongly activated by matching inputs as when a sentence like "he is hitting the books" is processed. Abstract schemas take on a more important role when processing novel expressions not previously experienced and unknown words. Double R Grammar also assumes that schemas containing functional role categories like subject and object are preferred over schemas containing syntactic categories like NP and VP. This follows from Double R Grammar's assumption that functional categories like subject and object are meaning based and not purely syntactic (although they are grammatically marked) and from the priority of semantics assumption and the lack of a distinct syntactic component. Finally, Double R Grammar assumes that schemas containing functional categories which are grammatically or morphologically marked (e.g. *hitter*, *hittee*, subject and object) will be more easily abstracted from and matched to inputs then schemas containing categories which are not grammatically marked (e.g. agent and patient).

The existence of schemas at multiple levels of abstraction is a generalization of the distinction between **semantic memory** and **episodic memory**, with semantic memory being composed of schemas which have been abstracted away from specific episodes and entities, and episodic memory containing schemas corresponding to specific events and entities. Further, multiple level schemas are consistent with the distinction between **prototypes**, **exemplars** and **specific instances**. Prototypes correspond to the most typical (and central) schemas of a given type and are abstracted from previous experience (and may be generalized), exemplars cover a wider range of possible schemas and are more closely tied to the actual experience (and less likely to be generalized), and specific instances are directly tied to experience. Finally, multiple level schemas can represent the distinction between **types** and **tokens**, with types corresponding to abstract schemas that are generalized from past experience and tokens corresponding to specific schemas that are directly tied to experience. Langacker (1991) argues that tokens are essentially just types that happen to refer to specific instances. That is, from the type-subtype

perspective (i.e. dimension of meaning), tokens are subtypes of a higher level type, although they differ from other types in also instantiating an instance of the type in the domain of instantiation (i.e. another dimension of meaning). His unification of types and tokens simplifies the (partially) compositional semantics he espouses, allowing for the creation of a uniform type hierarchy from the most general concepts (e.g. thing or process) to specific instances (e.g. "Richard the Lionheart", "World War II"). Jackendoff (1983) attempts an alternative unification of types and tokens in arguing that they are grammatically realized via the same grammatical forms (e.g. in "a cat is a mammal", "a cat" refers to a type and in "a cat sat on the mat", "a cat" refers to an indefinite token), and, therefore, not grammatically distinguished. Instead, tokens and types are only distinguished in conceptual structure. Double R Grammar maintains a grammatical distinction between linguistic types and tokens, arguing that the specifier functions to establish a token (or instance) given a head of a particular type. In Double R Grammar, both uses of "a cat" above represent linguistic tokens and not types, and only linguistic tokens function as referring expressions. However, in Double R Grammar, a referring expression may refer to a nonlinguistic type (a possibility that Jackendoff argues against) as well as a nonlinguistic token. That is, there are nonlinguistic types and tokens (as well as exemplars and possibly prototypes) in the situation model to which referring expressions refer, as well as linguistic types and tokens. Where Langacker and Jackendoff have a single level of conceptual (i.e. Jackendoff's conceptual structures) or semantic (i.e. Langacker's semantic pole) representation, Double R Grammar has linguistic representations of meaning that are ground in nonlinguistic representations of the referents of referring expressions.

In terms of how we acquire schemas, Double R Grammar agrees with Langacker (1987). An individual is exposed to specific instances of language use and initially acquires rather specific schemas corresponding to those instances. Based on repeated exposure to similar instances with minor differences he or she is able to abstract away from the differences, forming abstract schemas which capture the similarities and abstract away from the differences. Abstraction then provides the mechanism for producing or recognizing new instances to which the individual has not been exposed. For example, the abstract schema **|subj *hit* obj|,** in which the specific linguistic form of the subject and object is unspecified, could be used in the production or recognition of an infinite number of situation referring expressions (subject to practical limitations) and Double R Grammar may be viewed as a Generative Semantic theory. Thus, schemas allow for the kind of productivity that is apparent in natural language. Under this scenario, abstract schemas are learned and are not innate, and it is not a question of having abstract schemas available innately and then specifying parameters for specific languages (Berwick & Weinberg, 1984; Chomsky, 1981, 1982). In an interesting article entitled *Language Acquisition: Schemas Replace Universal Grammar*, Arbib and Hill (1988) put forward a similar position. Further, it does not seem reasonable from a psycholinguistic perspective to suggest that once an abstract schema has been learned, the more specific schemas on which it was based become unavailable for use. Nonetheless, numerous researchers have made proposals which would have just this effect. For example, Hudson's (1984, p. 29) **Linguist's Economy Principle** states that a linguist should

> Never record any property more than once in relation to any entity. That is, once you (the linguist) have decided there is a generalization to be made, you make it in relation to the relevant general category…and then suppress any mention of it in more specific entries which can inherit the information from the general one.

Similarly, in a model of semantic memory, Collins and Quillian's (1969) **Cognitive Economy Principle** assumes that properties of objects are associated directly with the most general category to which they apply and only indirectly with the more specific categories. The psychological validity of this principle in models of semantic memory has already been challenged by Conrad (1972) and Collins and Loftus (1975). As well, the focus on generalization and universality in grammar has led to the creation of very general linguistic categories and to the relegation of idiosyncratic details to the lexicon. However, the more general the categories become, the greater the amount of idiosyncratic detail which must be relegated to the lexicon.

Arguing against such principles, Langacker (1987, p.28) introduces the **Exclusionary Fallacy**, noting that

> The gist of this fallacy is that one analysis, motivation, categorization, cause, function, or explanation for a linguistic phenomenon necessarily precludes another.

In Double R Grammar, the validity of Langacker's Exclusionary Fallacy is accepted and it is suggested that the above principles commit this fallacy in the name of efficiency and economy of representation. But they ignore the possibility of a trade-off between economy of representation and efficiency of processing. Redundancy in the encoding of information in the internal lexicon at different levels of abstraction is accepted as the norm and improves processing efficiency. This redundancy is functional in that given the associative, content addressable memory capabilities of humans, the encoding of frequently used values reduces considerably the amount of computation required on average—often replacing computation with simple memory retrieval.

X-Bar Theory

The meaning based definitions of specifier, head, modifier and complement used in Double R Grammar can be contrasted with the syntax based definitions of these terms in X-Bar Theory (Chomsky, 1970; Jackendoff, 1977). In X-Bar Theory there are typically three levels of syntactic representation: 1) the base or minimal level (X°), 2) an intermediate or non-minimal level (X'), and 3) a maximal level called the maximal projection of the head (X'' or XP). The head is the syntactic type (e.g. N, V, A, P) for which the variable X is a generalization. At each level in the representation, X corresponds to the same syntactic type and the head is said to project that type from X° thru X' to X''. Specifiers are defined configurationally with a description like "daughter of the maximal projection (XP) which is sister to a non-minimal head (X')." The configural definition of specifiers is widely accepted and specifiers are typically considered to be purely syntactic, making little or no contribution to meaning. I have even heard a transformational grammarian argue that the purely syntactic nature of specifiers provides evidence for the independent reality of syntax. Complements are defined configurationally as the linguistic elements which combine with a head (X°) to form a non-minimal head (X'). Modifiers (or adjuncts), to the extent that they are defined in X-Bar Theory, are linguistic elements which combine with heads without changing the level of the head. Viewed graphically (and ignoring modifiers), X-Bar Theory posits the following basic structure:

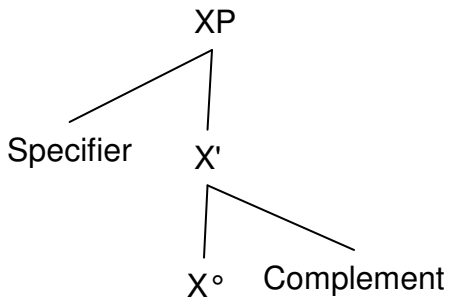

Figure 8: a common, simplified variant of X-Bar Theory

Double R Grammar accepts that X-Bar Theory captures an important grammatical generalization, but suggests that that grammatical generalization be given a semantic basis (Ball, 2003a). Doing so makes it possible to provide an underlying semantic motivation for X-Bar Theory and also offers the prospect of avoiding some of the pitfalls that a purely syntactic description can lead to. For example, the confusion over whether Tense (or Inflection) or V is the head of a predicate (i.e. VP or IP or TP) results from the assumption that the head must project the syntactic category of the predicate. But if this is true, then Tense must be the head of the predicate as is assumed by Chomsky (1995) since Tense is assumed to project the syntactic type of the predicate. On the other hand, V is the central semantic element of the predicate, with Tense filling a more peripheral semantic role. That is, V has a semantic prominence that does not hold for Tense. This leads Jackendoff (1977, 2002) to assume that V is the head of the predicate and not Tense. Similarly, for COMP. If COMP projects the syntactic type of a clause (i.e. CP or S), then COMP must be the head according to Chomsky (1995). But COMP is clearly a peripheral (and optional) element of a clause with V being the central semantic element. Recognizing that the role of a specifier and the role of a head are distinct, and equating syntactic type with referential type, it is easy to see that Tense projects the referential type of the predicate, whereas V projects the semantic (AKA relational) type of the predicate. Similarly, COMP projects the referential type of a complement clause whereas V projects the relational type of the clause. The optionality of COMP reflects the fact that clauses have a default referential type (i.e. situation referring expression), but that default can be overridden by a complementizer as in "That he likes you (is nice)" (i.e. objectified situation referring expression).

In a similar confusion, Abney's (1987) DP Hypothesis posits that determiners head determiner phrases (i.e. what are traditionally called noun phrases) with the determiner subcategorizing for a noun phrase complement. In the expression "the man", "the" heads a determiner phrase (DP) with "man" functioning as an NP complement. However, there is a basic problem with this hypothesis. The semantic type of object referred to by a DP is determined by the noun which heads the noun phrase complement (using Abney's terminology) and not the determiner. Thus, the complement must project the semantic type of the DP, not the head. However, the idea that a complement can project the semantic type of an expression runs counter to the whole notion of what a head is. The general direction in X-Bar Theory of treating functional elements like I (Inflection), T (Tense), D (Determiner), and C (Complementizer) as the heads of phrases like IP, TP, DP and CP leads McCawley (1989) to lament "…one of the things that annoys me about syntactic categories as they're treated in real recent MITish stuff is that it's really become hard for MITish people to say 'modifier' anymore. I mean, all sorts of

things that are to me obvious modifiers now get represented as heads of things they aren't heads of."

Abney (1991) resorts to the introduction of semantic heads or s-heads to distinguish content based heads from functional heads. "Intuitively, the s-head of a phrase is the most prominent word in the phrase" (ibid., p. 2). Abney's s-head comes close to the notion of head used in Double R Grammar—although his treatment of the noun head of the object of a preposition as the s-head of the prepositional phrase contrasts with the Double R Grammar treatment in which the preposition is the head. Distinguishing s-heads from functional heads improves on the DP Hypothesis, but still leaves open the problem of projecting the semantic type of the expression from the embedded s-head complement. Further, treating a word like "man" in the expression "the man" as an NP complement and maximal projection, distances the notion of maximal projection from referring expression and introduces a phrasal level of representation where none is needed except to meet the syntactic requirement that complements are maximal projections.

Cann (1999) puts forward a syntactically based dual-head proposal in which the specifier is treated as a secondary head with both the specifier and the head projecting features. Cann's syntactic treatment comes close to that of Double R Grammar, although his approach is not semantically motivated and he does not adopt a referential basis for distinguishing the role of the specifier from the semantic role of the head.

In the version of X-Bar Theory diagrammed in Figure 8, a single complement combines with the head and it is unclear how multiple complements should be treated. From a relational perspective this variant of X-Bar Theory is too restrictive. Relational elements may relate multiple arguments, with transitive relational elements including transitive verbs and prepositions being prototypical. But X-Bar Theory provides no mechanism for representing the relationship of the subject or first argument to the relational element and the specifier position has been co-opted in several variants of X-Bar Theory for this purpose. In fact, Cann (1999) suggests that Chomsky's main motivation for introducing the specifier was to provide an account for the subject. In Double R Grammar, the subject is a complement of the relational head of a clause, not a specifier. The subject says nothing about the referential status of the clause. That the subject happens to be in the typical position of a specifier in English is only true because the real predicate specifier (i.e. the auxiliary verb or tense marker on the main verb) is treated as a head and not a specifier. Further, treating the subject as a specifier still leaves open the question of how to handle multiple object complements. For example, in the expression "I bet you $50 we win" there are three complements "you", "$50", "we win" following the verb "bet" in addition to the subject complement. X-Bar Theory provides no mechanism for representing more than one object complement which does not violate some basic assumption of the theory (e.g. allowing additional complements to combine with non-minimal heads). Jackendoff (1977) gets around this problem by treating the term "specifier" as referring to everything to the left of the head and the term "complement" as referring to everything to the right of the head and according these terms no theoretical significance. And Chomsky in his original formulation of X-Bar Theory (1970) allows the term "complement" to refer to multiple complements. Unlike Jackendoff, Double R Grammar accords these terms theoretical significance, and despite Jackendoff's claim that Chomsky did not accord them theoretical significance, Chomsky's choice of the terms "specifier" and "complement" suggests otherwise as does Chomsky's suggestion that determiners (in NPs), auxiliaries (in VPs) and degree adverbials (in APs) function as specifiers (i.e. they specify NPs, VPs and APs, respectively). Not only does Double R Grammar accord

these terms theoretical significance, Double R Grammar attempts to provide a semantic motivation for their existence.

Finally, X-Bar Theory is an overgeneralization in that it suggests that all heads take complements. However, in Double R Grammar only relational heads take complements. The heads of object referring expressions do not take complements when that head is non-relational, as is typically the case.

The key advance made in X-Bar Theory is the distinction between specifiers and modifiers (or adjuncts). Chomsky (1970) realized that the specifier played a different syntactic role than adjuncts. The specifier combines with a non-maximal head to form a maximal projection whereas adjuncts combine with a non-maximal head without forming a maximal projection. Unfortunately, the purely syntactic basis of X-Bar Theory leaves this distinction unmotivated and the specifier has taken on a range of different (semantically unmotivated) syntactic functions in various versions of X-Bar Theory (e.g. the treatment of the subject as a specifier; the treatment of the specifier position as a landing site for movement). In Double R Grammar a maximal projection corresponds to a referring expression and it is the specifier that typically determines a referring expression. Modifiers do not perform this referential function and they do not combine with heads to form maximal projections (unless the head is already a maximal projection); rather, they perform a semantic function in constraining the semantic range of the head they modify.

In earlier linguistic treatments, specifiers and modifiers were typically lumped together as modifiers, adjuncts or attributes (e.g. Lyons, 1968; Hockett, 1958, and McCawley above) and the term specifier does not occur. For example, in discussing endocentric constructions, Lyons (1968, p. 233) states that "the constituent whose distribution is the same as that of the resultant construction is called the *head*; the other constituent is called the *modifier*." Current versions of dependency grammar (e.g. Hudson 1984) and (combinatorial) categorial grammar (e.g. Steedman 2000) continue to treat specifiers and modifiers alike as dependents of the head with which they combine, although Montague Grammar (Rick Cooper, personal communication) apparently treats the determiner as a head and not a modifier (pre-dating this use in current versions of X-Bar Theory).

Some variants of X-Bar Theory (and categorial grammar) adopt a strong version of the **binary branching hypothesis** which allows a non-terminal to expand into at most (and at least) two linguistic elements. The binary branching hypothesis is supported on primarily formal grounds which have no semantic basis—although a processing argument can be made—and its acceptance clouds the representation of the relationship between a relational element and its arguments when there is more than one argument. Double R Grammar allows more than one complement to be represented on a single level of representation, especially at the clausal level, and more directly represents this relationship.

The binary branching hypothesis is an example of a formal approach to syntactic analysis which is too rigid to be supportable. There are too many dimensions of meaning that need to be encoded for such a rigid hypothesis to be sustainable. The same holds true for X-Bar Theory more generally. The specifier in X-Bar Theory is the locus for the encoding of referential meaning. However, referential meaning may also be encoded via morphological marking (e.g. singular vs. plural number marking), or may be an inherent part of a lexical item (e.g. deictic words, proper nouns and pronouns). As such, a separate specifier need not always mark maximal projections (i.e. referring expressions), and the specification function may well be spread across multiple lexical items—both of which violate the basic X-Bar Schema. Viewed as a representation of the prototypical structure of a nominal (ignoring complements) or clause

(ignoring the lack of a subject complement), the schema is entirely appropriate. Put forward as an inviolable element of universal grammar, X-Bar Theory is fraught with contradictions and inconsistencies typical of the kinds of grammatical analyses it was intended to replace. As a final example, consider that the proliferation of functional heads, combined with the binary branching hypothesis, leads to a proliferation in the number of levels in X-Bar Theory based representations. But there is nothing in the X-Bar Schema that predicts the order of composition of these levels. That is, the head does not subcategorize for its complements. While it is implicitly assumed that V combines with NP (obj) to form VP, and I (or T) combines with VP to form IP (or TP), and C combines with IP to form CP, nothing in the X-Bar Schema requires this, and T could just as well combine with PP or AP to form TP, or V could combine with TP or CP to form VP. Adding in additional phrasal level categories (e.g., NegP, AgrS, AgrO) only exacerbates this problem. Of course, it is typically assumed that lexical items provide the subcategorization of complements needed to avoid this overgeneration of possible structures, but given such subcategorization frames for complements and adding similar frames to handle the function of specifiers, X-Bar Theory becomes largely redundant and the subcategorization schemas of individual lexical items become nearly all that is necessary (which may explain the elimination of X-Bar Theory in the Minimalist Program). Nonetheless, X-Bar theory still represents a useful generalization over those schemas, although it needs to be made consistent with the schemas it generalizes and it is unlikely to be universally applicable in any strong sense.

