IMPLEMENTING COGNITIVE GRAMMAR ON A COGNITIVE ARCHITECTURE: A CASE STUDY WITH ACT-R

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF INFORMATICS

OF

MIDDLE EAST TECHNICAL UNIVERSITY

BY

EVGUENI A. STEPANOV

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COGNITIVE SCIENCE

SEPTEMBER 2004

Approval of the Graduate School of Informatics

Prof. Dr. Neşe Yalabık
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Deniz Zeyrek
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Deniz Zeyrek	Assist. Prof. Dr. Bilge Say
Co-Supervisor	Supervisor

Examining Committee Members

Assoc. Prof. Dr. Cem Bozşahin

Assist. Prof. Dr. Bilge Say

Prof. Dr. Hasan Gürkan Tekman

Dr. Meltem Turhan Yöndem

Prof. Dr. Deniz Zeyrek

(Evgueni A. Stepanov)

ABSTRACT

IMPLEMENTING COGNITIVE GRAMMAR ON A COGNITIVE ARCHITECTURE:
A CASE STUDY WITH ACT-R

Stepanov, Evgueni A.

MS, Department of Cognitive Science

Supervisor: Assist. Prof. Dr. Bilge Say

Co-supervisor: Prof. Dr. Deniz Zeyrek

September 2004, 169 pages

Cognitive Grammar is a theory within the framework of Cognitive Linguistics that gives an account of human linguistic ability based entirely on general cognitive abilities. Because of the general complexity and open-endedness of the theory, there is not much computational work associated with it. This thesis proposes that ACT-R cognitive architecture can provide the basic primitives for the cognitive abilities required for a better implementation of Cognitive Grammar. Thus, a language model was developed on the ACT-R architecture. The model processes active and passive sentences, constructs their propositional representations, and tests the representation on a sentence verification task of the experiment of Anderson (1974).

Keywords: Cognitive Grammar, Cognitive Architecture, ACT-R

ÖZ

BİLİŞSEL GRAMERİN BİLİŞSEL MİMARİDE GERÇEKLEŞTİRİLMESİ:
BİR ACT-R ÇALIŞMASI

Stepanov, Evgueni A.

Yüksek Lisans, Bilişsel Bilimler Bölümü

Tez Yöneticisi: Y. Doç. Dr. Bilge Say

Ortak Tez Yöneticisi: Prof. Dr. Deniz Zeyrek

Eylül 2004, 169 sayfa

Bilişsel Gramer Bilişsel Dilbilimi çerçevesinde var olan ve insanların dil kabiliyetlerini tamamen genel bilişsel kabiliyetlerle açıklayan bir teoridir. Teorinin genel karmaşıklığı ve açık uçlu olmasından dolayı şimdiye dek teori ile ilgili pek fazla bilgisayarla modellenmeye dayalı çalışma yapılmamıştır. Bu tez ACT-R Bilişsel Mimarisinin Bilişsel Gramerin iyi bir uygulaması için gereken temel bilişsel kabiliyetleri sağladığını göstermeyi amaçlamaktadır. Bunun için ACT-R mimarisinde bir dil modeli geliştirilmiştir. Model etken ve edilgen tümceleri işleyip onların önermesel gösterimlerini oluşturmakta, bu önermesel gösterimi Anderson'un (1974) deneyindeki tümce doğrulama deneyinde sınamaktadır.

Anahtar kelimeler: Bilişsel Gramer, Bilişsel Mimari, ACT-R

ACKNOWLEDGEMENTS

I am extremely grateful to my supervisor, Assist. Prof. Dr. Bilge Say, for her guidance and support throughout the course of this work. Special thanks go to my co-supervisor, Prof. Dr. Deniz Zeyrek, for her suggestions and comments from the beginning till the end of the thesis work. I am also thankful to Assist. Prof. Dr. Cem Bozşahin for the Computational Linguistics course that gave me the confidence for the exploration of the computational side of the cognitive science, and Prof. Dr. Wolf König, without whom I would not have applied to the department of Cognitive Science. I offer special thanks to my friend Nilüfer Özyörük with whom we have similar research interests, and from discussions with whom I have benefited a lot.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Language is a skill that every person possesses. Most people use it without even thinking about its nature. Linguists in an attempt to explain the structure and properties of this innate ability have adopted different approaches and methods for this purpose. They developed a range of explanations varying between grammars using mathematical formulas and abstract devices to ones using almost nothing.

The acceptance of any of one of these explanations depends on the basic assumptions. If we accept that language ability is not special with respect to general cognition but works similar to other cognitive functions, the linguistic theory has to be consistent with what is already known about human cognition. Langacker (1987), who holds this perspective, suggests a theory of language that gives an account of human linguistic ability based entirely on general cognitive abilities, known as *Cognitive Grammar*.

If we adopt the idea that cognition is a kind of computation, and that cognitive tasks are performed by computing, then cognition could be explained as "a dynamic unfolding of computational processes" (Lewis, 1999a, para. 3). Thus, the implementation of a computational model is important for determining whether the theory is capable of accounting for experimental data (Davis, 2000). Most of the theories of language have received attention from computational linguists: there are language processing models and parsers implemented for them. On the other hand, because of its general complexity and open-endedness, only few attempted to implement Langacker's theory.

In "Implementing Cognitive Semantics" Holmqvist (1993) gives guidelines on how to implement Cognitive Grammar. Later, in "Conceptual Engineering" (1998) he describes the project based on the model developed in 1993. Another work inspired by Cognitive Grammar is "Computational Cognitive Linguistics" by Heinze (1994) who also tried to apply ideas from Cognitive Grammar to the field of Computational Linguistics. To my knowledge, neither of these attempts resulted in a full computational implementation of Langacker's theory.

Despite the difficulties of computational implementation, the theory is very appealing computationally since it suggests the treatment of meaning in a uniform manner with a small set of processes, and meaning in Cognitive Grammar is far beyond the logic used widely as a semantic representation. According to Langacker meaning is encyclopedic and includes almost everything known about an entity: its spatial form, color, activities it participates in, its past and future, and so forth.

Implementing Cognitive Grammar is a difficult task, since it requires the implementation of almost all cognitive systems. Although it is possible to implement just one part of language faculty, phonology, for instance, such a model will not reveal much about language. On the other hand, computational modeling of all human cognitive functions is not possible yet; although, it is the only way to achieve the ideal model, since there is no agreement upon where language ends and where thought begins, whether meaning is part of language or not. It seems premature to implement the language without first providing mechanisms to accumulate and organize perceptual and other forms of experience.

Cognitive architectures, such as ACT-R (Anderson & Lebiere, 1998a), are theories trying to give a unified account of mind, and explain how all components of mind are working together to result in cognition. ACT-R provides a model that integrates most cognitive processes, and it is a theory consistent with findings in cognitive psychology. At the same time, it provides a powerful tool with most of what is needed for an implementation of Langacker's theory. Since ACT-R claims psychological reality, and Cognitive Grammar requires general cognitive abilities, an ACT-R model of Cognitive Grammar will reflect strong and weak points of the both theories with respect to each other. This thesis is conceived as a starting point to realize this.

The ACT-R modeling tool is used for modeling different cognitive tasks. ACT-R modelers have addressed the language faculty along with other psychological phenomena like attention and memory. The models of language processing implemented in ACT-R vary from parsing and sentence memory to language acquisition and metaphor. They all have addressed the language phenomena from different perspectives and have been guided by some theory of grammar. Unfortunately, Cognitive Grammar is not one of them, although some models have certain overlapping aspects.

Although Holmqvist (1993, 1998) and Heinze (1994) have already tried to construct a computational implementations based on the theory, they both have neglected one important aspect – the theory has to be addressed within the context of general human cognitive abilities. Their aim was to provide computational linguistics with a new form of knowledge representation, whereas it is impossible to make significant implementation without first modeling all the cognitive systems in the theory, as Holmqvist has correctly noted.

1.1 Hypothesis, Scope and Methodology

It seems that the structures provided by ACT-R for the representation of cognitive abilities seem to meet the requirements of Cognitive Grammar. The hypothesis of this thesis is to show that this is indeed so by developing a computational model for a sentence verification task in the experiment of Anderson (1974) by modeling Cognitive Grammar primitives on the ACT-R architecture.

Although Langacker (1987, 1991) in his theory tries to account for almost all linguistic phenomena, this thesis is limited to some portions of it. Cognitive Grammar deals with both phonology and semantics equally, but this thesis is primarily concerned with semantics. The scope of this thesis is limited to the design of semantic representation and processing that will meet the requirements of Cognitive Grammar and ACT-R, and that are sufficient to model the sentence verification task in the experiment of Anderson (1974).

The importance of cognitive science is in the fact that a researcher can address the problem not just from the stand point of linguistics or psychology, but as a whole, if he or she is equally aware of the achievements in the relevant disciplines. In order to account for the linguistic phenomena, it is necessary for the linguistic theory to be psychologically plausible. Furthermore, it is important to be able to construct a computational model without ruining the linguistic theory and losing its psychological plausibility. Cognitive modeling provides the computational means to hypothesize about mental structures and processes while retaining psychological plausibility. The modeling will be done by mapping Cognitive Grammar into the ACT-R architecture and putting much emphasis on not violating their assumptions.

Cognitive architectures, like ACT-R, do not propose a language specific module and, yet, they were successfully applied to language processing tasks. The models designed on these architectures overlap with ideas from Cognitive Linguistics (e.g. in terms of parallel semantic and syntactic processing (Jackendoff, 2002)). However, modelers are usually unaware of this. The field of linguistics was for a long period of time dominated by formal theories of grammar; consequently, it is expected that modelers will direct their attention to popular and computationally easy theories first. To direct the attention of ACT-R modelers (who are mostly psychologists) to Cognitive Linguistics is another aim of this thesis.

1.2 Organization

This thesis has three major parts: linguistic theory, cognitive architecture used for modeling and the model itself. Each is presented in a separate chapter where the preliminary theoretical background is followed by the presentation of the model. The organization and contents of the chapters are outlined below.

Chapter 2 presents Langacker's theory, i.e. Cognitive Grammar, in the framework of Cognitive Linguistics. Basics of the theory are presented first. They include general human cognitive abilities, lexical meaning and grammatical extensions of these cognitive abilities. Then, the implications of these cognitive abilities on meaning and structure of clauses are discussed. After that, the discussion of previous implementation attempts of the theory made by Holmqvist (1993, 1999) and Heinze (1994) will be presented.

The ACT-R cognitive architecture that was selected as an implementation platform for Cognitive Grammar is presented in Chapter 3. It is discussed with respect to its architectural assumptions (the symbolic level), the connectionist aspects of the theory (the subsymbolic level), and the way knowledge is represented and processed with respect to these. Since there are also other language models implemented on the architecture, they are discussed here with respect to their meaning representations and relevance to Cognitive Linguistics. The aim of this chapter is to give the general idea about the ACT-R theory and state the reasons why it was selected as an implementation platform among other architectures.

Chapter 4 presents the model of the sentence verification task of the experiment of Anderson (1974). The description of the experiment is followed by the discussion of other ACT-R models of the same task. The next section presents the model developed in this thesis. It is discussed with respect to the structures and processes that are required for the task from the standpoint of Cognitive Grammar. The last section of Chapter 4 discusses the performance of the model in terms of speed and accuracy. The performance is evaluated with respect to the real data obtained from the experiment by Anderson (1974) and the performance of other models of the same task, which are Anderson, Budiu, and Reder (2001) and Lebiere (2002).

The concluding Chapter 5 summarizes the major contributions of this thesis and the possibilities of expanding the model in the future.

CHAPTER 2

Cognitive Grammar as an Approach to Cognitive Linguistics

In the guidelines of articles on the website of *Cognitive Linguistics*, an interdisciplinary journal of Cognitive Science, it is stated that Cognitive Linguistics investigates the interaction between language and cognition. The research is focused on:

- the structural characteristics of natural language categorization (such as prototypicality, cognitive models, metaphor, and imagery)
- functional principles of linguistic organization (such as iconicity)
- the conceptual interface between syntax and semantics
- the relationship between language and thought, including matters of universality and language specificity
- the experiential background of language-in-use, including the cultural background, the discourse context, and the psychological environment of linguistic performance

(Cognitive Linguistics, 2004, para. 2)

Cognitive Linguistics is a movement within the functionalist approach to linguistics unifying studies that try to explain the structure and dynamics of language from a cognitive perspective (Redeker & Janssen, 1999). The functionalist approach to linguistics is best identified as a belief that language structure depends on functions it serves and other factors such as environmental, biological, psychological, developmental, historical and sociocultural (Langacker, 1999b). Consequently, any description of language from the cognitive perspective has to be psychologically plausible and give an account of linguistic phenomena with respect to general human cognitive abilities. The processes used in language comprehension have to be used for other cognitive tasks as well, since it is believed that there is nothing special about language with respect to other complex cognitive tasks like problem solving.

Modern linguistics beginning with Chomsky (1957, 1965) has been in a quest to establish a limited set of rules for generation of grammatically correct and semantically acceptable expressions. Since the grammar device producing these expressions is also a mental phenomenon, the formal (generative) linguistics might also be referred to as *cognitive*, but there are major differences between the approaches (Ungerer & Schmid, 1996). The term *cognitive linguistics* can be applied to any linguistic theory that accepts that language is a mental phenomenon; consequently, much of the modern linguistics is cognitive. For the theory in order to be a part of the Cognitive Linguistic movement, it has to see language as integrated in general cognition (i.e. accept that there is nothing special about the language with respect to other cognitive abilities like reasoning and problem solving).

Cognitive Grammar (Langacker, 1987, 1991, 1999a) is one of the approaches to Cognitive Linguistics. The central claims of the theory are:

1. Semantic structure is not universal; it is language specific to a considerable degree. Further, semantic structure is based on conventional imagery and is characterized relative to knowledge structures.
2. Grammar (or syntax) does not constitute an autonomous formal level of representation. Instead, grammar is symbolic in nature, consisting in the conventional symbolization of semantic structure.
3. There is no meaningful distinction between grammar and lexicon. Lexicon, morphology and syntax form a continuum of symbolic structures, which differ along various parameters but can be divided into separate components only arbitrarily.

(Langacker, 1987, pp. 2-3)

The theory places much emphasis on the need to characterize the language and linguistic structures within the data known about human cognition in general. The grammar of a language is held the same with certain cognitive abilities. Consequently, an account of language and grammar cannot be given without the description of human cognitive abilities.

The model developed in this thesis is based on Cognitive Grammar. The reasons for selecting this theory is that it views language as an integral part of human cognition and tries to explain its structure in term of general cognitive abilities; and that it received so little attention from computational modelers despite its being a "powerful full-scale model of language" (Holmqvist, 1998, p. 153).

The following sections of this chapter form a short introduction to Cognitive Grammar, which from place to place is extended to include the relevant ideas from cognitive semantics in general. Section 2.1 presents basics of the theory: general human cognitive abilities that Langacker claims to be fundamental for linguistic ability, the nature of lexical meaning and their grammatical reflections. Section 2.2 presents implications of these cognitive abilities on the structure of clause and grammatical relations within a clause. The concluding Section 2.3 is concerned with previous computational implementations of Cognitive Grammar.

2.1 Basics of Cognitive Grammar

According to Langacker (1987, p. 65) "the grammar of a language is simply an inventory of linguistic units". Linguistic units available to speakers of a language are *phonological*, *conceptual*, and *symbolic* structures. Phonological structures are symbolizations of conceptual structures, and their association constitutes a symbolic structure. These symbolic relations are means for analysis of linguistic expressions and they are the main concern of Cognitive Grammar.

Another claim of the theory is that “there is no meaningful distinction between grammar and lexicon” and that “lexicon, morphology and syntax form a continuum of symbolic structures, which differ along various parameters but can be divided into separate components only arbitrarily” (Langacker, 1987, p. 3). Consequently, lexicon (if we take the term to mean this continuum) is the proper starting point for the introduction to Cognitive Grammar.

A lexical item is the “commonality in form and meaning observable across a substantial number of *usage events* (i.e. actual utterances in their full phonetic detail and contextual understanding)” (Langacker, 1999a, p. 2). Lexical items are acquired through *reinforcement* of this commonality (i.e. the *entrenchment* of common features). The process of acquisition involves *decontextualization* (i.e. filtering out of non-recurrent features) and *schematization* (i.e. abstraction from specific details). Since usage events are instances of social interaction, lexical items are mostly experiences shared by the community. Furthermore, the acquisition of lexical items reflects basic cognitive abilities that organize and store these experiences (Langacker, 1999a).

2.1.1 Cognitive Abilities

Natural languages, as seen by Cognitive Linguists, are based on the way humans experience the world. Consequently, it is natural to presuppose that the language is shaped by the environment. However, humans have limited perceptual and motor abilities; thus, the range of possible experiences is limited to the ones allowed by the human perceptual-motor system. Furthermore, organization and storage of these experiences are constrained by cognitive abilities other than the perceptual-motor ones, and these abilities also affect the language. This subsection will deal with general human cognitive abilities that are important for language comprehension.

Humans have an inborn capacity only for certain kinds of experience. This capacity is determined by the limitations of the perceptual-motor system. Human beings have only five senses, and the experience of the world is limited to what they see, hear, taste, smell and touch. Besides these we experience inner feelings: mental and bodily sensations such as emotions, pains, pleasures, and so forth. All these realms of basic experiences and the conceptions of time and space are collectively referred to as *basic domains* by Langacker (1987, 1991, 1999a).

These basic domains determine human conception of the world, and function as a background for the characterization of specific concepts. These basic domains are experienced by means of other cognitive abilities, such as the ability to *compare* two experiences and detect either their difference or their identity, the ability to use one experience for the *categorization* of the other, the ability to *abstract* experiences and conceive them differently with respect to the level of specificity and detail, the ability to direct and focus the *attention* and perceive scenes in terms of *figure/ground* organization; which are responsible for the emergence and organization of these specific concepts (Langacker, 1999a).

There are other abilities considered by Langacker to be fundamental to linguistic semantics. One of them is the ability to *establish relationships* (i.e. conceive objects or events in connection) for different purposes like comparison. Another one is the ability to *group* entities (events) together for the purposes of manipulating them as a unitary entity (Langacker, 1999a). The grouping could be performed on any kind of entities and done based on any commonality. For example, groups like forest, a set of metal objects, and furniture are brought together because of different reasons such as proximity, physical or functional properties. *Mental scanning* is another cognitive ability which is used for the conceptualization of a complex structure. There are two kinds of scanning – *sequential* and *summary*. In sequential scanning one configuration is transformed into another (e.g. the perception of a moving object), and in summary scanning every event adds something to "a single configuration, all facets of which are conceived as coexistent and simultaneously available" (Langacker, 1987, p. 145) (e.g. the perception of a path of motion).

Metaphor and *image schemas* (Lakoff, 1987; Johnson, 1987) are also considered by Langacker (1987, 1999a) to be essential to language. Image schemas are "highly abstract conceptions, primarily configurational, which are grounded in everyday bodily experience and play an essential role in structuring our world" (Langacker, 1999a, p. 3). The universal image schemas include source-path-goal, contact, container, balance, part-whole, and so forth (Johnson, 1987). Lakoff and Johnson (1980) and Lakoff (1987) suggested that an *abstract domain* (i.e. cognitive realm like politics) cannot be understood on its own; that it requires analogy from some basic domain (they are referred to as *target domain* and *source domain* respectively). For example, the domain of thinking uses verbs that originally belong to the domain of spatial motion (e.g. *arrive at conclusion*, *keep up with somebody*). Setting correspondences between elements of two domains is referred to as a metaphor. Lakoff and Johnson (1980) call a correspondence between two domains a *conceptual metaphor*. Conceptual metaphors are very conventional; most people use them without seeing as such. The great deal of fixed expressions like *come to a conclusion* was acquired as it is (i.e. the abstract domain of thinking was structured like the domain of motion from the beginning, i.e. without mapping the domains first). Image schemas are not specific to any domain, but in a conceptual metaphor the target domain inherits the image schemas of the source domain.

The ACT-R cognitive architecture, which is presented in Chapter 3, provides the basic cognitive abilities that are required for language processing. The abilities like entrenchment, attention, and comparison are already integrated in the architecture. The other abilities like abstraction, grouping, and conceptual metaphor are mostly necessary for the acquisition of language and the creation of novel structures. They can be modeled on ACT-R, but since the model developed in this thesis is limited to the structures necessary for the sentence verification task, which does not require acquisition of novel structures, but rather the categorization of sentences by existing ones, the model developed in this thesis does not explore the possibility of implementation of these cognitive abilities.

2.1.2 Lexical meaning

In Cognitive Grammar lexical items do not have a single meaning, but rather a number of related senses that form a *complex category*, a network consisting of *categorizing relationships*. There are two kinds of categorizing relationships – *instantiation* and *extension*, and different meanings of a lexical item are either instantiations of the central value (*prototype*), or extensions from it (Langacker, 1999a).

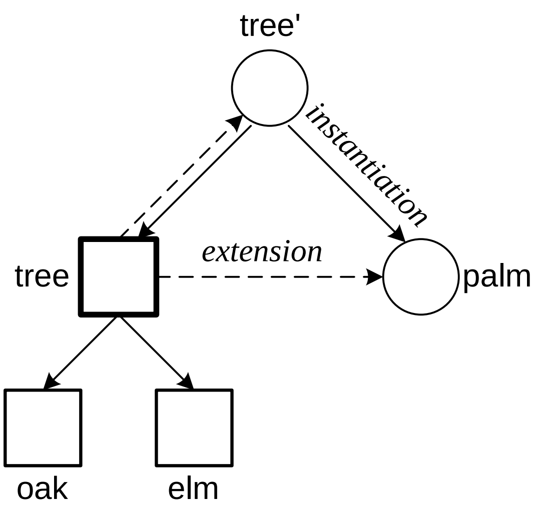


Figure 2.1: Extension and instantiation relations in a hierarchy

For example, consider the conception of *tree*, which is schematic with respect to *oak* and *elm*, and *oak* and *elm* are instantiations of *tree* (i.e. *tree* is a superordinate for them). To add the conception of *palm* into the *tree* hierarchy of a person who is familiar only with trees like *elm* and *oak* will require the *extension* operation, rather than *instantiation*, because *palm* violates some features of the *tree* schema (i.e. it cannot be directly categorized as an instance of *tree* category because it looks different than the trees the person is used to). In this case the *tree* schema is abstracted further to result in a schema *tree'*, such that *palm* and *tree* will both be unproblematic instantiations of the *tree'* schema, like shown in the Figure 2.1, where squares represent well established schemas, and circles represent novel structures.