Langacker's Conceptual Schema for Nominals and Clauses.

Langacker (1991) provides a detailed Cognitive Linguistic description of the conceptual content of nominals and clauses which is closely aligned with the basic composition of referring expressions as described in Double R Grammar. Langacker puts forward the following schematization of the conceptual content of nominals and clauses:

(G(Q(I(T)))

where G = **grounding predication**, Q = **quantifying predication**, I = **instantiating predication**, and T = **type specification**. A grounding predication grounds an expression in the context of utterance of the expression, where that context includes the speaker and hearer and the immediate environment of the speaker and hearer. The most obvious grounding predication is a deictic word that refers to the speaker, the hearer or some other person in the immediate context (e.g. "I", "you", "he" or "she"). The determiner in a nominal expression and the first auxiliary (or modal) verb in a clause also function as grounding predications. In the case of a clause, the first auxiliary grounds the situation expressed by the clause into the context of utterance. A quantifying predication quantifies the number of discrete entities or events that are ground by the grounding predication. In a nominal, the most obvious quantifying predication is a number like the number "two" in "the two books". In the expression "some books", "some" functions as both a grounding and a quantifying predication. Note that "two" may combine with a separate grounding predication (e.g. the determiner "the") whereas, "some" does not. To distinguish these different uses of quantifiers, Langacker categorizes them into absolute quantifiers like "two" and relative quantifiers like "some". A relative quantifier is relative to some reference set. Thus, "some" represents a quantity relative to a reference set and grounds the quantity with respect to that reference set, whereas "two" is an absolute quantity independent of any reference set.

Adverbs like “everyday” and “repeatedly” often function as quantifying predications in clauses. An instantiating predication instantiates an instance that may be further quantified and ground in the context of utterance. According to Langacker (1991, p. 147), the head of a nominal (or clause) functions as the instantiating predication. Instantiation is different from grounding. Instantiation creates or identifies an instance of a type in the domain of instantiation, but does not necessarily ground that instance in the immediate context of the speaker and hearer. Finally, the word (or expression) that functions as the head of a nominal expression or clause provides a type specification which identifies the type of object or relation that the expression profiles. Thus, according to Langacker, the head of an expression minimally functions to provide both a basic type specification and to instantiate an instance of that type in the domain of instantiation (i.e. the space domain for nominals and the time domain for clauses). For nominals, this is true, whether the head is singular or plural. If the head is plural, an instance of a collective type—what Langacker calls a **replicate mass**—is instantiated.

Langacker uses the functional categories head, modifier and complement (but not specifier) in describing how grounding, quantifying, and instantiating predications, and type specifications compose together. Essentially, the head is a constituent which combines with a modifier such that the head provides the profile of the composite expression. A modifier, then, constrains the type specification of the head, but does not provide the profile for the composite expression. That is, the profile of the head projects to the composite expression, not the profile of the modifier. Absolute quantifiers function like modifiers in that the head they combine with provides the profile of the quantified expression. Langacker treats grounding predications special in that they not only combine with heads, but, unlike other modifiers, they profile the head they combine with. Note that it is the head that a grounding predication profiles, not the grounding predication itself. Further, it is the addition of a grounding predication that results in a full-fledged nominal. According to Langacker, “the two components [grounding predication and head] have equal claim to the status of local head, since both their profiles correspond to the composite-structure profile (that of the nominal as a whole).” (1991, p. 147-8). With respect to nominals grounded by the determiner “the”, Langacker states that “to the extent that *the* is regarded as the head, the other component—which elaborates the head—is a complement. To the extent that the elaborating structure is regarded as the head, *the* constitutes a modifier.” (1991, p.147). In the (G(Q(I(T)))) schema, the parentheses reflect the order of composition with the type specification first composing with the instantiating predication which then composes with the quantifying predication and finally the grounding predication. Thus, a grounding predication presupposes a quantifying predication which presupposes an instantiating predication which presupposes a type specification. Each level of composition reflects a modifier-head or head-complement relationship. Note that the order of composition is independent of the surface order of the constituents and the component elements may be morphological as well as syntactic.

In Double R Grammar there is a fourth functional category called the **specifier**. The grounding predication typically corresponds to a specifier with the specifier functioning as the “referential head” of a composite expression (the quotes around “referential head” indicate the non-standard use of the term “head” in this expression). The specifier or “referential head” combines with the “relational head” (encompassing non-relational objects) to form a composite expression, with the “relational head” providing the type specification for the composite expression and the “referential head” projecting the referential type of the composite expression. The introduction of the specifier function avoids the need to view the “relational head” as a complement as suggested by Langacker. It allows the head (as opposed to a complement) to

project the relational type—thereby, retaining a semantic basis for the notion of a head and at the same time maintaining a distinction between heads and complements (i.e. complements do not project relational type to composite expressions). It avoid the inconvenience of the suggestion that "the" is the head of the expression "the book"—contrary to any semantic notion of what a head is. (This same argument was used against Abney's DP Hypothesis and his introduction of s-heads as complements of a DP head in the previous section.)

There is a close correspondence between Langacker's grounding predication and the function of a specifier as the determinant of the referential type of an expression, and between Langacker's type specification and the function of a head as the determinant of the relational type of an expression. Further, Langacker's conception of modifiers as providing a higher-order type specification is entirely consistent with the function of modifiers in Double R Grammar. Less clear is the correspondence between Langacker's quantifying and instantiating predications and the functional categories of Double R Grammar. The fact that a quantifier may function as a specifier (e.g. "two" in "two books"), or as a modifier (e.g. "two" in "the last two books"), or even as the head of an expression (e.g. "two" in "I want two") argues against its treatment as a separate functional category. In this regard, a quantifier is more like a noun or a verb that can take on multiple functional roles, than it is a separate functional category and the treatment of quantifiers as a part of speech (where a part of speech reflects the inherent meaning of a lexical item) as opposed to a functional category is suggested. The quantifying predication may be encoded in multiple functional roles even within a single expression as in "these two books" where the specifier "these" indicates quantity as does the modifier "two" and the head "books". Similarly, the grounding predication appears to be encoded in multiple functional roles as in the nominal "two books" where the specifier "two" provides an (indefinite) grounding predication and the plural marker on the head noun "books" provides an additional (indefinite) grounding predication. However, Langacker argues that number marking on a head noun is part of the basic type specification with the head noun instantiating an instance of the basic type (i.e. a replicate mass when the noun is plural), and that number marking does not provide a separate quantifying predication (1991, p. 147). If Langacker's position is accepted, then quantifying predications and instantiating predications can be distinguished. Otherwise, assuming all nominal heads are marked for number and that number provides a quantifying predication, then nominal heads are quantified as well as instantiated and this distinction cannot be maintained. Double R Grammar assumes that the number marking on heads supports a quantifying predication and, for plurals, a grounding predication, as well—deviating from Langacker in this respect. The ungrammaticality of the expression "these book" reflects a conflict in the quantifying predication provided by the specifier "these" (plural) and the head "book" (singular) and supports the idea that single count nouns provide a quantifying predication. However, the failure of single count nouns to function as full nominals (e.g. "book" in "I like book") reflects their lack of a grounding predication. On the other hand, plural count nouns provide both a quantifying and a grounding predication and can function as full-fledged (indefinite) nominals (e.g. "books" in "I like books").

One way of integrating Langacker's account of conceptual composition with Double R Grammar is to treat grounding predications, quantifying predications and type specifications as conceptual features that supplement the semantic content of heads, modifiers, specifiers and complements. Note that these conceptual features are not the grammatical diacritics that Langacker argues against in syntactic approaches to linguistic representation. Rather, they represent additional semantic information that is provided by the word or expression functioning as a head, modifier, specifier or complement. For example, in the expression "the book" the

word “the” is functioning as a specifier which provides a grounding predication, whereas “book” is functioning as a head which provides a quantifying predication and a type specification. Note that “the” (unlike “a”) does not provide a quantifying predication since it is consistent with both “the book” and “the books”. Schematically, we can represent the functional form of the expression “the book” as

(Spec [G] (Head [Q,T]))

where [G] indicates that the specifier provides a grounding predication and [Q,T] indicates that the head includes a quantifying (i.e. singular) predication and a type specification. Using a tree diagram to represent this schema and the words that instantiate the functional roles gives

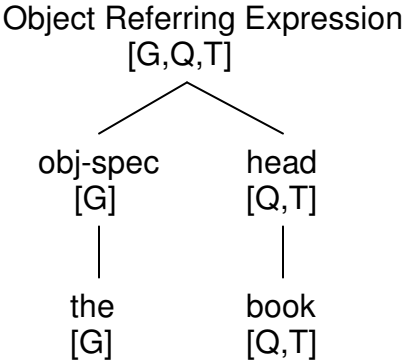

Figure 9: adding abstract conceptual features

Note that [G] and [Q,T] identify the conceptual roles of the specifier and head, but do not provide any details about those conceptual roles. For example, Q indicates that the word “book” provides a quantifying predication without saying what that predication is—namely, singular. Likewise, “the” provides a grounding predication—namely, definite grounding. If we substitute these more detailed descriptions into the tree diagram we have

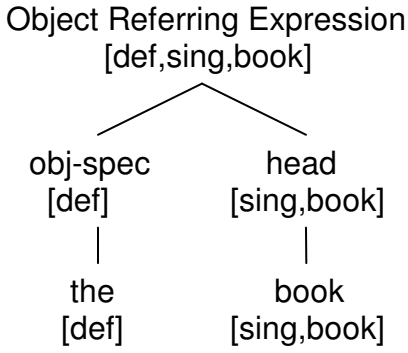

Figure 10: more specific conceptual features

where [def] indicates the definite grounding predication of "the", [sing] indicates the singular quantifying predication of "book" and [book] indicates the type specification. This representation is very close to that adopted in Double R Grammar when features are added.

In Double R Grammar, features are a means of providing additional semantic detail at a particular level of abstraction—in particular, at the level of abstraction represented by the functional categories head, modifier, specifier, and complement (and their subtypes). For example, the functional category specifier may be subcategorized as object specifier or predicate specifier and object specifier may be subcategorized as definite object specifier or indefinite object specifier. Alternatively, a feature notation may be used in which the more abstract specifier category is marked for the relevant semantic features as in

specifier [obj, definite]	vs. definite-object-specifier (e.g. "the")
specifier [obj, indefinite]	vs. indefinite-object-specifier (e.g. "a")
specifier [pred, finite]	vs. finite-predicate-specifier (e.g. "is")
specifier [pred, nonfinite]	vs. nonfinite-predicate-specifier (e.g. "to")

These are really just alternative representations of the same semantic content. That is, regardless of how the semantic content is represented, the word "the" provides information about the definiteness and "objectness" of the head it profiles and the word "to" (i.e. the infinitive marker) provides information about the non-finiteness and "predicateness" of the head it profiles.

The correlation between categories and features goes back at least to Chomsky (1970, p. 208) where he suggests the replacement of categories by sets of features, although he continues to use category labels for convenience. However, categories are more than just a convenience in Double R Grammar. They are the basis for the creation of schemas at multiple levels of abstraction. Further, there is no assumption that the feature set of a category is necessarily fully determinate, nor that the inheritance of features in a hierarchy of categories is absolute and indefeasible, nor that all features are of equal importance to a category.

The integration of Langacker's conceptual schema with Double R Grammar would be facilitated by the addition of the specifier function to his description. The addition of the specifier function makes it possible to provide more constrained and semantically motivated definitions of the traditional head, modifier and complement functions than is otherwise possible. The specifier is the locus for the encoding of referential information. Modifiers and heads are the locus for the encoding of information about the relational type of expressions. Complements are the locus for encoding information about the participants in relations. In Langacker's terms, the specifier supports the encoding of grounding and (optionally) quantifying predications. Heads and modifiers support the encoding of type specifications and, via number marking, quantifying and grounding predications. Quantifying predications are primarily referential and are typically expressed by quantifiers functioning as specifiers, but may also be expressed by quantifiers functioning as modifiers that constrain the relational type of the heads they modify. In addition to encoding referential information, specifiers profile the heads they specify. In addition to encoding semantic information, modifiers profile the heads they modify (or at least defer to the heads). Complements encode referential and relational information about the participants in relations, but that information is not profiled in the larger relational expressions in which they participate.

In discussing the grounding predication of clauses, Langacker argues that only the first auxiliary or modal verb provides the grounding predication and that all other auxiliaries form

part of the head. Further, the composition of these components proceeds from the main verb outwards. For example, in the expression “he could not have been kissed”, “kissed” first combines with “been” which combines with “have” which combines with “not” which combines with “could”. A similar position was adopted in an earlier version of Double R Grammar in which the first auxiliary or modal (also called the operator) filled the specifier role, with other auxiliaries functioning as modifiers of the main verb (note that Langacker treats the outer auxiliary as a head which combines with a verbal complement). However, there are reasons for modifying this position. Auxiliaries are members of a closed class verb subtype that look and behave very much like other specifiers (i.e. they are short, frequently occurring, and provide a referential function). Further, from a processing perspective, delaying the composition of complex auxiliaries until the main verb is processed, would strain the capacity of short-term working memory. In the processing of “he could not have been kissed”, if auxiliaries do not compose together until the main verb is encountered, five separate linguistic chunks (e.g. “he”, “could”, “not”, “have”, and “been”) would need to be retained in short-term working memory until the main verb “kissed” is processed. Allowing auxiliaries to compose together in forming a complex specifier “could not have been”, avoids the need to retain separate chunks in short-term working memory.

Double R Process: The Psycholinguistic Theory

Double R Process is a psychologically plausible processing mechanism for constructing integrated representations of referential and relational meaning. Double R Process is highly interactive. Meaning representations are constructed directly from input texts. There is no separate syntactic analysis that feeds a semantic interpretation component. The processing mechanism is driven by the input text in a largely bottom-up, lexically driven manner. There is no top-down assumption that a privileged linguistic constituent like the sentence will occur (vice Townsend & Bever, 2001).

In Double R Process there is no phrase structure grammar and no top-down control mechanism. How then does Double R Process construct representations of input text? Operating on the text from left to right, schemas corresponding to lexical items are activated. For those lexical items which are relational or referential, these schemas establish expectations which both determine the possible structures and drive the processing mechanism. A short-term working memory (Kintsch, 1998) is available for storing (or keeping active) arguments which have yet to be integrated into a relational structure, partially instantiated relational and referential structures, and completed structures. If a relational entity is encountered which expects to find an argument to its left in the input text then that argument is assumed to be available in short-term working memory. If the relational entity expects to find an argument to its right in the input text, then the relational entity is stored (or kept active) in short-term working memory as a partially completed structure and waits for the occurrence of the appropriate argument. When that argument is encountered it is instantiated into the relational structure in short-term working memory. In the case where the text contains a profiled or otherwise salient non-relational entity (e.g. the subject), that entity may be made separately available in short-term working memory. Otherwise, non-profiled and non-salient arguments are incorporated into relational structures and are not separately available in short-term working memory. This keeps the number of separate linguistic units which must be maintained in short-term working memory to a minimum. A key component of the processing mechanism is the activation and selection of schemas associated with the

referential or relational units in the input text. These schemas set up expectations which drive the processing mechanism and they also function as the key determiners of the structure of the input text.

It has been suggested that knowledge of language consists largely in the availability of schemas at multiple levels of abstraction, with more concrete schemas carrying most of the burden for language processing. But how might these schemas be organized, and how might they be accessed in language processing? Double R Process assumes that these schemas are organized in the form of an associative network over which a spreading activation process operates (e.g., Anderson, 1983, Collins & Loftus, 1975). As a piece of text is processed, schemas containing representations of lexical items which correspond to lexical items in the input text will be activated and will in turn activate associated schemas to some degree. Given this spreading activation mechanism, it follows that those schemas which most closely correspond to the input text will be most strongly activated. For the most part, these schemas will be concrete and lexically laden. Very abstract schemas which contain no lexical items can only be indirectly activated since they have no direct correspondence to the input text. Further, a schema may be activated despite the fact that it does not correspond exactly to the input text. This fact makes it possible for the system to deal with degraded or erroneous input, although in general the closer the correspondence between the input text and a schema, the higher the activation of that schema. In addition to the activation mechanism, there must be some selection mechanism for choosing among the activated schemas. In the simplest case this mechanism may simply select the most highly activated schema, and this selection process may be automatic. But selection of a particular schema should not preclude subsequent change in the context of new information and it may also be the case that more than one schema may be selected under certain circumstances (e.g., in the case of puns and double entendres). Thus, the selection process is both tentative (i.e., subject to revision) and preference based (Wilks, 1975).