According to Cognitive Grammar the meaning of a lexical item is encyclopedic, that is, any knowledge about the entity is the part of its meaning. Consequently, a lexical item does not have a single semantic representation, but rather it is a point of access to many other concepts and knowledge systems. These concepts and knowledge systems together with basic domains, which were discussed in the previous section, are referred to as *cognitive domains* by Langacker (1999a).

Thus, a lexical item "evokes a set of cognitive domains as the basis of its meaning, and exhibits considerable flexibility in this regard" (Langacker, 1999a, p. 4). The domains activated by a lexical item are those in which "the entity it *designates* (i.e. its conceptual referent) figures directly" (Langacker, 1999a, p. 4). Moreover, a lexical item *ranks* these domains with respect to their *centrality* (i.e. the importance of the domain for the characterization of the designated entity). This ranking of the domains is also part of the meaning, since lexical items that activate the same domains can differ with respect to their ranking.

The domains activated by a lexical item constitute its conceptual *content*. However, the meaning of a lexical item is not just this content, but also in the *construal* of this content, since the content can be construed differently and result in different meanings. The construal depends on the cognitive abilities *specificity*, *background*, *perspective*, *scope*, and *prominence*.

Specificity is the ability to conceive the same entity with different levels of detail. For example, consider the taxonomy ordered from the more specific to the less specific: *English Shire* → *heavy horse* → *horse* → *mammal* → *animal*. Such a hierarchy reflects the process of schematization, since each following item is schematic with respect to the previous item. Although the list is not all inclusive, the word *horse* would be selected as an answer to the question "What is this?" accompanied by the pointing gesture to a grazing horse by the great majority. If we think of the possibilities as levels arranged from more general to more specific, the level of *horse* would be the most salient level, because it is the level used to call things (Taylor, 2002), and it is called the *basic level*. Entities at this level are "maximally contrastive and maximally informative" (Taylor, 2002, p. 132), that is, the features people name as a characteristic of an entity at this level tend not to be shared by other entities at this level, and the number of named features is usually high with respect to other levels (Rosch, 1975 in Taylor, 2002, p. 131). Concepts at higher levels are very general and applicable to a broader range of entities, and concepts at lower levels share too many features among themselves and the amount of information added to the basic level is not much (Taylor, 2002).

The ability to use one entity as a *background* for understanding another entity can be seen in the process of categorization, in which one entity is used as a background (*standard*) for the evaluation of another entity (*target*). In order to categorize an entity, it has to be compared to some other entity, either reactivated from the long-term memory or already present in short-term memory, and since this entity constitutes our prior knowledge, it is the background for categorization of other entities.

The same scene (situation) might be perceived differently because of cognitive orientation. *Perspective*, a way of seeing a situation, depends on the viewer's viewpoint, his primary focus of attention and the relation he establishes between the scene and himself. Viewpoint depends on a vantage point, position of the viewer with respect to the perceived scene, and an orientation, alignment of the viewer with respect to the visual field. From a single vantage only a certain portion of the situation are available for perception, but this portion could be perceived differently with respect to orientation (e.g. we can see the same scene normally or upside-down). People are used to perceive some objects from certain viewpoints, and the identification of these objects is done best from that point of view (Langacker, 1987, 1999a).

Vantage point determines the division of the scene into foreground and background. Objects on the foreground are perceived easier because they are closer to the viewer. Focus of attention is usually selected from the foreground, unless the attention is directed consciously. Moving objects on the visual field, or just objects that are different from the rest, draw our attention, and they are located not necessarily on the foreground. Selection of some entity as a focus of attention divides the scene into figure and ground. Figure is the focus of viewer's attention and not necessarily the most distinct entity on the scene, and ground is the rest of the scene. Choice of the figure determines the structuring of the scene.

According to Langacker (1987) a viewer, by the selection of viewpoint and the focus of attention "establishes a conceptual relationship between himself and the scene structured". The scene could be structured *subjectively* or *objectively* with respect to whether the viewer has included himself in the conceptualization. The ability to mentally rotate the object, i.e. virtually change the viewpoint, allows us to structure the scene differently from a single perspective. Consequently, we have the ability to take someone's perspective and give directions or descriptions accordingly.

Scope of an expression is the conceptual content it invokes (i.e. activated set of cognitive domains). For example, in a partonymy like *knuckle* → *finger* → *hand* → *arm* each following item functions as the scope for the preceding one. Thus, *finger* is the scope for *knuckle*.

Ranking of cognitive domains by a lexical item and figure/ground organization are examples of *prominence*. Another example of prominence is the fact that from the activated conceptual content (i.e. scope), "every expression selects some entity for designation", that is, "an expression imposes a particular *profile* on the conceptual *base*" (Langacker, 1999a, p. 7). For example, *knuckle* activates the conception of *finger*, and it profiles a certain subpart within it.

"An optimal description of language structure requires a notion of conceptual reference in which not just thing-like entities but also *relationships* are capable of being profiled" (Langacker, 1999a, p. 7). "A relationship generally has one or more *focal elements*" (i.e. *participants*) that are prominent within its profile (Langacker, 1999a, p. 8). The *primary figure* (i.e. the most salient element) within the profile of a relationship is called *trajector*, and the *secondary figure* (if there is one) is called *landmark.* The selection and organization of primary and secondary focal elements is the matter of figure/ground organization (which is primarily related to vision), and it is an important aspect of meaning. Expressions profiling relationships, similar to things, can activate the same conceptual content (i.e. cognitive domains), and can differ in meaning with respect to trajector/landmark organization. For example, the organization is responsible for the different phrases like *above* and *below*. The configuration they profile is the same, but selected trajectors are different. For example, we could say "the picture is *above* the clock" and "the clock is *below* the picture". If we would visualize these sentences, the images would be the same, but our primary focuses are different. In first sentence, trajector is *the picture*, and in the second it is *the clock.*

The lexicon in the model developed in this thesis was intended to reflect these notions as closely as possible. It makes use of the notions of profile, base and domain. The elements of construal such as trajector/landmark organization are parts of lexical meaning. However, since the lexicon and the complexity of lexical items are limited to the words used by Anderson (1974) for his experiment, not all the cognitive domains for lexical items were implemented. The conceptual base is limited to a single cognitive domain, which is assumed to be primary (i.e. most central). However, Chapter 4 explores the possibilities for the implementation of all aspects of meaning, and the structure of the lexicon designed in the model reflects these aspects (i.e. with the richer lexicon and the identification of cognitive domains of a lexical item all aspects of meaning would be reflected).

2.1.3 Lexical classes

Instead of using traditional lexical classes such as noun and verb, Langacker (1987, 1991, 1999a) proposes a semantics based classification, which results from the notion of profile. *Entity* is the term used to stand for everything. If an entity profiles a region in some domain, it is a *thing* which includes all abstract and concrete objects and substances. If the region is bounded, it is an object and if it is not, it is a substance. Things are represented by nouns, and the distinction between count and mass nouns depends on the boundedness of the profiled region.

The counterpart to a thing is a *relation*[1] which profiles the interconnection between entities in some domain. An entity is either a thing or a relation. Relations are further divided into temporal and atemporal. Temporal relations are *processes*, which constitute the traditional verb category. Atemporal relations, on the other hand, fall into two categories – stative (simple) and complex. Stative atemporal relations are perceived without reference to time, whereas complex atemporal relations are concerned with it. For example, *red* (adjective) is a stative atemporal relation and *across* is a complex atemporal relation.

[1] Langacker uses terms *relation* and *relationship* interchangeably.

Since both processes and complex atemporal relations are concerned with time the difference is based on the scanning operation deployed (see Section 2.1.1). Processes are scanned sequentially, and complex atemporal relations are the result of summary scanning. Consequently, the primary concern of process is evolution through time and the primary concern of complex atemporal relation is the result of that evolution. For example, note the difference between the process of *breaking* a glass and *broken* glass, in the first case we are concerned with action, and in the second with the result.

The traditional classes adjective, adverb and preposition fall into these categories and are divided with respect to the entities they interconnect. Adjectives are stative atemporal relations that have only one focal element which is a *thing*. Adverbs, on the other hand, have a *relation* as a focal element. Prepositions interconnect more than one (usually two) things or relations, and their landmark is generally a thing.

According to Langacker (1987, 1991, 1999a) a lexical class represents a complex category (i.e. a network) centered on a prototype that emerged through the process of schematization. The prototype is a *conceptual archetype*, since "it embodies a recurrent commonality in our everyday experience" (Langacker, 1999a, p.9). Archetypes initially emerge for concrete experiences, and then they are extended to abstract domains. For example, the prototype of the noun category is *physical object*, which, when further schematized to include abstract nouns, becomes a region in some domain (i.e. thing). The prototype for verb category is *energy transfer*, that is, "an event in which an agent does something to a patient" (Langacker, 1999a, p.10).

2.1.4 Symbolic complexity

Linguistic expressions are *symbolically complex*, which means that they contain smaller symbolic elements. For example, consider lexical items arranged from symbolically less complex to more complex: *sharp* → *sharpen* → *sharpener* → *pencil sharpener* → *electric pencil sharpener*. Lexical items that are the least complex are morphemes.

A lexical item can be *analyzed* with respect to the *component structures* a symbolically complex expression (called *construction* or *composite structure*) contains. *Analyzability* is the matter of degree, that is, it is not always possible to identify component structures of a lexical item. For example, consider the word *computer*; it can be analyzed as having two component structures – *compute* and the suffix *–er* – which allows to understand it as 'something that computes'. However, *computer* is more than just 'something that computes', the expression refers to a particular object that cannot be inferred from meanings of *compute* and *–er*. Thus, *computer* exhibits partial analyzability. On the other hand, full analyzability is observed in expressions like *maker*, the meaning of which can be understood from its component parts *make* and *–er*. The meaning of a symbolically complex expression is usually more specific than the collective meaning of its component parts. Thus, Langacker (1999a, p. 16) says that "complex expressions exhibit only *partial compositionality*", and "rather than *constituting* a composite structure, the component structures *correspond* to certain facets of it".

An expression that has several parts groups them in a particular manner, called *constituency*. In Cognitive Grammar constituency is "a special kind of symbolic assembly" in which "the composite structure of one construction functions as a component structure with respect to another construction" (Langacker, 1999a, p.14). Besides the correspondence relationship between the composite and component structures, there is also a categorization relationship. Usually, one component structure elaborates a substructure of another component structure, and the elaborated subpart (called *elaboration site*) is usually schematic (i.e. has less detail) with respect to the structure that elaborates it. In a composite structure, one of its components is usually schematic with respect to it, and the composite inherits the profile of this component, but also becomes specified in a greater detail by other component structures. Such a component structure is called the construction's *profile determinant*.

2.1.5 Extension to grammar

Cognitive Grammar claims that there is no distinction between lexicon and grammar. Grammar rules are "schematizations of symbolically complex expressions, or constructions, and can thus be described as *constructional schemas*", which reflect the symbolic complexity and the commonality of these expressions (Langacker, 1999a, p. 19). Similar to lexical items, constructional schemas form complex categories (i.e. a network constructed by categorization relationships by instantiation or extension from a prototype). There are low-level constructional schemas (e.g. *X lid* for compounds like *jar lid* and *coffin lid*), as well as high level constructional schemas (e.g. *X Y* for any kind of two place compounds), and low-level schemas are instantiations of high-level schemas. These constructional schemas are used for construction and categorization of novel linguistic expressions (i.e. not lexical items).