Once selected, a schema must be integrated with the preceding linguistic context. The basic mechanism for integration is **elaboration** (Langacker, 1991). A relational schema typically provides only an abstract representation of its arguments. These abstract representations provide minimal information about the arguments. For example, the schema |**subj kick obj**| provides only limited information about the arguments of **kick**—i.e., that there is a subject argument that is an object referring expression which typically occurs before **kick** in the input stream, and that there is an object argument that is also an object referring expressions and which typically occurs after **kick** in the input stream. When the word "kicked" is processed, this schema is likely to be activated and selected. Its integration with the preceding linguistic context depends on the availability of an object referring expression capable of elaborating the subject argument (assuming the subject argument is integrated at this point).

Elaboration of the arguments of a relation is not the only possibility. Any component of a schema is a potential sight for elaboration, even a component which has already been elaborated. For example, if the input is the expression "the man kicked the bucket" then when the object "the bucket" is processed a specialized schema corresponding to the idiomatic expression "kicked the bucket" will be activated and will need to be integrated with the preceding context. Assuming the |**subj kick obj**| schema was activated by the processing of the lexical item "kicked", and the |**subj *kicked the bucket***| schema is subsequently activated, a mechanism for integrating these two schemas is needed. That mechanism will need to map the subject from the |**subj kick obj**| schema to the subject from the |**subj *kicked the bucket***| schema and map |…**kick obj**| from the |**subj kick obj**| schema to "kicked the bucket" from the |**subject *kicked the bucket***| schema. This

mapping needs to be automated since it does not take more time to process idiomatic expressions than non-idiomatic expressions (in fact the opposite is true). Elaboration which involves modification of an existing representation is called **accommodation**.

The notion of elaboration used in Double R Process follows (and extends) that of Langacker (1987, 1991). It is related to the more formal notion of **unification** which is explicitly used in several current linguistic theories (e.g. HPSG, Functional Unification Grammar, Tree Adjoining Grammar). However, elaboration, as used in Double R Process, differs from unification in several important respects. Elaboration is limited in terms of the recursive capability to elaborate substructures within structures. These limits follow from psychological limits on such capabilities in humans. Unification, as formally defined, has no such limits. Elaboration allows for the integration of partially matching structures, and may involve the integration of partially contrasting structures wherein the elaborating structure modifies (or replaces part of) the structure being elaborated (i.e. accommodation). Unification, as formally defined, does not support partial matching or accommodation. Elaboration implies directionality with a more concrete structure elaborating a more abstract structure. Unification implies no such directionality. The formal notion of unification, appropriately constrained to limit the recursive unification of substructures and extended to support accommodation, could be viewed as a mechanism of elaboration. Kempen and Vosse adopt a similar position in their Unification Space model of syntactic processing which assumes "a non-recursive form of feature unification." (Kempen, 1996, p. 215).

A consideration of the application of schemas to the processing of language offers several insights. If high-level abstract schemas drive the processing mechanism, then the processor can be said to be grammar driven. On the other hand, if low-level schemas with specific lexical items drive the processing mechanism, then the processing mechanism is essentially lexically driven. As noted above, it is an assumption of Double R Process, in agreement with Langacker (1987), that more specific schemas have "priority" over more abstract schemas in normal processing, and that most of the knowledge needed for the processing of familiar expressions has been lexicalized in the form of schemas containing specific lexical items. In unusual situations, abstract schemas may assume greater importance. For example, second language learners who are explicitly taught the grammar of a language may rely on more abstract schemas than native speakers of the language—not only as a result of instruction, but because they may lack the more specific schemas available to native speakers.

It is important to note that schemas set up expectations for the occurrence of various elements, but do not preclude the occurrence of other elements. That is, the application of a schema does not require the exact matching of the elements of the schema with elements of the input text. In this regard, schemas do not behave like the rules of a phrase structure grammar. For example, the |**subj verb obj**| schema sets up expectations for the occurrence of a subject, verb and object, but does not preclude the occurrence of an adverb or prepositional phrase in any input text to which the schema corresponds. Thus, the schema is entirely consistent with the sentence

He always eats fish on Friday

The occurrence of the adverb "always" and the prepositional phrase "on Friday" in this sentence activate additional schemas which must be integrated with the |**subj verb obj**| schema during the construction of a representation for this sentence. There is no need for a schema to specify all

possible elements which can occur at all possible positions in the schema so long as a mechanism for integrating multiple schemas exists. Double R Process provides such a mechanism. On the other hand, the existence of schemas which are completely specified with regard to their lexical content and are only weakly integrable with other schemas is not precluded (e.g. schemas corresponding to idiomatic and formulaic expressions). In general, abstract schemas will be highly integrable with other schemas, whereas, concrete schemas will be less integrable with other schemas.

The activation-selection-integration process which is the basis of Double R Process involves one automatic and two control processes: (a) an automatic spreading activation process, (b) a control process for selecting activated schemas from long-term memory and placing them (or making them active) in short-term working memory, and (c) a control process for integrating selected schemas in short-term working memory. The automatic process of spreading activation and the control process by which activated schemas are selected and placed (or made active) in short-term working memory will not be discussed further in this paper.

The control process of integrating selected schemas in short-term working memory can be described algorithmically in terms of the individual processing steps required to integrate the schemas which have been selected for further processing. This process will be illustrated by walking through the steps involved in the processing of the following English sentence:

The extremely old dog lover likes to sleep a lot

The processing of this sentence begins with the activation and selection of a schema corresponding to the first lexical item. The word "the" is identified and a referential schema which reflects its typical function as a specifier which combines with a head to form an object referring expression is selected:

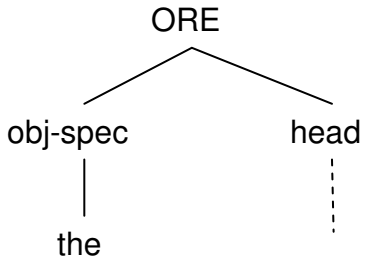

Figure 11: a specifier schema

In this representation, ORE stands for object referring expression, obj-spec stands for object specifier, and the dashed line extending from head indicates that this component has not yet been elaborated. This schema is equivalent to a schema of the form |**obj-spec head**| except that the type of the composite expression (i.e. object referring expression) is explicitly represented and the object specifier function is elaborated by the determiner "the". Part of speech information is left out of the schema since the focus is on identifying the functional role that the lexical item "the" takes on. Since the head is expected to occur after the specifier in this schema, the specifier must await the appearance of this head before combining with it to form an object referring expression. The specifier schema is retained in short-term working memory with its head unelaborated and the processing of this specifier is temporarily halted. The processing of the next

lexical item begins. The word “extremely” is identified and determined to be an adverb. In the context of an object specifier schema, “extremely” is presumed to be functioning as a relation modifier as represented by the following schema:

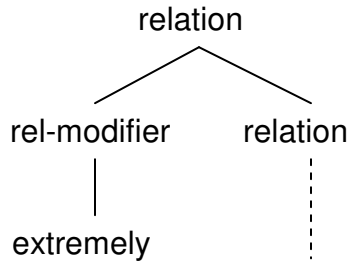

Figure 12: a relation modifier schema

According to this schema, the relation that “extremely” modifies typically occurs to its right in the input stream (at least in the context of an object specifier) and processing of the relation modifier is temporarily halted. The processing of the next lexical item begins. The word “old” is identified and determined to be an adjective. In the context of an object specifier and relation modifier, “old” is presumed to be functioning as an object modifier and the following schema is placed in short-term working memory:

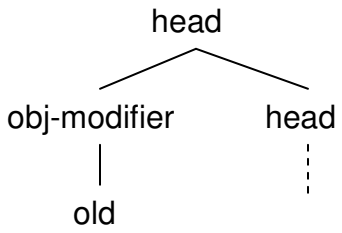

Figure 13: an object modifier schema

Note that there are three separate schemas in short-term working memory at this point. Since “old” is a relation, its schema can be integrated with the relation modifier schema giving:

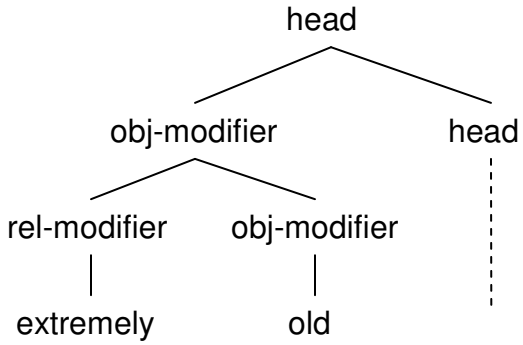

Figure 14: integrating a relation modifier with an object modifier

The head that occurs with “old” typically occurs to its right in the integrated schema and the processing of “old” is temporarily halted. Note that whereas the schema for “extremely” is integrated with the schema for “old” prior to the integration of a head into the “old” schema, this schema is not integrated with the specifier schema for “the” at this point—since integration with the specifier is presumed to require a fully elaborated head.

Processing continues with the next lexical item. The word “dog” is identified and determined to be a noun. In the context of an object specifier and object modifier schema, “dog” is determined to be a head that elaborates the head function in the schema for “old” giving:

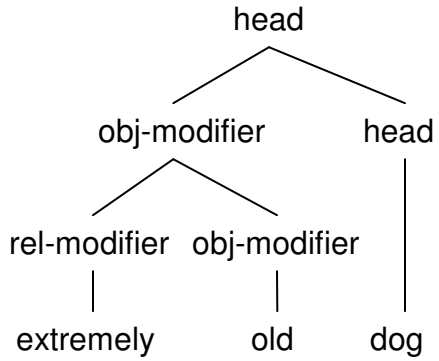

Figure 15: integrating a head with an object modifier

This modified head then elaborates the head function of the specifier schema of “the” giving an object referring expression:

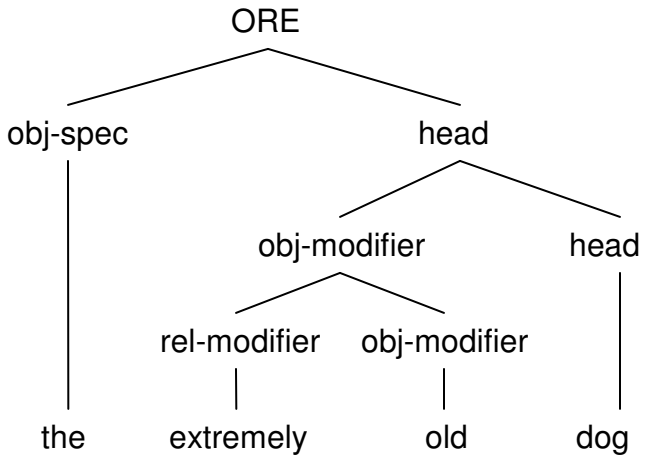

Figure 16: integrating a specifier with a head

At this point in processing, there is a single object referring expression which is a fully elaborated schema in short-term working memory. Object referring expressions are non-relational and do not typically establish expectations for other linguistic units. Processing continues with the next lexical item. The word “lover” is identified and determined to be a noun.

In the context of an object referring expression, “lover” is determined to be the head of that expression, displacing the current head and making it a modifier leading to:

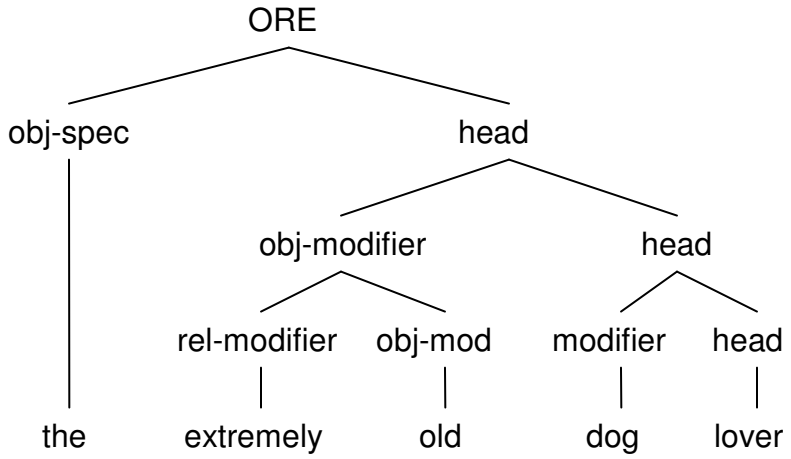

Figure 17: accommodating a second head noun

The processing of “lover” uses a special form of elaboration in which an existing schema is modified to accommodate the evolving context. The basic mechanism for processing “lover” in this context is to modify an existing schema to accommodate it. There is a (learned) procedure available to support such accommodation and it is not a question of backtracking and trying different alternatives until a correct structure is obtained. The major advantage of accommodation over backtracking is that the full current context is available to support accommodation, whereas in backtracking the context is typically unraveled and only the fact that the current choice is incorrect is typically available to the processing mechanism. That is, the context which forced the backtracking is not available to guide the selection of an appropriate structure.

Following the accommodation of “lover”, processing continues with the next lexical item. The word “likes” is identified and determined to be a present tense verb which determines a predicate referring expression functioning as a predicate in a situation referring expression with two participants, a subject and an object:

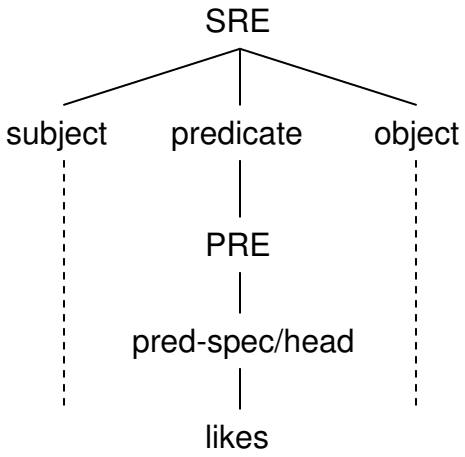

Figure 18: a relational head schema

where SRE is situation referring expression, PRE is predicate referring expression, and pred-spec is predicate specifier. In the context of an object referring expression, the object referring expression elaborates the subject of the situation referring expression giving:

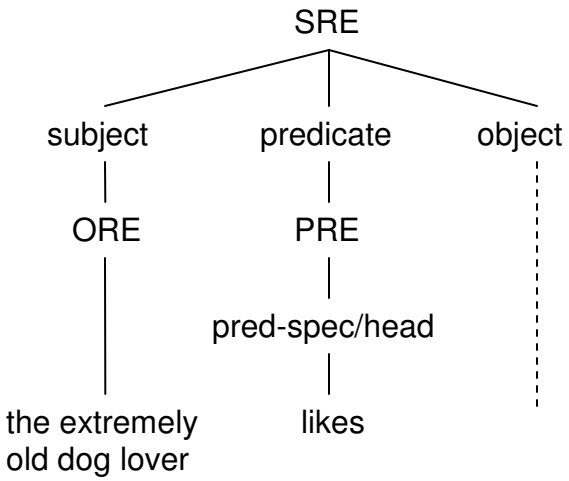

Figure 19: elaborating the subject argument

Note the assumption that the processing mechanism immediately elaborates the subject and does not wait until after the processing of the object to do so. The asymmetry of subjects and objects and the salience of the subject relative to the object is assumed not to be reflected in the processing mechanism in terms of the delayed elaboration of the subject relative to the object. Once the subject is elaborated, the processing of the schema for "likes" is temporarily halted and processing continues with the next lexical item. The highly ambiguous word "to" is identified and its function in the current context is not immediately determined. The subsequent context is needed to resolve the ambiguity. The next lexical item is processed and "sleep" is identified and determined to be a verb. In the context of "to", the non-finite, intransitive predicate referring expression "to sleep" is identified and its schema is retrieved and made available in short-term working memory:

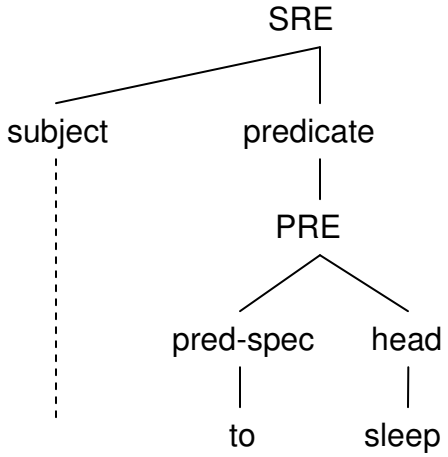

Figure 20: another relational head schema

This schema is integrated with the schema for "likes" elaborating the object argument, where this elaboration involves accommodating the clausal (i.e. infinitival) complement by converting the object (typically an object referring expression) to be a clausal complement. That conversion may involve the retrieval of a schema for "likes" which includes an infinitival complement and the integration of that schema with the existing schema for "likes", or it may more simply involve the conversion of the object argument of the existing schema to an infinitival complement:

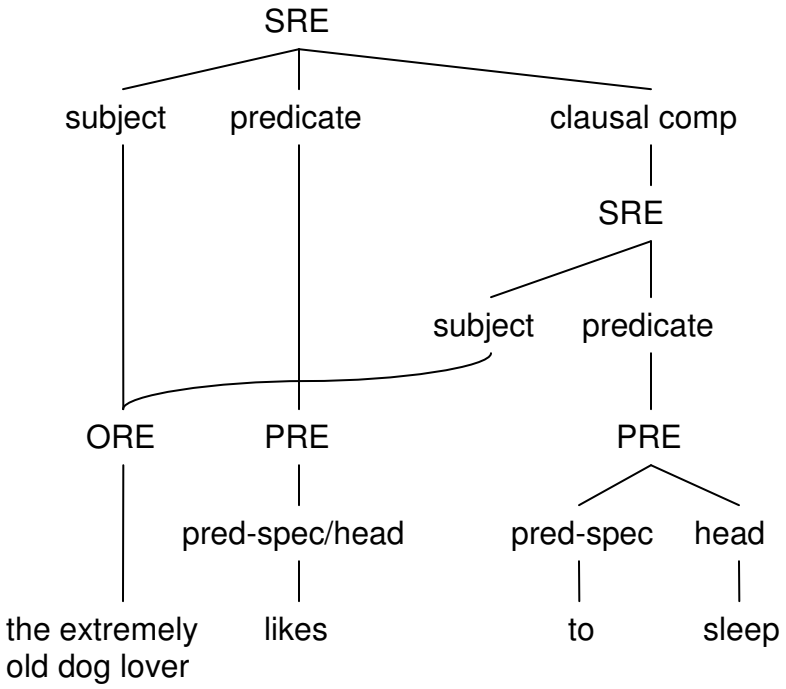

Figure 21: integrating two relational head schemas

Another part of the integration involves identifying the subject of "to sleep" with the subject of "likes" (a kind of mutual elaboration, or non-recursive unification). The result of this integration is a fully elaborated situation referring expression. Following the integration of the "likes" and "to sleep" schemas, processing continues with the next lexical item. The word "a" is identified and a specifier schema is retrieved. Then the word "lot" is identified and in the context of "a" and a situation referring expression, an idiomatic schema is retrieved.

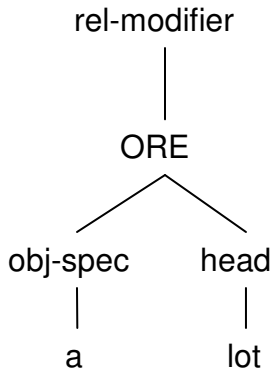

Figure 22: an idiomatic schema

This schema is idiomatic in that the object referring expression “a lot” is recognized as a expression which functions as a relation modifier despite it nominal form. This schema is integrated with the preceding context giving:

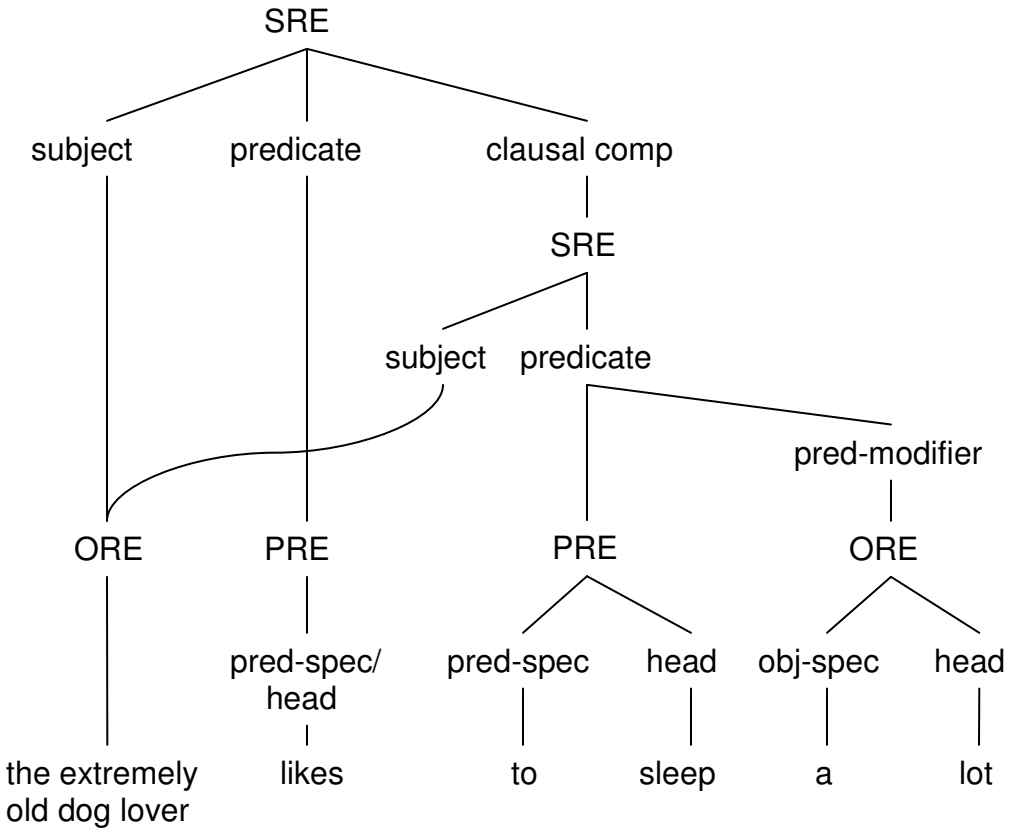

Figure 23: a fully elaborated situation referring expression

Note the decision to integrate the “a lot” schema with “to sleep” rather than “likes” in this example.

The processing of the input text is now complete and a representation of the referential and relational meaning of the expression—a situation referring expression describing a situation of "liking to sleep a lot" involving an "extremely old dog lover"—is available in short-term working memory for use in subsequent processing.

During the processing of this sentence, there were several decision points at which alternative processing decisions could have been made (e.g. integration of the subject with "likes" prior to processing the object, delaying the integration of a head noun until the lexical item following the noun is identified and determined not to be another noun). There is likely to be large variation across individuals (and even within individuals in different contexts) in terms of the selection of schemas and the timing of their integration. It may also be true that in contexts where one is only skimming a text, the integration process may be largely circumvented, resulting in only partially integrated representations. In general, it is assumed that humans learn **effective processing strategies** that tend to work efficiently given their goals and objectives, but that are subject to modification in particular contexts or when those goals and objectives change. For example, for verbs that take either an object or a clausal complement (e.g. "I believe him" vs. "I believe he likes you"), the frequency of occurrence of objects and clausal complements will contribute to determining which schema is initially preferred. However, frequency is not the only factor which influences schema selection. It may be more disruptive to the processing mechanism to have to recover from the inappropriate integration of an object referring expression as the object of a clausal complement verb, then to delay the elaboration of the object until it can be determined that the object referring expression is not the subject of a clausal complement. There is some evidence that experienced readers avoid such early commitments and are less likely to be "garden-pathed" when the subsequent context indicates the early commitment was incorrect. True garden-pathing, where readers are forced to backup and re-read a piece of text, is assumed to be uncommon and typically involves restarting from (near) the beginning of the text (carrying along contextual information) rather than any kind of formal backtracking. More usually, representations are modified to accommodate inputs which are inconsistent with the prior context rather than being deconstructed and rebuilt.

The preceding example considered the processing of a sample sentence into a representation of its referential and relational meaning. In conjunction with the creation of a representation of the text, a situation model (Kintsch 1998) is also constructed. The creation of the situation model corresponding to a text is driven by the grounding of the referential elements of the text being processed. That is, whenever a referring expression is recognized, the situation model is updated to reflect the occurrence of that referring expression. Thus, referring expressions are crucial to the processing mechanism as well as the representational system. As noted above, the construction of situation models is an active area of research (Ball, 2003b). Kintsch's Construction-Integration Theory (discussed below) is highly compatible with Double R Process and provides the theoretical framework for this important (but not yet fully elaborated) component of Double R Process.

Construction-Integration Theory

Construction-Integration (C-I) Theory is a discourse level theory of (language) comprehension which begins with the construction of the propositions in a chunk of discourse and continues with their integration into a coherent representation of meaning. The construction of the propositions in a chunk of discourse is based on guidelines which have never been fully

implemented in a working system (of any substantial size) and C-I Theory provides only a descriptive account of this process. The representation of the propositions in a chunk of discourse is called the **textbase**. As part of the construction process, the textbase is elaborated with knowledge from long-term memory into a **situation model**. An unconstrained associative activation process as well as an inferencing process support the elaboration of a textbase into a situation model. The situation model is a network of propositions, some of which originate in the processed text and some of which result from the elaboration process. The situation model is the output of the construction process.

The integration process involves the settling out of the activation of the propositions in the situation model in what Kintsch calls a constraint-satisfaction process. Since there are inhibitory as well as excitatory associations in the network of propositions, some propositions will be deactivated during the integration process and some propositions will become more active. At some point, a stable state will be attained in which the change in activations of the propositions in the network falls below a pre-determined criteria and the integration process terminates. The resulting network of activated propositions—some from the text, and some from long-term memory—represents the meaning of the discourse.

C-I Theory is fundamentally a discourse level theory. The construction process involves the creation of an activated network of related propositions followed by an integration process operating over the activated network of propositions. To extend C-I Theory down to the sentence level (and to integrate it with Double R Process), the construction-integration process would need to occur after the processing of each lexical item in a sentence (Kintsch, 1998, p. 101). At this level, the recognition of a lexical item would lead to the unconstrained activation of associated schemas in long-term memory (the construction process) followed by integration of the activated schemas with the prior context leading to the activation of the contextually most salient schema and the deactivation of non-salient schemas (the integration process). The recognition of the next lexical item would result in a similar process. It may be that full integration (i.e. full settling or constraint-satisfaction) does not occur until the end of the processing of a sentence (or text), but some initial integration is needed to support schema selection and integration prior to the end of the sentence as each lexical item is processed. In C-I Theory both the construction and integration processes are largely automated. That is, there is no overt control mechanism making decisions about how to combine activated schemas, just the unconstrained association (construction) and constraint-satisfaction (integration) mechanisms. In fact, Kintsch (1998, p. 118) notes the replacement of an earlier schema-based control system with a bottom-up constraint-satisfaction mechanism at the time the C-I architecture was first introduced in 1988. On the other hand, given the largely automated nature of the construction process, Kintsch's use of the term "construction" to describe this process is interesting (and perhaps confusing). It at least suggests a control mechanism for creating constructions. But according to Kintsch (1998, p. 94) the "…construction process is only weakly controlled and proceeds largely in an associative, bottom-up manner…." At the level of the processing of lexical items, Kintsch's use of the term construction follows from the claim that "word meanings are not something to be pulled from a mental lexicon but are ephemeral constructions that are generated during comprehension" (1998, p. 136). Thus, although the construction process is largely automated, the result of the process is something that did not exist previously (i.e. a contextually activated network representing the meaning of a lexical item), and it is in this sense that the process is constructive.

In Double R Process, the initial schema activation process is automated, not controlled, and it corresponds quite well to C-I's construction process. However, the selection and integration processes of Double R Process are controlled and differ from C-I's integration process in this respect—particularly the integration process of Double R Process which is highly controlled (although proceduralized). To the extent that there are control mechanisms in C-I Theory, they occur prior to the automated integration mechanism, whereas in Double R Process integration is the locus of controlled processing. Further, there is no automated settling out process involving inhibitory associations in Double R Process. In essence, Double R Process retains the schema-based control process that C-I Theory forsook in favor of an automated constraint-satisfaction process. However, the schemas used in Double R Process are activated by lexical items in a bottom-up manner and only create top-down expectations for the arguments of relational lexical items (and for heads in the case of specifiers and modifiers). Thus, they are quite unlike the much more elaborate schemas (e.g. restaurant script, grocery-shopping script) that Kintsch was at pains to eliminate from the integration process (but which are retained in the construction process to support inferencing).

In Kintsch's most recent work (Kintsch, 2001), he extends the argument against the existence of pre-fabricated word senses in the mental lexicon citing examples like "the horse runs" and "the paint runs" in arguing that the meaning of "run" is contextually determined and not pre-stored as distinct word senses. The elimination of distinct word senses in C-I Theory is a laudable objective. It makes conceivable the elimination of purely conceptual entities (i.e. entities not tied to perception) more generally (Barsalou, 1999)—although Kintsch does not appear to reach this conclusion himself. Kintsch makes use of Latent Semantic Analysis (LSA) (Landauer & Dumais, 1997) in explaining how the context of use of a word can be used to derive its meaning. "LSA is a mathematical technique that generates a high-dimensional semantic space from the analysis of a large corpus of written text" (Kintsch, 2001, p. 176). In LSA, words and texts are represented as vectors in that high-dimensional semantic space. The cosines between the vectors of individual words and/or texts can be computed and used as a measure of relatedness of those words and texts. Kintsch uses LSA as an objective means of determining the initial degree of association of the words in a text and of associated words which are activated as part of the construction process.

LSA has several advantages over more traditional co-occurrence matrices for determining the relatedness of words or texts. "LSA used a standard mathematical technique called singular value decomposition, which allows it to select the most important dimensions underlying the original co-occurrence matrix, discarding the rest." (ibid. p. 176). Using LSA, it is quite common to take a co-occurrence matrix containing 1000's of dimensions (1 dimension for each word or text) and to reduce that matrix to about 300 dimensions. The basic psychological claim is that the reduced matrix is actually a better representation of the relatedness of words and texts than the original co-occurrence matrix because singular value decomposition eliminates distracting information resulting from the accidental co-occurrence of words and texts and adds information resulting from indirect associations which are not captured by mere co-occurrence (e.g. words that are substitutes for each other, appearing in similar contexts but not appearing together).

Despite LSA's advantages over co-occurrence matrices as a representation of "semantic space", LSA is not adequate, in and of itself, as a full representation of meaning. Word order is not taken into account within the analyzed texts and the texts "the man bit the dog" and "the dog bit the man" would have essentially the same semantic vector. Also, antonyms like "above" and "below" lead to closely related semantic vectors which make it difficult to distinguish their

semantic contrast. And function words like "the" and "of" which occur in nearly every text cannot be distinguished. For a full representation of meaning, the meaningful effects of word order and grammatical function need to be considered, and something like a propositional representation which reflects these influences is needed. Thus, whereas LSA may prove adequate for determining the meaning of lexical items in context (without the need for distinct word senses), some mechanism for representing the relationships between lexical items is still needed.

Returning to the propositional level, it was noted above that the situation model that is the output of the construction process and the input to the integration process is a network of propositions. According to Kintsch (1998, p. 37) "the predicate-argument schema can be regarded as a basic unit of language…the predicate determines the number and kinds of arguments that may fill argument slots, that is, the semantic role of the participants." Kintsch provides significant empirical and theoretical motivation for his selection and use of the predicate-argument schema in representing the meaning of discourse. However, Hawkins (1984) analyzed the validity of the function-argument and head-modifier relationships from the perspective of linguistic universals and concluded that the head-modifier relationship is more universal. PM (Ball, 1991), the predecessor to Double R Grammar, agreed with Kintsch in accepting the predicate-argument relationship as basic. Double R Grammar accepts that both the predicate-argument relationship and head-modifier relationship are important, arguing that the predicate-argument relationship is fundamentally about the encoding of relational meaning, whereas the head-modifier relationship is fundamentally about the encoding of relational types (including objects). In addition, Double R Grammar distinguishes the specifier-head relationship from the head-modifier relationship and argues that the specifier-head relationship is the locus for the encoding of referential meaning.

According to Kintsch,

> the meaning of a simple sentence can be represented by a *complex proposition*…consisting of a predicate with several arguments, time and place circumstances and optional modifiers. *Predicates* are relational terms, frequently expressed in language as verbs, adjectives, or adverbs. Each predicate is characterized by a predicate *frame* that specifies how many and which *arguments* that predicate takes. A verb frame, for instance, may specify that a particular verb must have an agent, an object, and an optional goal, with restrictions on these categories, such that the agent must be human, or the goal must be a location. Not only can concepts be arguments, but a proposition may have as an argument another embedded atomic proposition. Sentence connectives, for instance, are predicates that require propositions as arguments. *Circumstances* refer to the whole propositional frame, specifying place and time. All propositional elements may be *modified* by additional atomic propositions. (1998, p. 54-55).

The general schema for complex propositions is given by

Category (action, event or state):
Predicate:
Arguments (agent, object, source, goal,…);
Modifiers
Circumstance:

Time:
Place:
(ibid., p.38).

Using this schema, Kintsch represents the sentence "Yesterday, Mary gave Fred the old book in the library" as

Action:
Predicate: GIVE
Arguments:
Agent: MARY
Object: BOOK
Modifier: OLD
Goal: FRED
Circumstance:
Time: YESTERDAY
Place: LIBRARY
(ibid., p.38)

Kintsch acknowledges that these propositional representations do not capture the full complexity of meaning expressed in language, but argues that they capture those aspects of meaning which are relevant for various theoretical and experimental purposes. "The choice of which sentence properties to represent in the propositional notation is pragmatic. Whatever seems of little importance for a given theoretical or experimental purpose is omitted." (1998, p. 39).