Notions of *profile determinacy* and *elaboration site* are used for the definition of traditionally grammatical notions. For example, a *head* of a phrase is "the profile determinant at a given level of organization", a *complement* is "a component structure that elaborates a salient substructure of the head", whereas a *modifier* is a component structure, "a salient substructure of which is elaborated by the head" (Langacker, 1999a, p. 21).

In Cognitive Grammar, other grammatical notions like *noun phrase* and *finite clause* are also defined through the same notions. Recall that nouns profile things, and verbs profile processes, which are conceptual archetypes. *Nominals* (a term used by Langacker (1987, 1991, 1999a) to refer to noun phrases) also profile things. Whereas nouns specify a *type* of thing, nominals specify an *instance* of that type. For example, in the phrase *the apple*, *apple* is the name of the category, and the phrase itself refers to a particular instance of that category. Langacker (1987, 1991, 1999a) uses the notion of *ground* (which is a speech event, its participants and the current discourse) to further define nominals and finite clauses. Nominals and finite clauses are always *grounded*, that is, have reference to the ground. Determiners, with respect to nominals, serve the grounding function, that is, they help the speaker and the hearer to establish reference with particular instance in the ground.

The relationship between verbs and finite clauses is exactly the same with the relationship between nouns and nominals, that is, verbs specify a type of process, whereas finite clauses specify a particular instance of that type. Finite clauses, like nominals, are grounded in the discourse. The elements that serve grounding function in English are tense inflections and modals. For example, consider the sentence *the man opened the door*, the verb *open* specifies (or profiles) a type of process that has two participants, the sentence, on the other hand, profiles a particular instance of that type and elaborates its participants.

Fauconnier (1985) proposes a notion of mental space useful for meaning construction. Since in cognitive semantics concepts refer to entities in the mind (conceptualizations), not to some objects in the real world, mental space is populated with these mental entities and with relations among them; in other words, mental space is where nominals and finite clauses are grounded. The idea of mental spaces is similar to the mental models theory (Johnson-Laird, 1983). In both theories the meaning of an expression is constructed as a set of populating entities and relations, and in both there is a possibility of several such sets existing simultaneously. To my knowledge, the difference between two theories is that in the mental models theory there are some implicit models together with explicit[2] ones and the models can be tagged for probability and other properties. In the mental space theory, as far as I know, there is no tagging and no implicit models, instead, mental spaces are structured with respect to the way they are conceptualized and all of them are explicit.

The notion of mental space is used in the model of this thesis to represent discourse. It is assumed that language users construct such mental spaces by default, for example, to keep all the relevant characters in the story accessible. The usage of mental spaces in sentence processing in discussed in Chapter 4.

2.2 Clause Structure

In Cognitive Grammar, “the meanings of grammatical constructs”, like lexical items “represent complex categories” (i.e. networks formed by relationships of instantiation and extension) (Langacker, 1999a, p. 23). Notions like grammatical

[2] Johnson-Laird is primarily concerned with reasoning, and his models are possible interpretations of the premises. Some possibilities may be left undetected and they are called implicit, whereas detected ones are called explicit.

relations, voice, transitivity, and so forth are understood through *conceptual archetypes* "that define prototypical values of grammatical elements" (Langacker, 1999a, p. 23).

2.2.1 Conceptual Archetypes

The way people conceptualize the world is crucial for the language. First of all, it is essential for concept formation and determines the scope of possible relations between entities. Moreover, grammatical relations are based on these conceptualizations as well.

In very basic terms, the world is seen as a space populated with physical objects interacting with each other. The most fundamental elements of the world are space, time, substance and energy. These elements give us conceptual archetypes – *thing* and *process* – that are highly schematic. These alone and their combinations form the conceptions of a physical object occupying a location in space and moving through space, the physical object being in some state or undergoing some change (Langacker, 1987, 1999a).

Langacker (1987, 1999a) proposes a "billiard-ball model" of world that could be summarized as consisting of physical objects moving in space and impacting other objects that react to these impacts in some way because of the energy transmitted during the impact. Fillmore (1968 in Taylor, 2002, p. 421) and Langacker (1991) has suggested the "action-chain model" as a prototype of energy transfer. The idea of one object initiating a transmission of energy through contact with another object, which might also make contact with another object and transfer energy further to it, or be the final element in the action chain. The other model proposed by Langacker (1991, 1999a) as a prototype is the "stage model" that reflects perceptual experience. Together with the "action-chain model" they form a "canonical event model" (Langacker, 1991, 1999a), which is the observation of a prototypical action.

The system of force dynamics introduced by Talmy (1988, 2000) is related to the "action-chain model" because it specifies the way entities interact with each other in a process. The author introduces notions of Agonist and Antagonist, two entities involved in a process. Agonist has the tendency either to preserve its present state or to change it. Antagonist, on the other hand, can transmit energy to Agonist, which can either resist the energy force or be affected if the force overcomes its natural tendency (Talmy, 1988, 2000). The system of force dynamics is a domain for characterization of concepts like *keep*, *let*, *remain*, *enable*, and modals *can*, *must*, *should*, and so forth, because they presuppose interaction of two forces. For example, in the sentence *X let Y fall* (e.g. I let the vase fall) presupposes that X is more powerful than Y and that X fails to stop Y from falling.

The "canonical event model" is the ground for the definition of *role archetypes*, such as *agent* and *patient*. An agent is an initiator of activity, i.e. energy source, and a patient is an object that undergoes change (Langacker, 1991, 1999a). The model is used for the definition of other linguistic notions as well. A finite clause "profiles a process construed as a single event" (Langacker, 1999a, p. 25). Prototypical subjects and objects are an agent and a patient respectively. The "canonical event model" is generally coded as a transitive clause, where the verb designates agent-patient interaction, and a subject and a direct object designate an agent and a patient, respectively.

Besides the canonical event model, there are other conceptual models as well. Similarly to canonical event model, which is coded as a transitive clause, the other models have their own ways of coding. For example, the conceptual archetype of the object moving in space is coded by a clause in which "the head is an intransitive motion verb, the subject codes the mover, and a locative complement specifies the source, path, or goal of the motion" like in the sentence *I went to the school* (Langacker, 1999a, p. 25). According to Cognitive Grammar, a language "exhibits an array of basic clause types" developed for the coding of certain conceptual archetypes, and these clauses are then extended for the coding of different situations by means of *conceptual metaphors* (i.e. mapping of two different domains (usually structuring an abstract domain in terms of a basic domain)) (see Section 2.1.1) (Langacker, 1999a, p. 26).

When extended to include all kinds of domains, a finite clause, which according to the canonical event model was a single event of agent-patient interaction that was coded as a transitive clause, would be defined simply as a *grounded process* with two participants (in the case of a transitive clause) that have "the status of trajector and landmark, which are manifested by the subject and object nominals" (Langacker, 1999a, p. 27).

The canonical event model was used in the model developed in this thesis as a background for the categorization of transitive clauses. Conceptual archetypes – thing and process – are used for the classification of words into lexical classes. However, notions of force-dynamics were not used in the model. They are presented here with the sole purpose of providing a fuller picture about the complexity of semantic structure.

2.2.2 Grammatical relations

Since Cognitive Grammar claims that any grammatical notion (e.g. syntactic primitives) can be defined in semantic terms, Langacker (1987, 1991, 1999a) claims that notions of *subject* and *object* are universal (even for languages that make a little if any reference to these notions) and definable in terms of something more basic. Subject and object prototypes are characterized through notions of semantic role, discourse function, and prominence.

First of all, the semantic roles agent and patient are prototypical for subjects and direct objects, respectively. Second, subjects and objects are nominals (i.e. things grounded in discourse). Third, "the subject and object relations hold at the clausal level of organization, and the clause is a basic unit of discourse" (Langacker, 1999a, p. 29). Consequently, a prototypical subject is an agent and the primary clausal topic, and a prototypical direct object is a patient and the secondary clausal topic at the same time (although these roles are not universal, they are typical). Clausal topic is "a *reference point* serving to situate, organize, and interpret the specifications in an ongoing fashion" (Langacker, 1999a, p. 29). In this case, only the subject and object prototypes can be topics. However, if clausal topic is defined as a reference point only within a single clause (which according to Langacker (1999a) is also applicable to all subjects and objects), then subject and object can be defined with respect to *figure/ground organization*.

Semantic subjects and objects are complex categories, that is, networks of semantic roles that they can take. Besides the roles of agent and patient discussed in the previous section with respect to the "canonical event model", there are other archetypal roles as well, such as an *instrument*, which is an object used by an agent to affect a patient; *mover*, which is an object that changes its position with respect to the surrounding; *experiencer*, which is a being occupied with a mental activity. Langacker (1987, 1991, 1999a) uses the term *zero* to refer to the archetypal role of an entity that is viewed as not participating in the process but just occupying some space or having some property. Besides these, Langacker (1991, 1999a) allows intermediate and other archetypal roles that may emerge.

Theta roles are grouped under two groups: thematic and non-thematic. The division is based on the notions of conceptual autonomy and dependence, "a natural basis for ergative / absolutive organization" (Langacker, 1999a, p. 35). Conceptually autonomous concepts can be conceptualized on their own, without the need of other concepts; dependent concepts, on the other hand, presuppose autonomous concepts for their conceptualization (e.g. we can conceptualize a ball on its own, whereas we cannot conceptualize a knuckle without evoking the concept of finger). With respect to processes, autonomous processes can be conceptualized independently of causation, but dependent processes cannot be conceptualized as such (e.g. it is possible to imagine the opening of a door without the force that triggered the process, whereas it is impossible to imagine killing without evoking some murderer and victim). Transitive verbs that do not have intransitive variants are examples of dependent processes, and intransitives are examples of autonomous processes. Having taken an autonomous process, it is possible to add dependent elements to form a more complex process that will also be autonomous (Langacker, 1999a).

Autonomous processes (that have a single participant) are referred to as *thematic* processes. Thus, *thematic* roles are the ones participating in thematic processes; they are zero, experiencer, patient and mover. Langacker (1991, 1999a) calls the role schematic to these as *theme*. Archetypal roles agent and instrument are non-thematic since they cannot be conceptualized without the transmission of energy to some other object.

Applied to the notions of subject and object, these archetypal roles motivate the selection of a subject. According to Fillmore (1968 in Langacker, 1999a) in English a subject is either an agent or an instrument (e.g. in sentences *the man opened the door* and *the key opened the door*), or a theme, if there is no agent or instrument (e.g. *the door opened*). These roles are discussed with respect to the action chain model in Langacker (1991, 1999a), and since action chain model represents energy transfer from one physical object to another, the ordering of the roles in this model would necessarily be *agent* → *instrument* → *theme*. Consequently, the notion of subject could be defined as the head of the profiled action chain, or in broader sense (if we consider cases where there is no energy transfer like in the sentence *Janet resembles Margo*) as "the primary figure with respect to the profiled relationship (or the primary clausal topic)" (i.e. trajector), which is the matter of figure/ground organization and the selection of the primary figure is "always a matter of construal" (Langacker, 1999a, pp. 33-34). The notion of direct object, in this case, will be a tail of the profiled action chain, or in broader sense "the secondary figure with respect to the profiled relationship" (i.e. landmark) (Langacker, 1999a, p. 34).

Recall from Section 2.1.2 that lexical items that profile relationships impose certain trajector/landmark (or figure/ground) organization. Consequently, verbs that profile processes and function as clausal heads also impose certain trajector/landmark organization on a finite clause. Thus, a subject is "a nominal that elaborates the clausal trajector" and a direct object is "a nominal that elaborates the clausal landmark" (Langacker, 1999a, p. 34).

2.2.3 Marked coding

It was mentioned in Section 2.2.1 that the canonical event model, which prototypically represents energetic interaction of two objects, is usually coded as a transitive clause in English. This coding is referred to as the *unmarked coding* by Langacker (1999a). Besides this unmarked coding, languages have varieties of constructions for the different coding of situations for different purposes like focusing on a participant that generally is not prominent. In this case, other participants that usually are focal elements remain unprofiled. For example, in the sentence *the door opened*, although the verb usually profiles whole action chain, in this case the agent is not profiled. Such kind of coding is referred to as *marked coding*.

Although there is a variety of different marked codings, in this thesis only passive voice is considered, because it is the only structure besides transitive clauses that was used in the model. The way active and passive sentences are interpreted is discussed in Chapter 4. In passives, according to Langacker (1991, 1999a) there is no change in profiling, but there is a shift in focal prominence. The primary difference between sentences *the man opened the door* and *the door was opened (by the man)* is in that in the latter case, the trajector is a theme, and the tail of the action chain is coded as a subject (Langacker considers that *be opened* and *open* both profile the entire action chain). According to Langacker (1999a, p.40), passive subject is atypical, but conforms to "the most schematic categorization, namely primary figure within the profiled relationship".

2.3 Computational Implementations of Cognitive Grammar

This section presents attempts to implement Cognitive Grammar – Holmqvist (1993, 1998) and Heinze (1994). Unlike the model designed in this thesis (discussed in Chapter 4), their implementations are not concerned with human language processing capacity, rather they attempt to use the ideas from the theory for applications in Artificial Intelligence and Computational Linguistics to provide language understanding in a human like manner. Heinze (1994) presents a semantic representation called *L-Space*, which is based on the Lattice Theory, and he uses it to enhance a message understanding system. He does not attempt to reflect Cognitive Grammar as a whole. The primary concerns of his implementation are neither the development of effective knowledge representation nor conformation to Cognitive Linguistics as a whole. Consequently, his model will not be elaborated further. Unlike Heinze (1994), Holmqvist (1993, 1998) attempts to simulate the representation and the processes described in the theory as closely as possible. His work is discussed below with respect to the representations and the processing he used.

Holmqvist's (1993) model could be divided into three parts – Representations of Activated Lexical Units, Semantic Composition Processes, and Mechanisms for Valence Suggestion and the Incremental Updating of the Schema Population (Holmqvist, 1998).

A linguistic unit has two poles – semantic and phonological; the semantic pole in Holmqvist is represented by a structure intended to model the image schemas of Langacker (1987) and Lakoff (1987). The structure consists of the matrix of domains ordered by centrality values (see Section 2.1.2), list of parts ordered by their saliences (i.e. their importance to the meaning), and list of wholes ordered by their saliences. This structure is capable of expressing the meaning in the way proposed by Langacker. The use of this representation is made clear in the following paragraphs. The model presented in this thesis has a similar representation and is partially inspired by Holmqvist (1993, 1998).

The semantic composition process assumes that humans do image superimposition[3]: "individual lexical units are superimposed to form a composite structure", because "lexical units having predications in different domains can be viewed as images" (Holmqvist, 1998, p. 157-158). The computational counterpart of superimposition consists of a number of smaller processes – domain identification, predication identification, value adjustment, and part and whole accommodations. Domain identification selects domains in which all the units to be superimposed are specified. The process is responsible for contextual disambiguation and anomaly detection (i.e. if the selected domains are not central to both concepts, than there is an anomaly; for example, in a phrase *green ideas* taken literally there is no common domain). The processes of predication[4] identification and value adjustment[5] select the predications in the domains that are left after domain identification and join them into a composite, adjusting their values in related dimensions. The processes of part and whole accommodation work alike – lexical units are mapped into one another.

[3] The reason why Holmqvist chose superimposition as a process for combination of semantic units is his assumption that "image schemas are structures with largely spatial (and imaginative) form" (Holmqvist, 1993, p. 22).

[4] In the domain a schema predicates (or profiles) a certain structure, which is called a predication (or profile).

[5] The predications have to be adjusted since some concepts like *tall* and *long* are relative; consequently, the "image" has to be rotated, shrank or expanded in order to be superimposed.

Semantic and grammatical expectations are valence suggestion mechanisms in the model. The former "suggests valence relation between two lexical units if their predications coincide in one or more central domains" (Holmqvist, 1998, p. 166), the latter, which is the special kind of the former, specifies the direction (left or right) where the semantic expectation could be satisfied. For example, *walk* requires some legged creature to be its agent; *man* schema has legs as its parts, and both concepts are specified in the domain of space. Consequently, the profile of *walk* and the profile of *man* coincide in the domain of space with respect to *leg* schema. The grammatical expectation mechanism is based on Behaghel's principle, which "claims a correlation between closeness of morphemes and closeness in valence relations" (Holmqvist, 1998, p. 167).

Schema population is a sort of semantic short-term memory that contains constructed composite schemas, which are continuously evaluated to discard the ones that cannot be part of the final meaning.

The implementation seems to meet the requirements of Cognitive Grammar, but it is computationally expensive both in terms of size of used structures and their computation (e.g. schema population keeps all possible schemas).

CHAPTER 3

ACT-R as a Cognitive Architecture for Cognitive Grammar

The previous chapter discussed cognitive theories of language, particularly a theory of Langacker (1987, 1991, 1999a) known as Cognitive Grammar. One of the major points of Cognitive Grammar was that “the linguistic structure can only be understood and characterized in the context of a broader account of cognitive functioning” (Langacker, 1987, p. 64); therefore, a comprehensive description of language has to be done in the context of a complete theory of human cognition. Theories that try to explain cognition as a whole are presented in the form of cognitive architectures (Newell, 1990).

According to the Computational Theory of Mind, the most influential form of functionalism, the mind could be seen as software that runs on the brain, the organization of which makes the mind special (Block & Rey, 1998). In the context of the Computational Theory of Mind, cognitive architectures have to specify the resources of the brain and the organization of these resources to produce cognition (Pylyshyn, 1998). These resources are basic processes and subsystems that determine the overall functioning of the architecture. The basic processes are mostly cognitive abilities that were discussed in the previous chapter. Subsystems are “informationally encapsulated mechanisms" (Fodor, 2000, p. 55), also known as *modules*, that perform or control these processes.

Among scientists there is agreement neither on the number of modules and their organization, nor on the way cognitive abilities work. The absence of a unitary vision on the structure of the mind resulted in several theories of cognition such as ACT-R (Adaptive Control of Thought, Rational) (Anderson & Lebiere, 1998a; Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004), EPIC (Executive-Process Interactive Control) (Kieras & Meyer, 1998), SOAR (States, Operators, And Reasoning) (Newell, 1990; Lewis, 2001), and CAPS (Just, Carpenter, & Varma, 1999). All four architectures are symbolic production systems. To be symbolic means to be "capable of manipulating and composing symbols and symbol structures - physical patterns with associated processes that give the patterns the power to denote either external entities or other internal symbol structures" (Lewis, 1999a, para. 4). To be a production system means to control the processing by means of production rules, which are independent condition-action pairs that are executed if their conditions are met; consequently, modeling a cognitive task on these architectures is mainly providing a correct set of such production rules (Lewis, 1999a).

Cognitive modeling is not limited to symbolic systems; connectionist modeling was applied to tasks such as reading aloud and language acquisition (e.g. learning part tense, early lexicon development and acquisition of syntactic rules) (McLeod, Plunkett, & Rolls, 1998). Although connectionist cognitive modeling was successfully used to model these complex tasks, there is no neural network architecture of cognition because of some problems (e.g. storage of data) that can only be solved by the modification of the standard networks (Taatgen, 1999). The ideas from connectionist modeling, like activation, are very appealing and were borrowed by some symbolic architectures resulting in hybrid systems. ACT-R and CAPS are hybrid systems having a symbolic production system as a control mechanism for processing, and using activation as a control mechanism for declarative memory (Taatgen, 1999).

Since Cognitive Grammar is a theory close to connectionism, because Langacker (1987, 1991, 1999a) makes an extensive use of the notions of network and activation for the explanation of different linguistic phenomena like the structure of lexical items and the contextual priming; the architecture for its implementation should be hybrid. As mentioned in the previous paragraph, there are two hybrid systems: ACT-R and CAPS. CAPS does not incorporate a learning mechanism,[6] and since Langacker's theory places much emphasis on it, CAPS is not the best candidate for implementation. Moreover, it does not incorporate mechanisms for low level cognitive tasks and perceptual-motor abilities. ACT-R, on the other hand, has mechanisms for learning as well as for most of the other cognitive abilities mentioned in Chapter 2. This makes the ACT-R architecture a better platform for the implementation of cognitive linguistic inspired models of language processing.

The ACT-R architecture[7], that is, its modules and their integration, is discussed in the following section. Section 3.2 discusses the representational commitments imposed by the architecture. Subsymbolic assumptions of the architecture, that is,

[6] The learning here is used to refer to the processes of declarative and procedural knowledge acquisition as well as adjustment of specific subsymbolic values like activation that control the use of these.

[7] Note on terminology: *architecture* and *theory* are used to refer to the theoretical part of ACT-R, *system* and *modeling environment* are used to refer to the software, and *model* is used to refer to a task specific code (software) that runs on ACT-R environment.

ideas borrowed from connectionism, are discussed in Section 3.3. In Section 3.4 the ACT-R modeling environment is presented, and, finally, models of language processing already implemented on ACT-R are presented in Section 3.5.

3.1 The ACT-R Architecture

ACT-R's history begins in 1973 with the HAM theory of human memory (Anderson & Bower, 1973), which in 1976 became the declarative memory part of the ACT theory (Anderson, 1976) that added the production rule system of procedural memory to it (Anderson & Lebiere, 1998a). In 1983 the theory was extended into ACT* with the addition of "subsymbolic processing and a theory of production rule learning" (Anderson & Lebiere, 1998a, p. viii). With the addition of rational analysis[8] to the subsymbolic component and some other changes in 1993 the theory evolved into ACT-R (Anderson & Lebiere, 1998a). The current version of the theory is ACT-R 5.0 (Anderson, et al., 2004) which has the perceptual-motor component ACT-R/PM (Byrne & Anderson, 1998, 2001) that provides means of communication with the environment.

3.1.1 Modules and buffers

The basic architecture of ACT-R is made of "a set of modules, each devoted to processing a different kind of information" (Anderson et al., 2004, p. 5). Modules reflect human perceptual-motor abilities and "higher-level cognition" (Anderson et al., 2004, p. 10). The behavior of these modules is controlled by a central production system, which communicates with them through their buffers, and their content can

[8] "A rational analysis is an explanation of an aspect of human behavior based on the assumption that it is optimized somehow to the structure of the environment" (Anderson, 1991, p. 1193); with respect to ACT-R, it caused improvement of activation calculations and production learning (Anderson & Lebiere, 1998b).

be modified both by the central production system and the modules. The central-production system can make requests to the modules to either perform some action like the search of the visual field, or it can harvest a piece of information a module puts in its buffer upon these requests.