Using an even more basic schema

predicate[argument,argument,…]

Kintsch provides the following propositional representation for the sentence "the blood returns to the heart quickly"

QUICK[RETURN[BLOOD,HEART]]

The capitalized words in these representations are intended to represent unambiguous concepts, not ambiguous lexical items. Specifiers like "the" and the tense of the verb are simply ignored. In this more basic representation, the modifying relationship between "quickly" and "return" is represented in the same manner as the predicate-argument relationship between "return" and "blood" with no indication of which element is the head in each case (i.e. "return" in both cases). Note that "quickly" and "quick" apparently map to the same concept QUICK. In the basic representation, the role of the arguments (e.g. agent, patient) is not explicitly represented, nor are any semantic feature restrictions (e.g. human) on the concept functioning as the agent.

By comparison, the representations in Double R Grammar are explicitly linguistic. They do not abstract away from the linguistic context. In particular, the meaning of specifiers is explicitly represented in terms of their referential function and tense and aspect are not ignored. Double R Grammar based representations contain actual and ambiguous lexical items which are disambiguated via the associations they invoke with other lexical items and in terms of the

objects and relations to which they refer. In this respect, Double R Grammar extends Kintsch's arguments against the existence of static word senses (Kintsch 2001) to the existence of abstract concepts and propositions as well. There must be a perceptual basis, whether linguistic or not, for all concepts and propositions. If that perceptual basis is linguistic, then it is explicitly linguistic, not conceptual.

In C-I Theory, the basic mechanism for showing the relationships between propositions in a propositional network is via conceptual argument overlap. Since the arguments are represented by unambiguous concepts, determining argument overlap simply involves a comparison for concept equality. Given the linguistic basis of representations in Double R Grammar, argument descriptions are unlikely to match and the mechanism of argument overlap is not available. For example, arguments are typically first introduced with an indefinite description and subsequently referred to by definite descriptions which may consist of no more than a pronoun. In place of argument overlap, it is the determination that two object referring expressions (or situation referring expressions) refer to the same object (or situation) that establishes the association.

It is not claimed that Double R Grammar has a completely worked out theory of reference, anaphora resolution, lexical disambiguation, structural disambiguation, etc. However, in not abstracting away from the linguistic input, Double R Grammar offers the prospect for extension to handle these aspects of meaning more completely. C-I Theory can likewise be extended to handle more subtle aspects of linguistic meaning, but doing so will require less and less abstraction from the linguistic input. The big first step for C-I Theory is to do away with the abstract concepts and propositions that have served it so well to date. Kintsch has already gone some ways in this direction in advocating the abolition of distinct word senses.

The Atomic Components of Thought – Rational

The Atomic Components of Thought – Rational (ACT-R) cognitive architecture is a theory of cognition based on decades of psychological research (Anderson, 1976, 1983, 1993; Anderson and LeBiere, 1998; Anderson et al., 2002). Since 1993, a computer simulation of the ACT-R architecture (developed by Anderson and his colleagues) has been made available to the research community to support the development of computational cognitive models that explore the theory. The ACT-R simulation is referred to herein as the ACT-R cognitive modeling environment. The ACT-R cognitive architecture and modeling environment is being used by researchers around the globe to develop and test cognitive models covering a wide range of behaviors. The ACT-R architecture includes a production system integrated with a declarative memory (DM) system. The distinction between procedural and declarative memory is a cornerstone of ACT-R and is supported by extensive empirical evidence. ACT-R is a hybrid architecture which provides symbolic productions and declarative memory chunks and subsymbolic mechanisms for production selection and declarative memory chunk retrieval. Production selection and declarative memory chunk retrieval are implemented as highly parallel processes, however, once selected, production execution is serial—only one production can be executed at a time—and only one declarative memory chunk is retrieved. These symbolic and subsymbolic components and mechanisms (as implemented in the cognitive modeling environment) provide the cognitive infrastructure needed to model human behavior at a low enough grain size to model the time course of cognition at the millisecond level. This makes it possible to model real-time human performance in the ACT-R cognitive modeling environment.

ACT-R Version 5 extends the ACT-R architecture, adding a perceptual-motor system for interacting with the external world which has been incorporated into the cognitive modeling environment. The perceptual-motor system of ACT-R 5.0 makes it possible to develop embodied models of cognition. With the addition of the perceptual-motor system, ACT-R has evolved into a theory of the mind that consists of multiple modules and is concerned with explaining how these modules are integrated to produce behavior. Thus, ACT-R has transitioned from a "unified theory of the mind" focused on higher level cognition to an "integrated theory of the mind" that encompasses perception and motor action in addition to higher level cognition.

The production system is the heart of the ACT-R architecture and is responsible for coordinating the activities of the other modules. The basic cycle of cognition in ACT-R involves the selection and execution of a production. This cycle is assumed to take at least 50 msec to complete. A production is a learned condition-action rule. Production selection involves the matching of the conditions of all productions in procedural memory in parallel and selection of a single production for execution. The execution of a production completes the action side of the condition-action rule. The production system interfaces with the other modules via a collection of buffers. It is the information in these buffers which is matched against the condition side of each production during the production selection process.

The ACT-R architecture in depicted in Figure 24 below

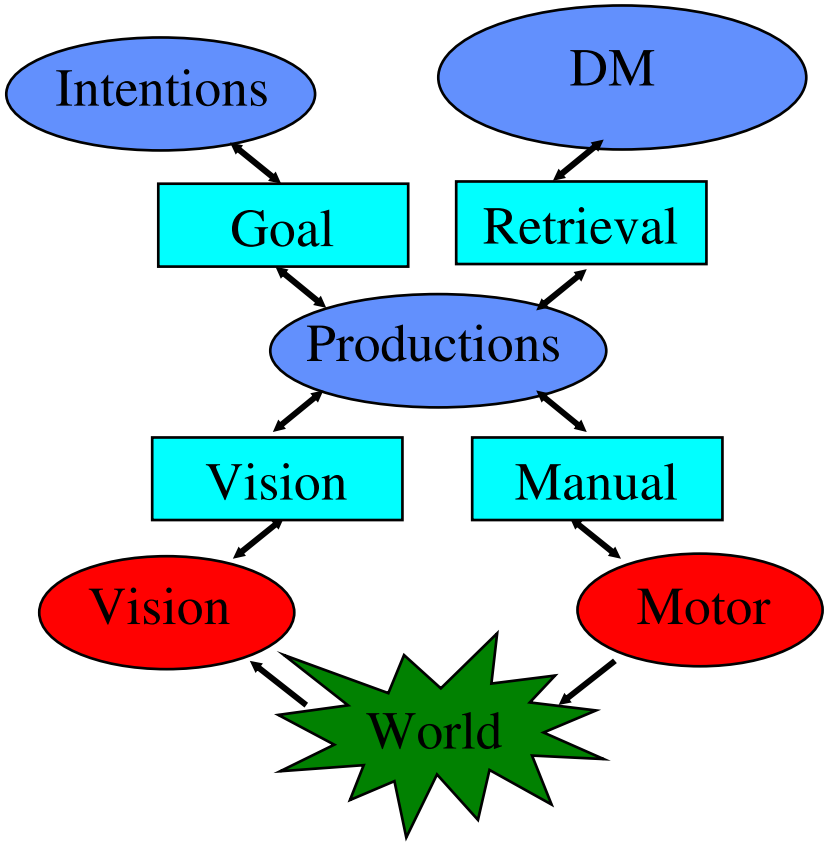

Figure 24: the ACT-R Architecture

The production system interfaces with the intentional system via the goal buffer, with the declarative memory system via the retrieval buffer, with the vision system via the vision buffer

(actually a visual object buffer and a visual location buffer) and with the motor system via the manual buffer. The intentional and declarative memory systems are internal cognitive systems (along with the production system), whereas the vision and motor systems interface with the external world.

At any point in time, the information stored in the buffers determines which productions are eligible for selection and execution. Production execution may in turn cause the information in the buffers to change via creation of a new goal, retrieval of a new declarative memory chunk, shifting attention to change the contents of the visual buffer, or effecting a motor action.

There are several key equations that determine the subsymbolic behavior of ACT-R. The two most important equations are the **activation equation** and the **production utility equation**. The activation equation establishes the activation of declarative memory chunks and determines which declarative memory chunk will be retrieved from memory and placed in the retrieval buffer:

$$A_i = B_i + \sum_j W_j S_{ji} + \sum_k P_k M_{ki} + \sigma$$

A_i = total activation of declarative chunk i

B_i = base level activation of a declarative memory chunk independent of activation from the goal chunk based on the history of use of the chunk

$\sum_j W_j S_{ji}$ = activation of declarative memory chunks by the current goal chunk based on the number of slots S_{ji} in the goal chunk which match the declarative memory chunk and on the weight W_j accorded each slot.

$\sum P_k M_{ki}$ = activation of declarative memory chunks by the retrieval template where P_k is the importance of a slot in the retrieval template and M_{ki} is the degree of match of the slot in the retrieval template to a slot in the declarative memory chunk.

σ = noise which adds stochasticity to the activation equation

The production utility equation determines the likelihood that the selection and execution of a procedure will lead to the achievement of the current goal. Like the declarative memory retrieval process, the production selection process occurs in parallel across all productions and the production with the highest utility is selected and executed:

$$U_i = P_i G - C_i + \sigma$$

U_i = utility of i^{th} production

G = value of goal the production is meant to facilitate

P_i = probability of success of goal given the execution of production i

C_i = cost of using production i to achieve goal

σ = noise which adds stochasticity to the production utility equation

ACT-R provides a collection of parameters for setting the values of the variables in these (and other) equations, but many of the parameters come with default values that are well established and typically accepted.

ACT-R is a general cognitive architecture that was not specifically designed to support language comprehension. However, to the extent that language comprehension uses general cognitive and perceptual capabilities and is not a distinct, separate module of the mind, ACT-R is a reasonable architecture for modeling language comprehension systems. More specifically, to the extent that language comprehension makes use of the production system, the declarative memory system, and the intentional system for the higher-level processing of language, and to the visual (and/or aural) system for lower-level perception of language, ACT-R is an appropriate architecture. Double R Theory assumes that language comprehension does not involve a separate and distinct language (or syntax) module, and that ACT-R is in fact an appropriate architecture.

Of course, language comprehension systems must actually be implemented in the ACT-R cognitive modeling environment that simulates the ACT-R architecture, and the specific commitments of the cognitive modeling environment (as opposed to the architecture) impose additional constraints. For example, it is unclear if the highly parallel declarative memory activation and production selection processes can be efficiently implemented on a serial computer, and efficiency concerns may have impacted the implementation of these processes in the cognitive modeling environment. The spreading activation mechanism, as reflected in the activation equation of the ACT-R architecture, could have been implemented as in the C-I Theory as a settling process involving all the chunks in declarative memory. However, Kintsch (1998) notes that the amount of time required for a matrix of activations to settle is indeterminate. Given the commitment of the ACT-R cognitive modeling environment to be able to model the time course of behavior at the millisecond level, implementation of a settling mechanism over all of declarative memory was not feasible. In fact, C-I Theory cannot be used to model the time course of comprehension (in real-time) for precisely this reason. Instead of implementing a settling mechanism operating over declarative memory, the spreading activation mechanism of ACT-R is limited to spread from the slots in the goal chunk in the goal buffer to matching declarative memory chunks. Some empirical support has been provided for this implementation decision. For example, Anderson (1993?) notes the lack of priming effects between indirectly related words like “bull” and “milk” (indirectly associated via “cow”) in arguing for the approach to spreading activation adopted in ACT-R 5.0. However, the single level spread of activation makes it difficult to model even basic priming effects (e.g. Meyer & Schvaneveldt, 1971) in the ACT-R cognitive modeling environment since all associates must be directly associated via their slot values and for other than extremely restricted data sets this leads to an unreasonable proliferation of the number of slots in the associated declarative memory chunks (e.g. think of all the possible associates of “cow”). The result is a strong a priori constraint on the possible forms of representation. Any two declarative memory chunks that are associated must share slot values in order to spread activation. The ACT* architecture that preceded ACT-R (Anderson, 1983) supported multiple level spread of activation and did not impose this constraint.

The use of the goal chunk as the locus for spreading activation is only one of the functions of this overloaded element of the ACT-R cognitive modeling environment. In addition to functioning as the source for spreading activation and representing the intentional component of

ACT-R, the goal buffer is the locus for the creation of abstract declarative memory chunks and is one of the key determinants of the production selection process. Objects attended to in the visual buffer of ACT-R become perceptual chunks in declarative memory and these perceptual chunks can subsequently be retrieved and placed in the retrieval buffer. But the mechanism for creating abstract, conceptual chunks is the creation of a new goal. For example, whereas the visual buffer is the locus for the creation of perceptual chunks corresponding to words as visual objects, the creation of a word chunk which abstracts from the perceptual form of the word (i.e. particular font, size, location, orientation, etc.) requires creation of a new goal to represent that abstraction. But there is a basic conflict between the function of the goal for production selection and its function to create declarative memory chunks. It is common in ACT-R based cognitive models to include state information in the goal chunk to guide production selection. For example, use of a "state" slot in the goal chunk to encode the current state (or context) is quite common. However, such state information is not appropriate for the creation of abstract declarative memory chunks which are abstracted from the particular contexts of use. Each abstract declarative memory chunk created will retain this state information and there is no mechanism for the state information to decay more rapidly than the relevant information. Further, since the type of a declarative memory chunk must be declared when the cognitive model is created, the declaration of the chunk type will include slot declarations for the irrelevant state information (e.g. a procedural "state" slot in a declarative memory chunk of type "noun").

In Double R Theory, abstract categories and schemas are learned from experience of specific instances of those categories and instantiations of those schemas. The abstraction mechanism may or may not be goal directed. Even if goal directed, it probably does not involve the instantiation of the abstraction as the goal itself. If the intent (or goal) is to create an abstraction, it is hard to see how that abstraction can be (or become) the goal.

As a production system, the ACT-R cognitive modeling environment provides a mechanism for the selection and execution of productions in a forward chaining manner. No mechanism for backward chaining or backtracking is provided. The ACT-R cognitive modeling environment provides limited pattern matching capabilities primarily limited to determining that the values of the slots in two different declarative memory chunks are equal or different or to matching a value in the slot of one declarative memory chunk to a variable slot in another declarative memory chunk. The ACT-R cognitive modeling environment does not support the matching of two variable slots together and does not support the matching of substructures in slots, matching only on the names of such structures. Thus, the ACT-R cognitive modeling environment has only an extremely limited unification capability. The ACT-R cognitive modeling environment provides support for single inheritance, allowing declarative memory chunks to be organized into a type hierarchy, but does not support multiple inheritance. Earlier versions of ACT-R had a goal stack that is replaced by a single goal buffer in ACT-R Version 5. The goal stack provided a mechanism for controlling behavior by supporting the pushing of the current goal and introduction of a subgoal and the subsequent popping of the subgoal and reinstatement of the previous goal.

Distinguishing limitations of the cognitive modeling environment from the underlying architectural theory is somewhat problematic, although in some cases the distinction is clear. The elimination of the goal stack is theoretically motivated. In removing the stack, it was argued that the goal stack was too powerful in certain respects (e.g. its unbounded depth, its perfect memory) and that goals should be stored in declarative memory and not in a separate memory stack. (Anderson et al., 2002; Altmann and Trafton, 2002). On the other hand, the lack of multiple

inheritance is clearly a limitation of the cognitive modeling environment which is not based on any theoretical considerations that I am aware of. The limited pattern matching capabilities of the cognitive modeling environment probably reflect its implementation in Lisp as opposed to Prolog, although Prolog's full unification capability is psychologically too powerful. The lack of backtracking is another reflection of the underlying implementation language, but psychological limits on backtracking are well attested in any case. Overall, the developers of the ACT-R cognitive modeling environment have been very successful in not allowing that environment to negatively impact the underlying architectural theory which is firmly based on psychological research and not tied to implementation decisions. And despite the limitations of the ACT-R cognitive modeling environment, it is the best available environment for the development of computational cognitive models and the ACT-R architecture is the best available theory of human cognition.

Double R Model: The Computational Psycholinguistic Implementation

In the mid-eighties when this research started, Lisp and Prolog were the dominant programming languages for the implementation of language processing systems within the Computational Linguistics and Artificial Intelligence community of which I was a member. PM, the predecessor of Double R Model, was implemented in Prolog over the course of six years prior to the end of 1991 (Ball, 1991). PM relied extensively on Prolog's unification and backtracking capabilities. While Lisp and Prolog offer powerful capabilities to support the development of language processing systems, consideration of psychological constraints on language processing was generally not a concern, and these languages were too powerful in certain respects. The emergence of Soar in the late eighties/early nineties and the more recent emergence and availability of ACT-R—cognitive modeling environments with an explicit psychological basis—now facilitate the implementation of computational psycholinguistic models of language processing (and computational cognitive models more generally)—although it is yet to be demonstrated that these models can scale up to the level of Computational Linguistic or AI systems.