The architecture is a combination of serial and parallel processing. The modules are mostly working independently from each other, which is one of the parallel processing examples in the architectures. However, the buffers can contain only a single piece of information at a time each, and only a single production is executed at a time; these are examples of serial processing. This serial processing constraint enables the system to be always in control of the direction of computation (Anderson et al., 2004).

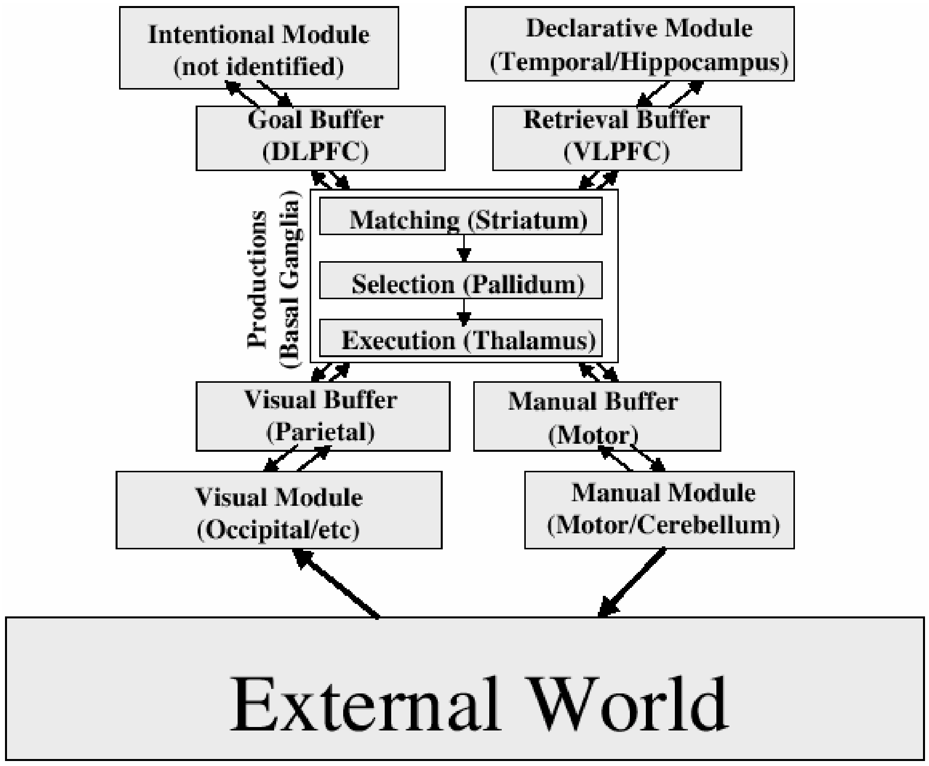

Intentional Module (not identified)
Declarative Module (Temporal/Hippocampus)
Goal Buffer (DLPFC)
Retrieval Buffer (VLPFC)
Productions (Basal Ganglia)
Matching (Striatum)
Selection (Pallidum)
Execution (Thalamus)
Visual Buffer (Parietal)
Manual Buffer (Motor)
Visual Module (Occipital/etc)
Manual Module (Motor/Cerebellum)
External World

Figure 3.1: Basic architecture of ACT-R 5.0 (Anderson et al., 2004, p. 74)

Although in the ACT-R theory the number of modules is not yet fixed, several have been implemented (Anderson et al., 2004). There are visual, audition, motor, declarative, procedural and goal modules that have been already implemented. Some existing modules, their buffers, and the brain regions roughly corresponding to them are illustrated in Figure 3.1. Although there were proposals about the existence of a syntactic module (Fodor, 1983), the lack of neural evidence about the existence of the language module kept cognitive architectures apart from implementing such a module. Even with the absence of a language specific module a number of successful language-related models were implemented. They are discussed in Section 3.5. The following two subsections discuss the central production and perceptual-motor systems.

3.1.2 The core production system

The basic premise of the ACT-R theory is that "there are two types of knowledge – declarative and procedural" (Anderson & Lebiere, 1998b, p. 5), and that cognition is the result of their interaction (Anderson et al., 2004). Declarative or explicit memory holds the knowledge of general facts and personal experience (Tulving's (1972) semantic and episodic memories respectively, with the possibility of the latter being a specialized form of the former (Tulving, 1986)). Procedural (nondeclarative or implicit) memory, on the other hand, holds the processes and skills necessary to perform actions. Another kind of memory necessary for performing actions is goal memory, which holds the information about the current intention. Humans may respond differently to the same situation and the response "depends on knowledge of what the current goal is and the ability to sustain cognition in service of that goal without any change in the external environment"

(Anderson et al., 2004, p. 15). These three memory systems together with goal and retrieval buffers constitute the core production system of ACT-R.

3.1.3 Perceptual-Motor System

The perceptual-motor extension of ACT-R was borrowed from EPIC (Kieras & Meyer, 1998), and consists of vision, motor, auditory and speech modules as shown in Figure 3.2. Although this system is only a crude approximation to the real perceptual-motor system, it provides “the basic timing behavior of the perceptual and motor systems, the output of the perceptual systems, and the input to the motor system” (Anderson et al., 2004, p. 10), useful for simulating the behavior of real subjects in psychology experiments. Since the language processing model presented in this thesis also simulates a psychology experiment, it includes parts that simulate reading and key presses. Thus, it uses vision and motor modules of the architecture.

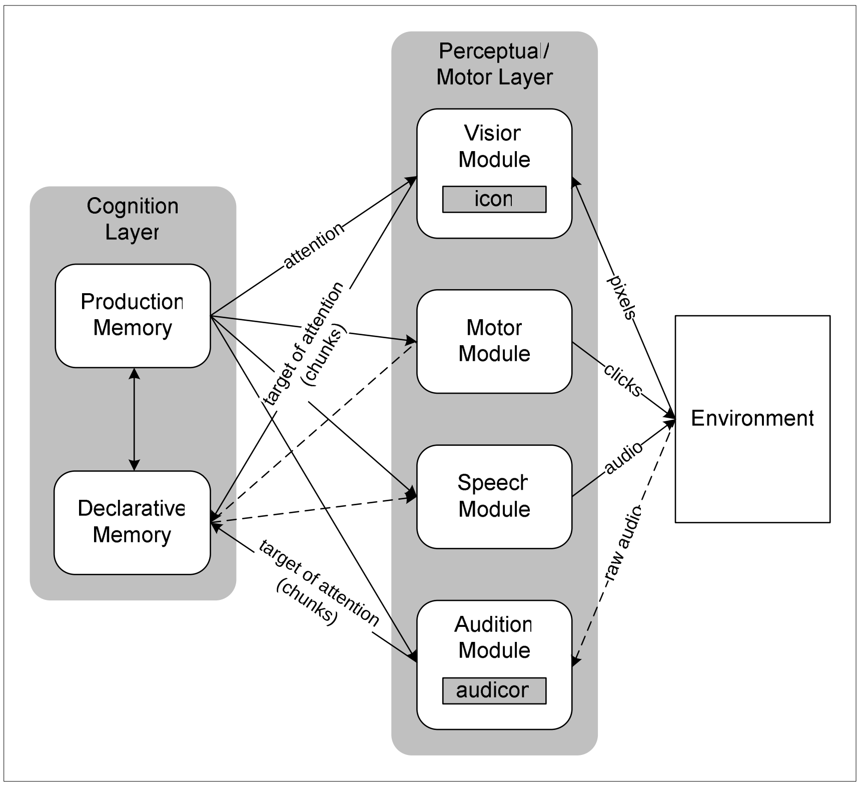


Figure 3.2: ACT-R/PM Architecture (Byrne & Anderson, 1998, p. 173)

ACT-R's perceptual-motor system is different from EPIC's with respect to the theory of the visual system: ACT-R is mostly concerned with visual attention, rather than perception (Anderson et al., 2004). The visual system of ACT-R consists of two modules: visual-location and visual object, each having its own buffer; "a visual-location module and buffer represent the dorsal 'where' system and a visual-object module and buffer represent the ventral 'what' system" (Anderson et al., 2004, p. 11). The visual system takes a window (analog to the visual field) and processes it: every object in the window is represented by a number of features such as color and location in the visicon (analog to the iconic memory (Sperling, 1960) that stores brief visual experiences before their passing into short-term memory) (Byrne, 2004). The visual-location module finds the location on the visual field (window) according to the features provided by the productions, the provided features are matched to the features stored in the visicon and the location of the object that meets the requirements is passed to the visual-location buffer. This gives the sense of where objects are and what their basic features are, but not what the objects are. In order to identify the object the system has to attend to it, this is achieved by the use of the location provided by the visual- location module. A system shifts its attention to the location specified in the visual-location buffer and puts the representation of the object into its visual buffer (Anderson et al., 2004).

The system also has the ability to track moving objects. This is achieved by keeping the system attended to some object and continuously updating the visual-location and visual buffers (Byrne, 2004). Although this ability is very limited, it could be used for the simulation of behavior in a changing environment.

The audition system is very similar to the visual system. It stores features in the audicon (analog to the echoic memory (Darwin, Turvey, & Crowder, 1972) that is an auditory counterpart of the iconic memory). The aural-location system provides the temporal location of the required feature, and the aural system identifies the sound. Both subsystems operate through buffers similar to visual system. Beside these two "when" and "what" systems, there is a proposal for the "where" system that would find the spatial location of the sound, i.e. where it comes from (Byrne, 2004). Unlike the visual system, features in the audicon cannot be attended after three second delay.

The speech module is designed to simulate the verbal responses of subjects in experiments, and it is very limited. The module has two abilities: speech and subvocalization. The strings spoken or subvocalized are placed into audicon, where they can be attended by the audition module (Byrne, 2004).

Another output mechanism of the system is the motor module. It is limited to simulating the function of hands, and the actions performed are limited to key presses and mouse clicks. Performing an action has three stages: preparation, length of which depends on the complexity of action and previous action; initiation, a 50 millisecond transition period from preparation to execution; and execution, the length of which also depends on the complexity of the action to be performed (Byrne, 2004). There are certain movement styles that determine the preparation and execution times like punching, plying and pecking.

Although the perceptual-motor system discussed here is not sufficient for the creation of structures required by cognitive semantics, such as the abstraction of commonality between perceptual experiences triggered by different instances of the same type and the creation of the corresponding schema, it is sufficient to model processes like reading and auding, the term introduced by Sticht (1978, 1979 in Crowder & Wagner, 1992, p. 112) for speech perception, quite closely to human performance. Being able to model these processes more precisely enables a modeler to test other processes underlying language comprehension more accurately.

The ACT-R community has developed a theory of how these buffers and modules interact to result in cognition. The modules already implemented in ACT-R were mapped into regions of the human brain (see Figure 3.1) to allow modeling of fMRI (functional Magnetic Resonance Imaging) and PET (Positron Emission Tomography) studies. With respect to language processing, the functional mapping of brain regions is important, because it forces the design of language processing models to be more accurate. There are fMRI and PET studies of language processing like Mazoyer et al. (1993), who studied the activation of cortical areas in listening to a story, which reveal a lot about neural localization of language. Even though the model presented here is not concerned with the neural localization of language, any experiment that has neural imaging data and that is to be simulated on a cognitive architecture having the theory of neural localizations would have to use all and only the modules that correspond to the cortical areas that were active during the real experiment.

3.2 Knowledge Representation and Symbolic Learning

As it was mentioned above ACT-R assumes that there are two kind of knowledge – declarative and procedural. ACT-R represents declarative knowledge units in terms of chunks (Miller, 1956), and procedural knowledge units in terms of production rules (Anderson & Lebiere, 1998c). The information given in this section is taken mostly from Anderson and Bothell (2004).

3.2.1 Chunks

Chunks are "configurations of elements that encode various things that we know" (Anderson & Lebiere, 1998b, p. 5). What information a chunk encodes is model specific. It can encode facts like that dog is a mammal or that it is a noun. In ACT-R syntax these facts are written as:

```
Fact1                       LFact105
  isa    category-fact        isa    lexical-fact
  animal dog                  word   dog
  class  mammal               categ  noun
```

Fact1 and LFact105 are arbitrary names of the chunks. The name is followed by a number of slots having certain values. ISA slot is different with respect to others since it specifies the type of chunk. Types of chunk are declared in the model, and there is no convention on how many slots it should have or how they should be named, except that the names should be meaningful (e.g. lexical-fact for lexical category information). Slot names like chunk names are also given arbitrarily.

Declarative memory chunks are either specified by the modeler or learned by the model. Learning a chunk simply means its addition to declarative memory. There are two ways to accomplish this. A chunk can be directly encoded from the environment: visual and audition modules encode features as declarative memory chunks, or it can be a result of an accomplished goal (Anderson & Lebiere, 1998d).

3.2.2 Production Rules

Production rules are condition-action pairs as it was mentioned before. Like chunks they also have certain representational commitments in ACT-R. An example below illustrates the production that checks whether dog is a mammal:

```
(p check-category
   =goal>
     isa       classification
     state     “questioned”
     animal    dog
     kindof    mammal
==>
   +retrieval>
     isa       category-fact
     animal    dog
     class     mammal
   =goal>
     isa       classification
     state     “requesting”)
```

If we translate this representation into English if-then pair form it would look like:

```
If the goal is to check classification
    and the question was received
    and the animal is dog
    and the kind to check is mammal
Then
    try to remember whether there is a fact that
    animal dog is
    classified as a mammal
And go on to the next step.
```

Check-category is an arbitrary name similar to chunk names. The head of the chunk is separated from its slots by the “>” sign. The terms ‘goal’ and ‘retrieval’ are reserved for the chunks in the goal and retrieval buffers. The “=” sign is used to denote variables. In the example it precedes buffer names, which means that they will be bound to the name of the chunk in the corresponding buffer. The “==>” is a separator between condition and action parts. The condition part specifies what must be the contents of the buffers for the production to be executed, and the action part specifies the changes to be made to the contents of the buffers. The “+” sign on the action side denotes a retrieval request to the declarative module (i.e. the production rule requests the declarative memory module to look for and retrieve a chunk that matches the criteria (chunk type and slot values) in the retrieval buffer). On the condition side the "=" sign is used to check the content of the buffer, whereas on the action side it is used to modify the chunk currently in the buffer.

Similar to the process of declarative knowledge acquisition, there is a process of production rule learning as well. In this process, called production compilation, declarative knowledge is transformed into procedural form. A new production is formed by the combination of two productions into one. The process can only be used if the first production makes a retrieval request used by the second one, because of two reasons. First, perceptual-motor buffers have the risk of jamming[9] if both productions make requests to the same buffer. Second, if two productions are combined "when the first production makes a request for a perceptual encoding or motor action and the second production depends on completion of this request" (Anderson & Bothell, 2004, p. 27), then in a combined production this dependence will be lost and the production will lead to unpredictable results.[10] The process of proceduralization makes model faster and more accurate, since it eliminates declarative memory retrievals that take time and produce errors by retrieving wrong chunks (Anderson & Lebiere, 1998d).

The processes of learning, declarative memory retrieval and production selection are using matching. If conditions match, then the production rule is selected, and a chunk retrieved from declarative memory is also selected according to the matching slot values. Besides symbolic matching requirements imposed by the

[9] Because production system and perceptual-motor module operate in parallel and the difference in time it takes to complete production (0.050s) and perform an action (e.g. move attention 0.085s), the second production can request the same module to perform another action before it completes the action requested by the first production. In such conditions the buffer of that module is jammed.

[10] For example: if the first production checks contents of the visual buffer for a string of characters and the second production depends on this string (which is the case in reading), then it is not possible to combine these two productions, because the combined production will not depend on the content of the visual buffer (i.e. a word's meaning will not depend on its spelling).

systems, the processes are controlled by subsymbolic parameters, which are discussed in the following section.

3.3 Subsymbolic Level

Since ACT-R is a hybrid cognitive architecture it has two levels – symbolic and subsymbolic. Most of the symbolic level was discussed in the previous sections. Subsymbolic level, borrowed from connectionism, plays an important role in retrieval of chunks from declarative memory and selection of production rules. It is the subsymbolic level that determines overall behavior of the model (Anderson, Lebiere & Lovett, 1998).

3.3.1 Declarative retrieval process

Production rules make retrieval requests to the declarative memory specifying type of the chunk to be retrieved and some slot values. The system makes a parallel search in the declarative memory for chunks that meet the requirements, i.e. have the same slot values. It is usually the case that several chunks match the criteria, and the chunk with the highest activation is retrieved. The activation value of a chunk determines its probability of being retrieved and the speed of retrieval (Anderson et al., 2004). The activation of a chunk ***i*** is calculated according to the **Activation Equation 3.1**:

$$A_i = B_i + \sum_j W_j S_{ji} + \sum_k P_k M_{ki} + \varepsilon_1 + \varepsilon_2$$ **Activation Equation 3.1**

In the formula $\boldsymbol{B_i}$ is a base-level activation of the chunk ***i***, a value that reflects “general usefulness of the chunk in the past” (Anderson et al., 2004, p.18) and it rises with practice and falls with delay (Anderson et al., 2004). Base-level activation is calculated according to the **Base-Level Learning Equation 3.2**:

$$B_i = \ln(\sum_{j=1}^{n} t_j^{-d})$$ **Base-Level Learning Equation 3.2**

"Where $\boldsymbol{t_j}$ is the time since the $\boldsymbol{j}^{th}$ practice of an item" (Anderson et al., 2004, p.19), ***n*** is a total number of practices, and ***d*** is a decay parameter, whose default value[11] is taken to be 0.5 (Anderson et al., 2004). Practice of a chunk occurs in three cases: when it is initially created, when it is merged[12] with exactly the same chunk, and when it is retrieved from declarative memory (Anderson & Bothell, 2004).

The second addend in the Activation Equation - associative activation - "reflects the relevance to the current context" (Anderson et al., 2004, p. 18). Sources of activation, ***j***, are slot values of the chunk in the goal buffer that are chunks themselves. These chunks have certain amount of association to chunk ***i***. $\boldsymbol{S_{ji}}$ is strength of association from chunk ***j*** to the chunk ***i***, calculated by the formula:

$$S_{ji} = S - \ln(fan)$$ **Strength of Association Equation 3.3**

where ***S*** is the maximum associative strength between two chunks, usually defaulted to 2, and "$\boldsymbol{fan_j}$ is the number of facts associated to term ***j***" (Anderson et al., 2004). $\boldsymbol{W_j}$ is the attentional weighting[13] of the sources of activation (***j***) that are equated to ***W*** divided by ***n***, with ***n*** being the number of the sources of activation and ***W*** usually defaulted to 1 (Anderson et al., 2004).

[11] Default values are either taken from empirical data (e.g. 0.085s to move attention) or found out as recurring approximations in a number of models (e.g. decay parameter 0.5).

[12] If a chunk is created with exactly the same slot values with another chunk in declarative memory, instead of adding it to the declarative memory it is merged with the original.

[13] The importance given to a particular slot in a goal chunk.

The third addend in the Activation Equation is the matching component. Elements to be matched, $\boldsymbol{k}$, are slot values specified in the retrieval request. $\boldsymbol{P_k}$ is a match scale, which "reflects the amount of weighting given to the similarity in slot $\boldsymbol{k}$" (Anderson & Bothell, 2004), and which has the default value of 1. $\boldsymbol{M_{ki}}$, match similarity, "reflects the similarity between the value $\boldsymbol{k}$ in the retrieval specification [i.e. value of the slot in the retrieval request] and the value in the corresponding slot of chunk $\boldsymbol{i}$ [i.e. value of the same slot in the retrieved chunk]" (Anderson & Bothell, 2004). The value of $\boldsymbol{M_{ki}}$ depends on the maximum similarity and difference values which are 0 and -1 by default, respectively. If the slot values match it is 0, if not it is -1. Consequently, this addend could be thought as loss of activation because of mismatch.

The last two addends are permanent ($\boldsymbol{\varepsilon_1}$) and transient ($\boldsymbol{\varepsilon_2}$) noise values. Permanent noise is added to the activation value of a chunk once when it is created; and transient noise is a random value added to the activation of a chunk upon every retrieval attempt (Anderson & Bothell, 2004). Together with another value used in the process of declarative retrieval, a retrieval threshold[14] $\boldsymbol{\tau}$, which is a value specifying minimum activation necessary for the chunk to be retrieved, they are used to simulate errors. Errors of omission occur because of the failure to retrieve the right chunk because its activation value falls below the threshold when noises are added. Errors of commission, similar to errors of omission, occur due to the variation of activation and retrieved chunk is not the right one (Anderson & Bothell, 2004).

How fast a chunk is retrieved depends on its activation, as was mentioned before, and calculated according to the **Latency of Retrieval Equation 3.4**:

[14] Retrieval threshold can be seen as a "retrieval-failure" chunk that is retrieved when its activation is higher than the activation of the requested chunk (in case if there is no partial matching).

$$RT_i = Fe^{-A_i}$$ **Latency of Retrieval Equation 3.4**

In the formula $\boldsymbol{A_i}$ is an activation value of a chunk, and ***F*** is a latency factor, which together with the retrieval threshold value is the most variable parameter across ACT-R models (Anderson et al., 2004). The general relationship between the two parameters is stated as:

$$F \approx 0.35e^{\tau}$$

This means that the chunk with the activation value equal to the retrieval threshold is retrieved approximately in 0.35 seconds (Anderson et al., 2004).

3.3.2 Production selection process

Similar to the retrieval request, where several declarative memory chunks meet the retrieval criteria, several production rules match the buffers at the same time (thus can be selected). Besides the symbolic matching, production rules are selected according to some subsymbolic value. In case with declarative memory chunks this value is activation, whereas in case with production rules this value is utility. A production with the highest utility value is selected, and the utility of a production ***i*** is calculated according to the **Production Utility Equation 3.5**:

$$U_i = P_iG - C_i + \varepsilon$$ **Production Utility Equation 3.5**

$\boldsymbol{P_i}$ is the probability that the goal will be achieved when the production ***i*** is selected. ***G*** reflects the importance of achieving the goal[15] or maximum expendable time for the achievement of this goal (Anderson, Lebiere & Lovett, 1998). The default value

[15] ACT-R simply assumes that the value of the goal is some internal importance given to it by a person without considering the reasons and its exact value, which is usually irrelevant to model's predictions (Anderson, Lebiere, & Lovett, 1998).

of the ***G*** parameter is 20 seconds. C_i is the production cost and reflects the time that will be spent to achieve the goal if the production ***i*** is selected. Like activation values of declarative memory chunks, production utilities are also noisy (Anderson et al., 2004). The noise ε is added to make the execution variable.