Double R Model is implemented using the ACT-R 5.0 cognitive modeling environment (Anderson & LeBiere, 1998; Anderson et al. 2002). Unlike typical computational psycholinguistic models which tend to focus on specific linguistic issues (e.g. the processing of garden-path sentences, lexical disambiguation, the effect of verb argument preferences on language processing), Double R Model is intended as the basis for the development of large-scale, functional language comprehension systems. In this regard, it is perhaps better viewed as a computational linguistic system with a focus on psychological plausibility than as a purely psycholinguistic system.

Double R Model is currently capable of processing an interesting range of grammatical constructions including: 1) intransitive, transitive and ditransitive verbs; 2) verbs taking clausal complements; 3) predicate nominals, predicate adjectives and predicate prepositions; 4) conjunctions of numerous grammatical types; 5) modification by attributive adjectives, prepositional phrases and adverbs, etc. On the other hand, there are numerous grammatical forms that are not yet handled (e.g. relative clauses, various types of reduced clausal form, contextually determined prepositional phrase attachment). Double R Model accepts as input as little as a

single word or as much as an entire chunk of discourse (if the model stops in the middle of a text, it's a bug not a feature). Unrecognized words are simply ignored. Unrecognized grammatical forms result in partially analyzed text, not failure. Although Double R Model can process an entire discourse, it does not yet attempt to establish connections between the pieces of that discourse (via anaphora resolution, topic identification, pragmatic inferences, etc.). The model currently constructs representations of relational meaning and identifies referring expressions. The model lacks a mechanism for the construction of a situation model to referentially ground the representations and does not yet provide a grounded implementation of referential meaning. The model currently performs only limited lexical and structural disambiguation. It is not yet clear if the single level spreading activation mechanism of ACT-R 5.0 will prove adequate to support lexical and structural disambiguation. Integration of a spreading-activation mechanism like that available in C-I Theory (Kintsch 1998) could overcome this concern. The model lacks a backtracking mechanism for recovering from mistaken analyses, even when such backtracking is supported by empirical evidence (e.g. garden-path sentences). Automated techniques for scaling up the model to handle real-world input are under investigation, including the use of LSA Landauer et al., 2003), WordNet (Miller et al., 2003), FrameNet (Fillmore et al., 2003), and CYC (Lenat et al., 2003) for this purpose.

Inheritance and limited pattern matching as an alternative to and improvement on full unification

Unification allows for the unbounded, recursive matching of two logical representations. Unification is an extremely powerful pattern matching technique used in many language processing systems based on Prolog. Unfortunately, it is psychologically too powerful. For example, the following two logical expressions can be unified:

p(a,B,c(d,e,f(g,h(i,j),K),l))
p(X,b,c(Y,e,f(Z,T,U),l))

where capitalized letters are variables and lowercase letters are constants. Humans are unlikely to be capable of performing such unifications consciously or otherwise without significant effort and an external scratch pad to retain the large number of separate bindings involved.

On the other hand, although extremely powerful, unification does not support the matching of types to subtypes. Thus, if we have a verb type with intransitive and transitive verb subtypes, unification cannot unify a chunk of type "verb" with a chunk of type "intransitive verb" or "transitive verb". Unification's inability to match types to subtypes often results in a proliferation of rules (or conditions on rules) to handle the various combinations (PM fell victim to this limitation of unification). For example, the verb type can be variableized and a test for the valid types can be used to constrain the variable (e.g. Verb-Type equal verb or Verb-Type equal intrans-verb or Verb-Type equal trans-verb). With inheritance, a production that checks for a verb type will also match a transitive verb and an intransitive verb type (assuming an appropriate inheritance hierarchy). Humans appear to be able to use types and subtypes in appropriate contexts with little awareness of the transitions. For example, when processing a verb, all verbs (used predicatively) expect to be proceeded by a subject, but only transitive verbs expect to be followed by an object. Thus, humans presumably have available a general production that applies to all verbs (or even all predicates) which will look for a subject preceding the verb, but

only a more specialized production for transitive verbs (or transitive predicates) which will look for an object following the verb (or for a transitive verb preceding an object).

Inheritance supports the matching of two representations without requiring the recursive matching of their subparts so long as the types of the two representations are compatible. Types are essentially an abstraction mechanism which makes it possible to ignore the detailed internal structure of representations when comparing them. For example, once the model has identified an expression as an object referring expression, the model can match the object referring expression against productions without consideration of the internal structure of the object referring expression. Of course, there may be productions that do consider the internal structure, but types are useful here as well. Instead of having to fully elaborate the internal structure, types can be used to partially elaborate that structure. For example, if a production is specifically concerned with object referring expressions headed by a quantifier (e.g. "some of the books"), the production can check to see that the head is of the appropriate type, providing a (limited) unification like capability where needed.

In sum, ACT-R's inheritance and pattern matching capabilities provide a psychologically plausible alternative to a full unification capability.

Context accommodation as an alternative to and improvement on backtracking

Context accommodation is a mechanism for changing the function of an expression based on the context without backtracking. For example, when an auxiliary verb like "did" occurs it is likely functioning as a predicate specifier as in "he did not run" where the predicate is "run" and "did not" provides the specification for that predicate. However, auxiliary verbs may also function as predicates when they are followed by a noun phrase as in "he did it". Determining the ultimate function of an auxiliary verb can only be made when the expression following the auxiliary is processed. In a backtracking system, if the auxiliary verb is initially determined to be functioning as a predicate specifier, then when the noun phrase "it" occurs, the system will backtrack and reanalyze the auxiliary verb, perhaps selecting the predicate function on backtracking. However, note that backtracking mechanisms typically lose the context that forced the backtracking. Thus, on backtracking to the auxiliary verb, the system has no knowledge of the subsequent occurrence of a noun phrase to indicate the use of the auxiliary verb as a predicate. Thus, the system can only randomly select a new function for the auxiliary verb which may or may not be that of a predicate.

A better alternative is to accommodate the function of the auxiliary verb in the context which forces that accommodation. In this approach, when the noun phrase "it" is processed and the auxiliary verb functioning as a predicate specifier is retrieved, the function of the auxiliary verb can be accommodated in the context of a subsequent noun phrase to be a predicate. Context accommodation avoids the need to backtrack and allows the context to adjust the function of an expression just where that accommodation is supported by the context.

Feature Projection

Feature projection provides a mechanism for projecting lower level features up to the level at which they are needed for such things as resolving the referent of an object referring expression or determining the semantic category of an object or relation. This makes it possible for the root

node of a deeply nested meaning structure to contain the feature information needed for subsequent processing without the need to traverse the structure to find that information. Thus, the root node of the expression “the books” will have the [definite] feature to indicate that this expression is a definite referring expression and there is no need to traverse the structure to the representation of “the” in order to find this feature. The root node will also have the features [plural] and [book] to support reference identification.

Since it is possible for the features of expressions being combined to conflict some mechanism is needed for determining which feature projects when a conflict occurs. The following feature projection hierarchy is proposed: specifier > head > modifier. That is, specifier features dominate head features which dominate modifier features. For example, in the expression “the books” the specifier “the” has the feature [definite] whereas the plural noun “books” has the feature [indefinite]. The definiteness feature of the specifier projects to the overall expression and the expression “the books” is [definite].

Feature projection is currently only a projected feature of the model. However, it is expected to be important for tying the linguistic representations to a situation model and for discourse processing more generally.

Use of a Context Chunk and a Chunk Stack

The current ACT-R cognitive modeling environment provides only the goal and retrieval buffers (and perhaps the visual and aural buffers) to store the partial products of language comprehension. The lack of a goal stack is particularly constraining, since a stack is the primary data structure for managing the kind of recursion that is occurs in language. The key issue is the need to retain linguistic chunks in short-term working memory until they can be combined with other linguistic chunks. For example, in processing the phrase “the big ball”, the word “the” is processed first and recognized as a determiner. Determiners typically function as specifiers and combine with heads to form object referring expressions and the specifier must be retained in short-term working memory until the head is available to combine with it. The word “big” is processed second and recognized as an adjective functioning as a modifier. Modifiers combine with heads to form modified heads. The modifier must be retained in short-term working memory unit the head is available. Finally, the word “ball” is processed and determined to be a noun that can function as the head of an object referring expression. The modifier is retrieved from short-term working memory and combined with it to form a modified head. Then, the specifier is retrieved from short-term working memory and combined with the modified head to form an object referring expression. Note that there must be some mechanism for retrieving previously processed words from short-term working memory in last-in/first-out order during processing (subject to various kinds of error that can occur in the retrieval process). A stack provides this (essentially error free) capability. It is expected that a capacity to maintain about 5 separate linguistic chunks in short-term working memory is needed to handle most input—supporting at least one level of recursion (and perhaps two for the more gifted). The goal chunk could be adapted for this purpose, except that it is also the basis for creation of new declarative memory chunks and activation spread and these architectural needs would conflict. Further, it would be difficult to get the kind of stack like behavior needed out of the slots in the goal chunk.

To overcome these problems, Double R Model introduces a context chunk containing a bounded, circular stack of links to declarative memory. As chunks are stacked in the circular stack, if the number of chunks exceeds the limit of the stack, then new chunks replace the least

recently stacked chunks (supporting at least one type of short-term working memory error). The actual number of chunks allowed in the stack is specified by a global parameter. This parameter is settable to reflect individual differences in short-term working memory capacity. Chunks cannot be directly used from the stack. Rather, the stack is used to provide a template for retrieving the chunk from declarative memory. Essentially, the chunk on the stack provides a link to the corresponding declarative memory chunk. Since the chunk must be retrieved from declarative memory before use, the spreading activation and partial matching mechanisms of ACT-R are not circumvented as they were when popping goals off the goal stack as in earlier versions of ACT-R and retrieval errors are possible. Thus, the bounded, circular stack of links to declarative memory avoids the arguments against the goal stack of earlier versions of ACT-R, adds the insight of activated pathways to declarative memory, and retains the insights that motivated the inclusion of a goal stack in earlier versions of ACT-R.

The chunk stack is one element of the context chunk which is more generally used to separate out state information from the goal chunk. Since the goal chunk is the basis for creating new declarative memory chunks, storing the chunk stack in it would result in the chunk stack being stored with each new declarative memory chunk. While this might be used to support a kind of episodic memory where the context in which a word occurs is stored with the declarative memory chunk created during the processing of the word, ACT-R 5.0 does not currently provide a mechanism for transitioning episodic memory into semantic memory (i.e. abstracting from the context of use), and storing the context with a chunk has undesirable side-effects within the ACT-R cognitive modeling environment (e.g. it interferes with the spreading activation mechanism). To avoid such problems a separate context chunk is maintained and made available to all productions. Although, the existence of a separate context chunk that productions match to violates the ACT-R 5.0 architecture where only the buffers are supposed to be used for this purpose, earlier versions of ACT-R allowed multiple chunks to be matched on the left-hand side of productions and this functionality is still available in ACT-R 5.0.

The context chunk maintains several pieces of information in addition to the chunk stack. Its definition (as specified by a `chunk-type`) in the model is shown below:

```
(chunk-type context state rel-context sit-context text-context word prev-
word-1 prev-word-2 repeat chunk chunk-stack)

;; state =
;; retrieve-prev-chunk | -- previous chunk has been retrieved
;; process | -- process, default is to retrieve previous chunk
;; add-chunk-to-stack | -- add goal chunk to stack
;; add-chunks-to-stack | -- add goal and retrieval chunks to stack
;; rel-context = (relational context)
;; none | -- no context identified
;; obj | -- object context identified (e.g. following obj-spec)
;; pred | -- predicate context identified (e.g. following pred-spec)
;; sit -- situation context identified
;;  (e.g. following conjunction where first conjunct is sit-refer-expr)
;; sit-context = (situation context)
;; decl | -- declarative
;; yes-no-ques | -- yes-no-question
;; wh-ques | -- wh-question
;; imp | -- imperative
;; rel-clause | -- relative clause
;; passive | -- passive declarative
;; text-context = (text context)
```

```
;; end-of-text -- end of text
;; word -- current word (used for activation spread)
;; prev-word-1 -- previous word (used for activation spread)
;; prev-word-2 -- word before previous word (used for activation spread)
;; repeat =
;; yes |
;; no |
;; no-more
;; chunk -- current chunk
;; chunk-stack -- chunk stack
```

In this `chunk-type` definition, `context` is the name of the chunk, `state` is a slot that provides state information to guide production selection, `rel-context` is a slot that identifies the current relational context (typically determined by a specifier), `sit-context` is a slot that contains information about the current situation context, `text-context` is a slot that contains information about the larger discourse context, `word` contains the lexical item being processed, `word-prev-1` and `word-prev-2` contains the previous two words processed, `repeat` is `yes` if the word has been attended to previously and `no-more` if there are no more words in the input, `chunk` contains the most recently processed chunk, and `chunk-stack` contains the entire chunk stack.

In the ACT-R 5.0 cognitive modeling environment, only the goal chunk spreads activation to declarative memory. Since the `context` chunk contains contextually relevant information, it would be nice if it also spread activation (e.g. from the `word`, `prev-word-1` and `prev-word-2` slots). Unfortunately, this is not currently supported and an appropriate work around has not yet been developed.

Type Hierarchy

The importance of inheritance in the model was noted above. The current hierarchy of types is shown below with subtypes indented to the right of parent types. All subtypes are of type `top-type`, the base type. There are five general types: `lexical-type`, `referential-type`, `referring-expression-type`, `relation-top-type`, and `punctuation-type`.

```
Top-type
      Lexical-type
            Pronoun
                  Poss-pronoun
            Proper-noun
            Noun
            Adjective
            Verb
                  Intransitive-verb
                  Transitive-verb
                  Ditransitive-verb
                  Verb-subj-sitcomp (i.e. verb that takes
                        a situation complement)
                  Verb-subj-iobj-sitcomp (i.e. verb that takes an
                        indirect object and a situation complement)
                  Verb-subj-iobj-obj-sitcomp  (i.e.  verb  that  takes  two
                        objects and a situation complement)
            Preposition
            Adverb
```

```
      Determiner
      Quantifier
      Auxiliary
            Modal-Auxiliary
            Regular-Auxiliary
            Semi-Auxiliary
      Negative
            Not-negative (i.e. "not")

Referential-type
      Head
            Object-head
            Relational-head
      Specifier
            Object-specifier
                  Poss-object-specifier
                  Quantified-object-specifier
            Predicate-specifier
      Modifier
            Object-modifier (i.e. pre-head modifier)
            Post-modifier (i.e. post head modifier)
            Relation-modifier
            Predicate-modifier
                  Predicate-negative
                        Predicate-not-negative ;; i.e. "not"

Referring-expression-type
      Object-referring-expression
      Predicate-referring-expression
      Situation-referring-expression
      Location-referring-expression

Relation-top-type
      Object-type
      Relation-type
            Predicate-type
                  Intrans-predicate-type
                        Predicate-intransitive-verb
                  Transitive-predicate-type
                        Predicate-transitive-verb
                  Ditransitive-predicate-type
                        Predicate-ditransitive-verb
                  Sitcomp-predicate-type
                        Predicate-verb-subj-sitcomp
                  Iobj-sitcomp-predicate-type
                        Predicate-verb-subj-iobj-sitcomp
                  Iobj-obj-sitcomp-predicate-type
                        Predicate-verb-subj-iobj-obj-sitcomp
                  Predicate-adjective
                  Predicate-preposition
                  Predicate-auxiliary-verb
                  Predicate-nominal
            Preposition-object (i.e. prepositional phrase)
            Conjunction
Punctuation
```

The more specialized a production is, the more specialized the specification of the types of the chunks in the goal and retrieval buffers will be. The most general productions match a goal chunk whose type is `top-type` and ignore the retrieval buffer chunk.

Default Rules

The ACT-R cognitive modeling environment's single inheritance mechanism can be combined with the P parameter (P in the U = PG – C production utility equation) to establish default rules. For example, since all types extend a base type (i.e. `top-type`), then by using the base type as the value of the goal chunk in a production combined with a P value for the production that is lower than competing productions, then the production will function as a default production which is only selected if no more specific production matches. A sample default production is shown below:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; process chunk -- retrieve previous chunk
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; default
; - retrieve previous chunk
(p process-default--retrieve-prev-chunk
 =goal>
 ISA top-type
 =context>
 ISA context
 state process
 chunk-stack =chunk-stack
 =chunk-stack>
 ISA chunk-stack-chunk
 this-chunk =chunk
 prev-chunk =prev-chunk
 ==>
 =context>
 state retrieve-prev-chunk
 chunk-stack =prev-chunk
 +retrieval> =chunk
)
;; prefer competing productions
(spp process-default--retrieve-prev-chunk :p 0.75)
```

This default production causes the previous chunk to be retrieved from declarative memory if no other production is selected. Note that the production accesses the `chunk-stack` in the `context` chunk (via the `=chunk-stack` variable) and uses the value of the `this-chunk` slot (represented by the variable `=chunk`) to specify a retrieval from declarative memory (`+retrieval> =chunk`). The production does not use the value of `this-chunk` directly. To make this production a default production, the P parameter is set using the `spp` (set production parameter) command to a value of 0.75 (the default value for P is 1.0). The P parameter determines the likelihood of a goal succeeding if the production is executed. Default productions like this one are only likely to be selected and executed is there is no more specialized production available.

Lexical and Functional Entries

The lexical entries in the model provide a limited amount of information which is stored in the `word` and `word-info` chunks. The definition of the `word` and `word-info` chunk types are provided below:

```
(chunk-type word word-form word-marker)
(chunk-type word-info word-marker word-root word-type word-subtype word-
morph-type)
```

The `word-form` slot of the `word` chunk contains the physical form of the word (represented as a string in ACT-R); the `word-marker` slot contains an abstraction of the physical form. The `word-root` slot contains the value of the root form of the word. The `word-type` slot contains the lexical type of the word and is used to convert a `word-info` chunk into a `lexical-type` chunk for subsequent processing. A `word-subtype` slot is provided as a workaround for the lack of multiple inheritance in ACT-R 5.0. The `word-morph-type` slots supports the encoding of additional grammatical information (although that information is not currently being used).

Sample lexical entries for a pronoun, noun, verb participle, adjective, adverb, auxiliary, preposition, determiner, and quantifier are provided below:

```
(he-wf isa word
word-form "he"
word-marker he)
(he isa word-info
word-marker he
word-root he
word-type pronoun)

(cow-wf isa word
word-form "cow"
word-marker cow)
(cow isa word-info
word-marker cow
word-root cow
word-type noun
word-morph-type third-per-sing)
(cows-wf isa word
word-form "cows"
word-marker cows)
(cows isa word-info
word-marker cows
word-root cow
word-type noun
word-morph-type third-per-plur)

(running-wf isa word
word-form "running"
word-marker running)
(running isa word-info
word-marker running
word-type verb
word-root run
word-subtype intrans-verb
```

```
word-morph-type pres-part)

(rapid-wf isa word
word-form "rapid"
word-marker rapid)
(rapid isa word-info
word-marker rapid
word-type adjective)

(quickly-wf isa word
word-form "quickly"
word-marker quickly)
(quickly isa word-info
word-marker quickly
word-type adverb)

(is-aux-wf isa word
word-form "is"
word-marker is)
(is-aux isa word-info
word-marker is
word-root be
word-type reg-aux)


(on-wf isa word
word-form "on"
word-marker on)
(on isa word-info
word-marker on
word-type preposition)

(the-wf isa word
word-form "the"
word-marker the)
(the isa word-info
word-marker the
word-type determiner)

(all-wf isa word
word-form "all"
word-marker all)
(all isa word-info
word-marker all
word-type quantifier)
```

Note that there is no indication of the functional roles (e.g. head, modifier, specifier) that particular lexical items may fulfill. Following conversion of `word-info` chunks into `lexical-type` chunks (e.g., `verb`, `adjective`), functional roles are dynamically assigned by the productions that are executed during the processing of a piece of text. Since functional role chunks are dynamically created, only `chunk-type` definitions exist for functional categories prior to that processing. As an example of a `chunk-type` definition for a functional category, consider the category `pred-trans-verb` (i.e. transitive verb functioning as a predicate) whose definition involves several hierarchically related `chunk-types` as shown below:

```
(chunk-type top-type head)
(chunk-type (rel-type-top (:include top-type)))
(chunk-type (rel-type (:include rel-type-top)))
(chunk-type (pred-type (:include rel-type)) subj spec mod post-mod)
(chunk-type (trans-pred-type (:include pred-type)) obj)
(chunk-type (pred-trans-verb (:include trans-pred-type)))
```

The `top-type chunk-type` contains the single slot `head`. All types that are subtypes of `top-type` inherit the `head` slot. `Rel-type-top` is a subtype of `top-type` that doesn't add any additional slots. `Rel-type` is a subtype of `rel-type-top` that doesn't add any additional slots. `Pred-type` is a subtype of `rel-type` that adds the slots `subj`, `spec`, `mod`, and `post-mod`. It is when a relation is functioning as a predicate that these slots become relevant. `Trans-pred-type` is a subtype of `pred-type` that adds the slot `obj`. Finally, `pred-trans-verb` is a subtype of `trans-pred-type` that doesn't add any new slots. Summarizing, `pred-trans-verb` contains the slots `head`, `subj`, `spec`, `mod`, `post-mod`, and `obj`, all of which are inherited for parent types.

The following production creates an instance of a pred-trans-verb and provides initial values for the slots:

```
(p process-verb--convert-to-pred-trans-verb
 =goal>
 ISA verb
 head =verb
 subtype trans-verb
 =context>
 ISA context
 state convert-verb-to-pred-verb
 ==>
 +goal>
 ISA pred-trans-verb
 subj none
 spec none
 mod none
 head =goal
 post-mod none
 obj none
 =context>
 state retrieve-prev-chunk
)
```

In this production, a `verb` (subtype of `lexical-type`) whose `subtype` slot has the value `trans-verb` is converted into a `pred-trans-verb` for subsequent processing. The only slot of `pred-trans-verb` that is given a value other than `none` is the `head` slot whose value is set to be the `goal` chunk (i.e. `head =goal`). This production has the effect of assigning a transitive verb the functional role of predicate (specialized as a transitive verb predicate). Its selection and execution is based on the previous context which set the value of the `state` slot of the `context` chunk to `convert-verb-to-pred-verb` and on having a goal chunk of type `verb`.

Productions

Sample productions were shown above in the discussion of default rules and in the creation of functional roles. This section provides some additional examples. The `read-next-word` production initiates the find-attend-encode sequence for reading the next word from the computer screen (using ACT-R's perceptual component).

```
(p read-next-word
 =goal>
 ISA word
 =context>
 ISA context
 state start
 - repeat no-more ;; no more words
 ==>
 =context>
 state find
)
```

Note that the goal is represented by the declarative `word` chunk as opposed to a more procedurally oriented goal chunk like `read-word`. This is fallout from the fact that the goal chunk is the basis for creating declarative memory chunks. The `start` value of the `state` slot in the `context` chunk is the primary basis for the selection of this production (along with the type of the goal chunk). Again, putting the `state` slot in the `context` chunk avoids the need to encode procedural information in the goal chunk. The "`- repeat no-more`" entry in the production indicates that this production only applies if the value of the `repeat` slot is not (negation is indicated by the "-") `no-more` where `no-more` indicates that the last word in the text has already been read.

The next production uses the `word-marker` slot of the `word` chunk to retrieve the `word-info` chunk.

```
(p retrieve-word-info
 =goal>
 ISA word
 word-marker =word-marker
 =context>
 ISA context
 state retrieve
 ==>
 +retrieval>
 ISA word-info
 word-marker =word-marker
 =context>
 state retrieve-word-info
 )
```

The `word-info` chunk is then used to create a `lexical-type` chunk (e.g. `verb`) which becomes the goal. The productions that convert `word-info` chunks into `lexical-type` chunks are special in that the :effort parameter is set to 0.0. The :effort parameter determines how long it takes a procedure to execute (default is 0.05 sec or 50 msec). Setting the value to 0.0 means that the

production takes no time to execute. The presumption is that the procedure that effects this conversion is substituting for an automated type conversion mechanism that the ACT-R 5.0 cognitive modeling environment does not currently provide.

```
(p convert-word-to-verb
 =goal>
 ISA word-info
 word-type verb
 word-subtype =word-subtype
 =context>
 ISA context
 state convert
 ==>
 +goal>
 ISA verb
 head =goal
 subtype =word-subtype
 =context>
 state process
)
;; housekeeping only!!!
(spp convert-word-to-verb :effort 0.0)
```

There are a few other kinds of "housekeeping" productions which are accorded zero effort in the model (e.g. the stack chunking procedures). In general, "housekeeping" productions are used to effect various data manipulations that are external to the basic processing mechanism. The `process-verb--convert-to-pred-trans-verb` production discussed above is another example of a "housekeeping" production.

The next production matches a `verb` goal chunk and in the context of an `obj` (i.e. object referring expression) converts the `verb` type into a `rel-head` type

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; process verb in object context
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; process verb in object context

; process verb
; - obj-context
(p process-verb--obj-context--convert-to-rel-head
 =goal>
 ISA verb
 head =verb
 =context>
 ISA context
 state retrieve-prev-chunk
 rel-context obj
 ==>
 +goal>
 ISA rel-head
 mod none
 head =goal
 post-mod none
)
```

Rel-head (i.e. relational-head) is a subtype of head. The next production matches a head goal chunk (which could be a rel-head) and an obj-spec (i.e. object-specifier) retrieval chunk and creates a new obj-refer-expr (i.e. object-referring-expression) which becomes the goal. Together, these two productions support to use of verbs as (relational) heads of object referring expressions following an object specifier (e.g. “kick” in “the kick”).

```
; process head
; - obj-spec retrieved
(p process-head--prev-chunk-is-obj-spec
 =goal>
 ISA head
 =context>
 ISA context
 state retrieve-prev-chunk
 =retrieval>
 ISA obj-spec
 ==>
 +goal>
 ISA obj-refer-expr
 spec =retrieval
 mod none
 head =goal
 post-mod none
 referent none-for-now
 =context>
 state process
 rel-context none
 )
```

The creation of an object referring expression causes the value of the rel-context slot to be set to none indicating the end of the object referring expression context.

Processing Example

As an example of the processing of a piece of text and the creation of declarative memory chunks to represent the meaning of the text, consider the processing of the following text:

> The old dog lover is asleep.

The processing of the word “the” invokes the following procedures:

```
Time 0.000: Read-First-Word
Time 0.050: Find-First-Word
Time 0.100: Attend-Word-Option-2--Location-Retrieved
Time 0.235: Encode-Word-Form
Time 0.285: The-Wf Retrieved
Time 0.285: Retrieve-Word-Marker--Option-1--Marker-Retrieved
Time 0.335: Retrieve-Word-Info
Time 0.385: The Retrieved
Time 0.385: Retrieve-Word-Info--Option-1--Word-Info-Retrieved
Time 0.435: Convert-Word-To-Determiner
Time 0.435: Convert-Determiner-To-Obj-Spec
```

```
 Time 0.435: Process--Default--Retrieve-Prev-Chunk
 Time 0.485: None Retrieved
 Time 0.485: Process-Obj-Spec--Default--Add-Chunks-To-Stack
 Time 0.535: Add-Chunks-To-Stack--Option-2
 Time 0.535: Add-Chunk-To-Stack
```

and the following declarative memory chunks are created:

```
Goal25
 isa DETERMINER
 head The
Goal26
 isa OBJ-SPEC
 head Goal25
 mod None
```

It takes 535 msec to process the word "the" and two declarative memory chunks are created. The first chunk, `goal25`, is a `determiner` whose `head` slot has the value is `The`. This chunk represents the inherent part of speech of the word "the". The second chunk, `goal26`, is an `obj-spec` (i.e. `object-specifier`) whose `head` slot has the value `Goal25` and whose `mod` slot has the value `none`. This second chunk represents the function of "the" in this particular text. Note that if "the" were the only word in the input text, the creation of these two chunks would still occur since the processing mechanism works bottom-up from the lexical items and makes no assumptions about what will occur independently of the lexical items.

The processing of the second word "old" executes the following productions:

```
 Time 0.535: Read-Next-Word
 Time 0.585: Find-Next-Word-Step-1-Option-1
 Time 0.635: Attend-Word-Option-2--Location-Retrieved
 Time 0.770: Encode-Word-Form
 Time 0.820: Old-Wf Retrieved
 Time 0.820: Retrieve-Word-Marker--Option-1--Marker-Retrieved
 Time 0.870: Retrieve-Word-Info
 Time 0.920: Old Retrieved
 Time 0.920: Retrieve-Word-Info--Option-1--Word-Info-Retrieved
 Time 0.970: Convert-Word-To-Adjective
 Time 0.970: Process-Adjective--Obj-Context
 Time 1.020: Process--Default--Retrieve-Prev-Chunk
 Time 1.070: Goal26 Retrieved
 Time 1.070: Process--Default--Add-Chunks-To-Stack
 Time 1.120: Add-Chunks-To-Stack--Option-1
 Time 1.120: Add-Chunk-To-Stack
```

and leads to the creation of the following declarative memory chunks:

```
Goal32
 isa ADJECTIVE
 head Old
Goal33
 isa OBJ-MOD
 head Goal32
 mod None
```

where goal32 represents the inherent part of speech of "old" and goal33 represents the function of "old" (i.e. object-modifier) in the current context.

The processing of the third word "dog" executes the following productions:

```
 Time 1.120: Read-Next-Word
 Time 1.170: Find-Next-Word-Step-1-Option-1
 Time 1.220: Attend-Word-Option-2--Location-Retrieved
 Time 1.355: Encode-Word-Form
 Time 1.405: Dog-Wf Retrieved
 Time 1.405: Retrieve-Word-Marker--Option-1--Marker-Retrieved
 Time 1.455: Retrieve-Word-Info
 Time 1.505: Dog Retrieved
 Time 1.505: Retrieve-Word-Info--Option-1--Word-Info-Retrieved
 Time 1.555: Convert-Word-To-Noun
 Time 1.555: Convert-Noun-To-Head
 Time 1.605: Process--Default--Retrieve-Prev-Chunk
 Time 1.655: Goal33 Retrieved
 Time 1.655: Process-Head--Prev-Chunk-Is-Modifier--Option-1
 Time 1.705: Process--Default--Retrieve-Prev-Chunk
 Time 1.755: Goal26 Retrieved
 Time 1.755: Process-Head--Prev-Chunk-Is-Obj-Spec
 Time 1.805: Process--Default--Retrieve-Prev-Chunk
 Time 1.855: None Retrieved
 Time 1.855: Process--Default--Add-Chunks-To-Stack
 Time 1.905: Add-Chunks-To-Stack--Option-2
 Time 1.905: Add-Chunk-To-Stack
```

and creates the following declarative memory chunks:

```
Goal39
 isa NOUN
 head Dog
Goal40
 isa HEAD
 head Goal39
 mod Goal33
 post-mod None
Goal41
 isa OBJ-REFER-EXPR
 head Goal40
 referent None-For-Now
 spec Goal26
 mod None
 post-mod None
```

Goal41 is a full object referring expression and contain a referent slot to support a link to an object in the situation model corresponding to this piece of text. Unfortunately, the model does not currently establish the value of the referent slot.

The processing of the fourth word "lover" executes the following productions:

```
 Time 1.905: Read-Next-Word
 Time 1.955: Find-Next-Word-Step-1-Option-1
 Time 2.005: Attend-Word-Option-2--Location-Retrieved
 Time 2.140: Encode-Word-Form
```

```
 Time 2.190: Lover-Wf Retrieved
 Time 2.190: Retrieve-Word-Marker--Option-1--Marker-Retrieved
 Time 2.240: Retrieve-Word-Info
 Time 2.290: Lover Retrieved
 Time 2.290: Retrieve-Word-Info--Option-1--Word-Info-Retrieved
 Time 2.340: Convert-Word-To-Noun
 Time 2.340: Convert-Noun-To-Head
 Time 2.390: Process--Default--Retrieve-Prev-Chunk
 Time 2.440: Goal41 Retrieved
 Time 2.440: Process-Head--Prev-Chunk-Is-Obj-Refer-Expr--Step-1
 Time 2.490: Process-Head--Prev-Chunk-Is-Obj-Refer-Expr--Step-2
 Time 2.490: Process-Head--Prev-Chunk-Is-Obj-Refer-Expr--Step-3
 Time 2.490: Process--Default--Retrieve-Prev-Chunk
 Time 2.540: None Retrieved
 Time 2.540: Process--Default--Add-Chunks-To-Stack
 Time 2.590: Add-Chunks-To-Stack--Option-2
 Time 2.590: Add-Chunk-To-Stack
```

and creates or modifies the following declarative memory chunks:

```
Goal47
 isa NOUN
 head Lover
Goal48
 isa HEAD
 head Goal47
 mod Goal40
 post-mod None
Goal41
 isa OBJ-REFER-EXPR
 head Goal48
 referent None-For-Now
 spec Goal26
 mod None
 post-mod None
```

Note that `goal47` (i.e. “lover”) is now the head of `goal48` which is the head of the `object-referring-expression` (i.e. `goal41`) with `goal39` (i.e. “dog”) functioning as the head of `goal40` which is functioning as a modifier of `goal48`. Also, note that `goal33` (i.e. “old”) modifies `goal39` (i.e. “dog”) and not `goal48` (i.e. “dog lover”). This is equivalent to the expression “the lover of old dogs” rather than “the old lover of dogs”, both of which are possible interpretations. Having “old” modify “dog” rather than “dog lover” is probably not the preferred interpretation of this piece of text, but it does point out the advantage of having an implemented model to make such decision points apparent. The current model can be modified to support the alternative interpretation by addition of a production that has the intended effect, however, there is currently no mechanism for preferring this new production over the existing production. Such a mechanism would need to be able to distinguish between collocations like “old dog lover” and collocations like “old house renovator” where the modification works the other way.

Continuing with the next word “is” leads to the execution of the following productions:

```
 Time 2.590: Read-Next-Word
 Time 2.640: Find-Next-Word-Step-1-Option-1
 Time 2.690: Attend-Word-Option-2--Location-Retrieved
```

```
 Time 2.825: Encode-Word-Form
 Time 2.875: Is-Aux-Wf Retrieved
 Time 2.875: Retrieve-Word-Marker--Option-1--Marker-Retrieved
 Time 2.925: Retrieve-Word-Info
 Time 2.975: Is-Aux Retrieved
 Time 2.975: Retrieve-Word-Info--Option-1--Word-Info-Retrieved
 Time 3.025: Convert-Word-To-Reg-Aux
 Time 3.025: Process--Default--Retrieve-Prev-Chunk
 Time 3.075: Goal41 Retrieved
 Time 3.075: Process-Reg-Aux--Default--Convert-To-Pred-Spec
 Time 3.125: Process--Default--Add-Chunks-To-Stack
 Time 3.175: Add-Chunks-To-Stack--Option-1
 Time 3.175: Add-Chunk-To-Stack
```

and the creation of the following chunks:

```
Goal54
 isa REG-AUX
 head Is-Aux
Goal55
 isa PRED-SPEC
 head Aux-1
 mod None
 modal-aux None
 neg None
 aux-1 Goal54
 aux-2 None
 aux-3 None
```

Note that the `pred-spec` chunk type has a `modal-aux`, `neg`, and three auxiliary slots (`aux-1`, `aux-2`, and `aux-3`) to handle the range of predicate specifiers that can occur. For this instance of a `pred-spec` (i.e. `goal55`), `goal54` fills the `aux-1` slot and functions as the head of `goal55`.

The processing of the final word "asleep" executes the following productions:

```
 Time 3.175: Read-Next-Word
 Time 3.225: Find-Next-Word-Step-1-Option-1
 Time 3.275: Attend-Word-Option-2--Location-Retrieved
 Time 3.410: Encode-Word-Form
 Time 3.460: Asleep-Wf Retrieved
 Time 3.460: Retrieve-Word-Marker--Option-1--Marker-Retrieved
 Time 3.510: Retrieve-Word-Info
 Time 3.560: Asleep Retrieved
 Time 3.560: Retrieve-Word-Info--Option-1--Word-Info-Retrieved
 Time 3.610: Convert-Word-To-Adjective
 Time 3.610: Process-Adjective--Predicate-Context
 Time 3.660: Process--Default--Retrieve-Prev-Chunk
 Time 3.710: Goal55 Retrieved
 Time 3.710: Process-Pred-Type--Prev-Chunk-Is-Pred-Spec
 Time 3.760: Process--Default--Retrieve-Prev-Chunk
 Time 3.810: Goal41 Retrieved
 Time 3.810: Process-Pred-Type--Prev-Chunk-Is-Obj-Refer-Expr
 Time 3.860: Process--Default--Retrieve-Prev-Chunk
 Time 3.910: None Retrieved
 Time 3.910: Process--Default--Add-Chunks-To-Stack
 Time 3.960: Add-Chunks-To-Stack--Option-2
```

```
 Time 3.960: Add-Chunk-To-Stack
```

and creates the following declarative memory chunks:

```
Goal61
 isa ADJECTIVE
 head Asleep
Goal62
 isa PRED-ADJ
 head Goal61
 subj Goal41
 spec Goal55
 mod None
 post-mod None
```

In the context of the predicate specifier "is" (i.e. `goal55`), the adjective "asleep" functions as a predicate adjective filling the `head` slot of `goal62`. `Goal41` (an object referring expression) fills the `subj` slot (i.e. subject) of `goal62`.

Following the processing of "asleep" the model attempts to read the next word, executing the following productions:

```
 Time 3.960: Read-Next-Word
 Time 4.010: Find-Next-Word-Step-1-Option-1
 Time 4.060: Attend-Word-Option-1--Error-Retrieved
 Time 4.110: Attend-Word-Option-2--Error-Retrieved-Again
 Time 4.160: Process--Default--Retrieve-Prev-Chunk
 Time 4.210: Goal62 Retrieved
 Time 4.210: Process-End-Of-Sit--Pred-Type-Retrieved--Convert-To-Sit-Refer-
Expr
 Time 4.260: Process--Default--Retrieve-Prev-Chunk
 Time 4.310: None Retrieved
 Time 4.310: Process-Sit-Refer-Expr--Add-Chunks-To-Stack
 Time 4.360: Add-Chunks-To-Stack--Option-2
 Time 4.360: Add-Chunk-To-Stack
 Time 4.360: * Nothing to run: No productions, no events.
```

The failure to read a word signals the end of processing and a wrap-up production (i.e. `process-end-of-sit--pred-type-retrieved--convert-to-sit-refer-expr`) is executed. This production converts `goal62` into a situation referring expression resulting in the creation of `goal65` with `goal62` filling the `head` slot.

```
Goal65
 isa SIT-REFER-EXPR
 head Goal62
 referent None-For-Now
 mod None
```

At the end of processing a single chunk of type `situation-referring-expression` is available in the `chunk-stack` to support subsequent processing. Whether or not the `subject` of the `predicate-adjective` should be separately available in the `chunk-stack` to reflect the profiling of the subject in addition to the situation is an open research question.

The Significance of Double R Theory

Double R Grammar provides a meaningful basis for understanding the linguistic categories head, modifier, specifier and complement, and in so doing, clears up much of the confusion that has resulted from the attempt to explicate these linguistic concepts in purely distributional terms. Further, the consideration of relational meaning—especially the consideration of relational meaning in object referring expressions (i.e. nominals)—is an oft-overlooked dimension of meaning in most other grammatical approaches. Finally, showing that referential and relational meaning can be integrated into a single unified representation is the key contribution of Double R Grammar to the linguistic study of language.

Double R Process is a psychologically plausible processing mechanism for constructing representations of referential and referential meaning directly from input text without a separate syntactic analysis component. Double R Process, together with Double R Grammar, overcomes a key argument levied against numerous other "direct to meaning" based approaches – **representational inadequacy**. Direct to meaning approaches typically rely on the use of highly context dependent semantic categories (e.g. "vehicle-type", "employment-category") or semantic primitives (e.g. PTRANS, "cause-not-be-alive") which have been shown to be representationally inadequate for representing the full range of meaning based language variation. And propositional systems of representation which are based on abstract concepts, beg the question of how one goes from linguistic input to abstract concepts without relying on abstractions and simplifications that lead to similar representational inadequacies. Double R Grammar avoids these representational inadequacies by retaining the linguistic form of the input and not abstracting away from that form. Double R Process demonstrates the viability of a bottom-up, lexically driven approach to language comprehension based on a direct interpretation of text into meaning without a separate syntactic analysis component, without a phrase structure grammar, and without a special grammatical unit like the sentence (e.g. Townsend and Bever, 2001).

The explicit implementation of the underlying grammatical theory and processing mechanism in a psychologically constrained cognitive modeling environment provides a mechanism for testing and validating the theory. Further, Double R Model provides a basis for the development of large-scale language comprehension systems—an atypical goal for a psycholinguistically motivated language comprehension system.

The combination of an underlying linguistic theory for the integrated representation of referential and relational meaning, a processing mechanism for constructing integrated representations of referential and relational meaning, and a computational psycholinguistic model that demonstrates the viability of the linguistic theory and processing mechanism is a powerful demonstration of what can be achieved in an interdisciplinary research program within the broad sweep of the cognitive sciences.

Acknowledgments

I would like to thank Kevin Gluck and Wink Bennett of the Air Force Research Laboratory for allowing me to continue this research.

References

Abney, S. (1987). *The English Noun Phrase in its Sentential Aspect.* PhD dissertation, MIT.

Abney, S. (1991). "Parsing by Chunks." In Berwick, R., Abney, S. and Tenney, C. (eds.), *Principle Based Parsing*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Altmann, E. and J. Trafton (2002). "Memory for goals: an activation-based model." In *Cognitive Science*, Volume 26, pp. 39-83.

Anderson, J. R. (1976). *Language, Memory and Thought*. Hillsdale, NJ: LEA.

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, NJ: LEA.

Anderson, J. & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahway, MJ: LEA.

Anderson, J., Bothell, D., Byrne, M. and LeBiere, C (2002). *An Integrated Theory of the Mind*. http://act-r.psy.cmu.edu/papers/403/IntegratedTheory.pdf.

Arbib, M. & J. Hill (1988). "Language Acquisition: Schemas Replace Universal Grammar." In *Explaining Language Universals*. Edited by J. Hawkins. Oxford: Basil Blackwell

Ball, J. (1991). "Parsing English Directly into Propositional Representations." In M. McTear and N. Creaney (eds.) *AI and Cognitive Science '90*. New York: Springer-Verlag.

Ball, J. (1992). *PM, Propositional Model, a Computational Psycholinguistic Model of Language Comprehension Based on a Relational Analysis of Written English*. Ann Arbor, MI: UMI Dissertation Information Service.

Ball, J. (2003a). "Towards a Semantics of X-Bar Theory." http://www.DoubleRTheory.com/papers/other/SemanticOfXBarTheory.pdf

Ball, J (2003b). "Double R Grammar". http://www.DoubleRTheory.com/papers/doubler/Double R Grammar.pdf

Barsalou, L. (1999). "Perceptual Symbol Systems." In *Behavioral and Brain Sciences* 22, 577-609.

Barwise, J. & J. Perry (1983). *Situations and Attitudes.* Cambridge, MA: The MIT Press.

Berwick, R. & A. Weinberg (1984). *The Grammatical Basis of Linguistic Performance*. Cambridge, MA: The MIT Press.

Bloomfield, L. (1933). *Language*. Holt, Rinehart & Winston, New York, NY.

Cann, R. (1999). "Specifiers as Secondary Heads." In *Specifiers, Minimalist Approaches*, edited by Adger, D. Pintzuk, S, Plunkett, B & Tsoulas, G. Oxford University Press, Oxford, UK.

Chomsky, N. (1970). "Remarks on nominalization." In *Readings in Transformational Grammar*. Edited by R. Jacobs & P. Rosenbaum. Boston: Ginn.

Chomsky, N. (1981). *Lectures on Government and Binding*. Dordrecht-Holland: Foris

Chomsky, N. (1982). *Some Concepts and Consequences of the Theory of Government and Binding*. Cambridge, MA: The MIT Press.

Chomsky, N. (1995). *The Minimalist Program.* Ellis Horwood, The MIT Press, Cambridge, MA.

Clark, H. (1983). "Making sense of nonce sense." In *The Process of Language Understanding.* Edited by G. Flores d'Arcais & R. Jarvella. John Wiley, New York, NY.

Collins, A. & M. Quillian (1969). "Retrieval time from semantic memory." *Journal of Verbal Learning and Verbal Behavior*, 8, pp. 240-248.

Collins, A. & E. Loftus (1975). "A Spreading-Activation Theory of Semantic Processing." *Psychological Review*, 82, pp. 240-248.

Conrad, C. (1972). "Cognitive Economy in Semantic Memory." *Journal of Experimental Psychology*, 92, pp. 149-154.

Dixon, R. (1991). *A New Approach to English Grammar, on Semantic Principles*. Oxford University Press: New York, NY.

Ernst, T. (1991). "A Phrase Structure Theory of Tertiaries." In *Syntax and Semantics 25*, ed. By S. Rothstein. Academic Press, New York, NY.

Fillmore, C. (1977). "Scenes-and-frames semantics." In *Linguistic Structures Processing* pp. 55-82. Edited by A. Zampolli. Holland: North Holland Publishing.

Fillmore et al. (2003). FrameNet. http://www.icsi.berkeley.edu/~framenet/

Frazier, L. & J. D. Fodor (1978). "The sausage machine: a new two-stage parsing model." *Cognition*, 6, pp. 291-328.

Gazdar, G., Klein, E, Pullum, G. and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, MA.

Gomez, R. & Gerken, L. (2000). "Infant artificial language learning and language acquisition." *Trends in Cognitive Sciences* 4:5, 178-186.

Hawkins, J. (1984). "Modifier-Head or Function-Argument Relations in Phrase Structure? The evidence of some word order universals." *Lingua* 63, 107-138.

Hocket, C. (1958). *A Course in Modern Linguistics*. New York: The MacMillan Company.

Hudson, R. (1984). *Word Grammar*. Oxford: Basil Blackwell.

Jackendoff, R. (1977). *X-Bar Syntax.* The MIT Press, Cambridge, MA.

Jackendoff, R. (1983). *Semantics and Cognition*. The MIT Press, Cambridge, MA.

Jackendoff, R. (2002). *Foundations of Language.* Oxford University Press, New York, NY.

Johnson-Laird, P. (1983). *Mental Models*. Cambridge, MA: Harvard University Press.

Kamp, H. and Reyle, U. (1993). *From discourse to logic: Introduction to the model theoretic semantics of natural language, formal logic and discourse representation theory*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Kempen, G. (1996). "Computational Models of Syntactic Processing in Language Comprehension". In Dijkstra, T. & de Smedt, K., *Computational Psycholinguistics, AI and Connectionist Models of Human Language Processing*. London: Taylor & Francis.

Kintsch, W. (1998). Comprehension, a Paradigm for Cognition. New York, NY: Cambridge University Press.

Kintsch, W. (2001). "Predication". *Cognitive Science* 25, 173-202.

Lakoff, G. (1988). "Cognitive Semantics." In *Meaning and Mental Representation*. Edited by U. Eco, M. Santambrogio & P. Violi. Indianapolis: Indiana University Press.

Lakoff, G. (1987). *Women, Fire and Dangerous Things*. Chicago: The University of Chicago Press.

Lakoff, G., & M. Johnson (1980). *Metaphors We Live By*. Chicago: The University of Chicago Press.

Landauer, T. & S. Dumais (1997). "A solution to Plato's problem: The Latent Semantic Analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104, pp. 211-240.

Landauer et al. (2003). Latent Semantic Analysis (LSA). http://lsa.colorado.edu/

Langacker, R. (1987). *Foundations of Cognitive Grammar, Volume 1, Theoretical Prerequisites*. Stanford, CA: Stanford University Press.

Langacker, R. (1991). *Foundations of Cognitive Grammar, Volume 2, Descriptive Application*. Stanford, CA: Stanford University Press.

Langacker, R. (1998). "Conceptualization, Symbolization, and Grammar." In M. Tomasello (ed.), *The New Psychology of Language, Cognitive and Functional Approaches to Language Structure*. Mahway, NJ: LEA.

Lenat et al. (2003). CYC. http://www.cyc.com

Lyons, J. (1968). *Theoretical Linguistics*. New York: Cambridge University Press.

Meyer, D., & R. Schvaneveldt (1971). "Facilitation in recognizing pairs of words: evidence of a dependence between retrieval operations." *Journal of Experimental Psychology*, 90, pp. 227-234.

Miller et al. (2003). WordNet. http://www.cogsci.princeton.edu/~wn/

Pullum, G. (1991). "English nominal gerunds as noun phrases with verb phrase heads." *Linguistics* 29, pp. 763-99.

Quirk, R., S. Greenbaum, G. Leech, & J. Svartvik (1972). *A Grammar of Contemporary English*. London: Longman

Rosch, E. (1978). "Principles of Categorization." In *Cognition and Categorization*. Edited by E. Rosch & B. Lloyd. Hillsdale, NJ: LEA.

Sadler, L. & Arnold, D. (1994). "Prenominal adjectives and the phrasal/lexical distinction." *Journal of Linguistics*, 30, pp. 187-226.

Speas, M. (1990). *Phrase Structure in Natural Language.* Kluwer, London, UK.

Steedman, (2000). *The Syntactic Process*. Cambridge, MA: The MIT Press.

Stowell, T. (1989). "Subjects, Specifiers, and X-Bar Theory." In *Alternative Conceptions of Phrase Structure*, ed. By M. Baltin & A. Kroch, The University of Chicago Press, Chicago, IL.

Talmy, L. (2003). *Toward a Cognitive Semantics, Vols I and II.* Cambridge, MA: The MIT Press

Taylor, J. (1992). *Linguistic Categorization, Prototypes in Linguistic Theory*. Oxford: Clarendon Press.

Thomason, R., & R. Stalnaker (1973). "A Semantic Theory of Adverbs." *Linguistic Inquiry*, 4, pp. 195-220.

Tomasello, M. (ed) (1998). *The New Psychology of Language*. Mahway, NJ: LEA.

Townsend, D. and T. Bever (2001). *Sentence Comprehension*. Cambridge, MA: The MIT Press.

Wilks, Y. (1975). "Preference Semantics." In *Formal Semantics*. Edited by E. Keenan. NY: Cambridge University Press.

Wittgenstein, L. (1953). *Philosophical Investigations*. New York: MacMillan.