Both P_i and C_i values are changing with usage of the production rule, like base-level activation of declarative memory chunks. The probability of success (P_i) is calculated by the simple formula using the numbers of experienced successes and failures across all usages (Anderson & Bothell, 2004):

$$P_i = \frac{\text{Successes}}{\text{Successes} + \text{Failures}}$$ **Probability of Success Equation 3.6**

The cost of production (C_i) is learned by the formula similar to the one used to calculate the probability of success:

$$C_i = \frac{\text{Efforts}}{\text{Successes} + \text{Failures}}$$ **Production Cost Equation 3.7**

"***Efforts*** is the accumulated time over all the successful and failed applications of this production rule" (Anderson & Bothell, 2004, unit 8). Both P_i and C_i are given some initial values[16] to reflect the prior experience and avoid continuous selection of the production that was applied and succeeded first (Anderson et al., 2004).

Production rules, like declarative memory chunks, also must have the utility higher than some specified value in order to be selected. This minimal value is utility threshold and it acts similar to the declarative memory retrieval threshold.

[16] By default these values are: P = 1 and C = .05 s, because default values of Efforts is .05 s, Successes is 1, and Failures is 0. However, they can be set differently to favor the specific production.

Langacker (1987, 1991, 1999a) makes extensive use of notions like entrenchment and association that are clearly identifiable in the subsymbolic level of ACT-R. Both production selection and declarative retrieval processes described here make the system adaptive. The ability of the system to change the activation of declarative memory units with respect to the context is a basis for the explanation of polysemy in language. Different contexts act as sources of activation for different meanings and as a result the right meaning of the word is retrieved. The ability to adjust base-level activations with practice is the basis for the explanation of the shifts in meaning. For example: more frequent use of the word *mouse* to mean computer accessory might make this meaning the dominant one. Since much in the language is gradable many linguistic phenomena can be explained in terms of activation. The model presented in this thesis also makes use of the subsymbolic level[17] to model different notions of Cognitive Grammar in the representation as well as retrieval of lexical items, which are discussed in Chapter 4.

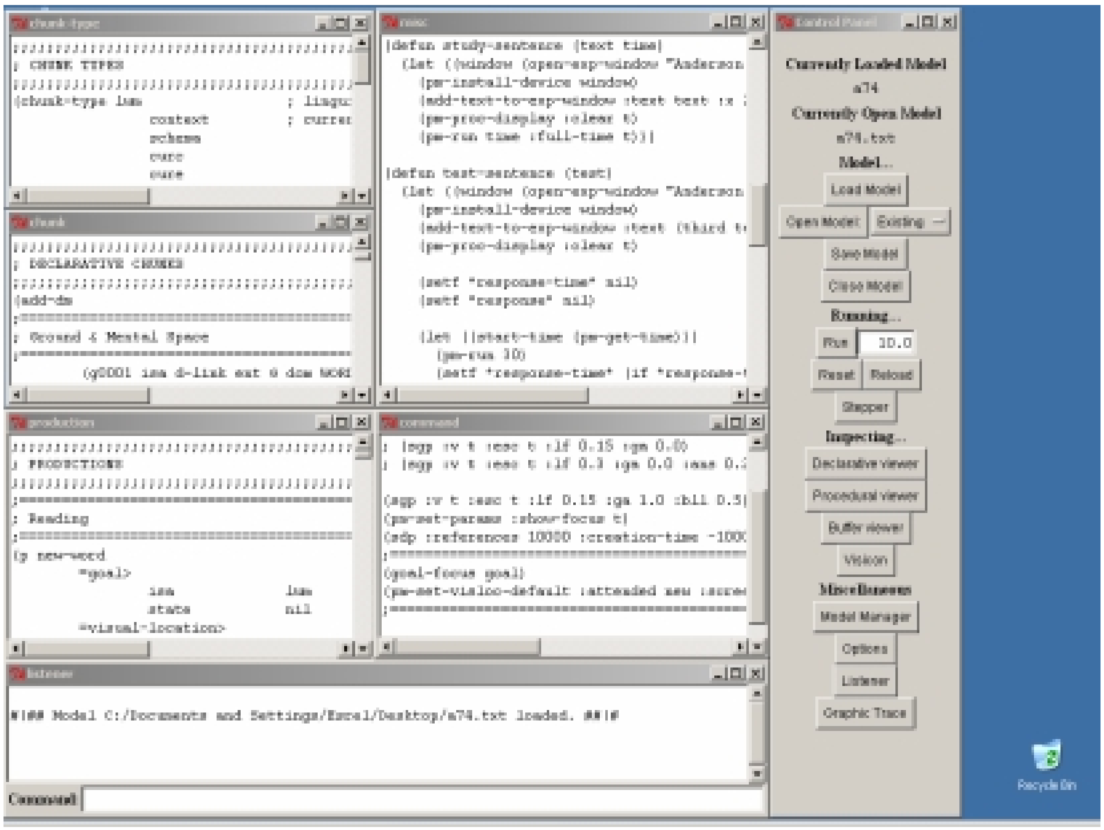
3.4 ACT-R Modeling Environment

The ACT-R theory is represented by ACT-R modeling environment as a set of functions and algorithms written in Common Lisp. There are two forms of the modeling environment – the basic and standalone versions; the basic version requires some Lisp environment, whereas the standalone does not. The standalone version comes together with ACT-R environment, which is a set of graphical user interface (GUI) tools for running and debugging models. The tools allow to analyze the state of the model with respect to contents of the buffers and subsymbolic values (e.g. activations) of chunks and productions.

[17] The use of the subsymbolic calculations is optional in ACT-R models.

The basic functions for designing of simple experiments come together with the environment. They are used to display items (e.g. to show text on the screen), to collect responses such as mouse clicks and key presses (i.e. record the time of action), and to analyze data in terms of correlation and mean deviation between two sets of values.

The model in ACT-R is a text file, and it includes five areas that are displayed in a separate window each when it is opened by the environment. These areas are: chunk-type declarations that specify the slots of chunks, initial content of declarative memory (i.e. chunks), initial content of procedural memory, functions to run the experiment, and global parameter specifications like threshold of activation and various other parameters that control the output of the model. Figure 3.3 shows these five areas together with the Control Panel and the Listener.



```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; CHUNK TYPES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(chunk-type lsm                        ; lingu
                  context              ; curren
                  schema
                  curr
                  cure
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; DECLARATIVE CHUNKS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(add-dm
;==============================================
; Ground & Mental Space
;==============================================
          (g0001 isa d-link ext 0 dom WORD
```

```
(defun study-sentence (text time)
  (let ((window (open-exp-window "Anderson
    (pm-install-device window)
    (add-text-to-exp-window :text text :x
    (pm-proc-display :clear t)
    (pm-run time :full-time t)))

(defun test-sentence (text)
  (let ((window (open-exp-window "Anderson
    (pm-install-device window)
    (add-text-to-exp-window :text (third t
    (pm-proc-display :clear t)

    (setf *response-time* nil)
    (setf *response* nil)

    (let ((start-time (pm-get-time)))
      (pm-run 10)
      (setf *response-time* (if *response-
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PRODUCTIONS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;==============================================
; Reading
;==============================================
(p new-word
        =goal>
              isa              lsm
              state            nil
        =visual-location>
```

```
;  (sgp :v t :esc t :lf 0.15 :gm 0.0)
;  (sgp :v t :esc t :lf 0.3 :ga 0.0 :mas 0.
(sgp :v t :esc t :lf 0.15 :ga 1.0 :bll 0.5)
(pm-set-params :show-focus t)
(sdp :references 10000 :creation-time -100
;==============================================
(goal-focus goal)
(pm-set-visloc-default :attended new :scre
;==============================================
```

```
#|## Model C:/Documents and Settings/Esral/Desktop/a74.txt loaded. ##|#
```

Figure 3.3: The ACT-R Environment

The Control Panel displays the current state of the environment (i.e. currently loaded and opened models), and contains a set of buttons to manipulate a model (load, run, save, etc.), to run various tools to inspect models, and configure and manage the environment. One of the tools, the Stepper, allows step-by-step execution of a model and at every step contents of the buffers and the procedural and the declarative memories can be viewed, as well as production rules applicable to current state and declarative chunks that match retrieval criteria.

The Listener works similar to a standard listener in Lisp: it is used for input and output. The input, which is generally a number of functions to be executed, is entered in the Command area of the Listener. The output is generally the traces produced by the model run.

For example, a model run can produces a trace like the one below:

```
Time  0.000: Vision found LOC21
Time  0.000: New-Word Selected
Time  0.050: New-Word Fired
Time  0.050: Module :MOTOR running command CLEAR
Time  0.050: Module :VISION running command MOVE-ATTENTION
Time  0.100: Module :MOTOR running command CHANGE-STATE
Time  0.135: Module :VISION running command ENCODING-COMPLETE
Time  0.135: Vision sees TEXT16
Time  0.135: Read-Word Selected
Time  0.185: Read-Word Fired
Time  0.185: S0001 Retrieved
Time  0.185: Encode-Word Selected
Time  0.235: Encode-Word Fired
```

The output varies with respect to the parameter set used. The trace presented here displays production selection and chunk retrieval information and the time of each operation.

The model in this thesis was developed using the standalone version of the environment for Microsoft Windows, which can be downloaded from ACT-R’s homepage (http://act-r.psy.cmu.edu).

3.5 Language Related Models

Published/presented models implemented on the ACT-R architecture are available on the architecture's homepage (http://act-r.psy.cmu.edu). Published language processing models fall into four categories: parsing, analogy and metaphor, language learning and sentence memory. This section discusses models with respect to the linguistic theories they follow and, especially, the meaning representations modelers used.

3.5.1 Parsing

Models presented in this subsection are categorized as parsing ones, since their authors are primarily concerned with syntactic phenomena. Lewis (1999b) deals primarily with sentence complexity and garden-path effects. He uses the fragmented representation of syntactic trees where every link is a separate memory chunk. After each word, the reader (or model) retrieves the lexical item which holds information about its syntactic category, and uses this information to integrate that word into the parse tree. He models language phenomena by the activation effects (decay and similarity) on these separate syntactic tree parts in the retrieval process. Phrase structure rules are in the form of production rules; thus, he assumes that language processing is procedural.

Emond (1997, 1999) is primarily concerned with anaphora resolution processes in the context of the simultaneous influence of syntactic, discourse and semantic information. He uses different levels of representation for lexical items: the string of characters, syntactic category and semantic information (person, state or event). Syntax has the form of an extended categorical grammar (Bach, 1983 in Emond, 1997, p. 5) and sentences are processed using the chart parsing algorithm. That is, the phrase structure rules are retrieved from declarative memory unlike Lewis (1999b). His work supports the idea that “anaphora resolution processes are initiated at the moment of reading an anaphoric element but that in case of ambiguity, these processes are not completed before additional information is provided through subsequent reading or listening” (Emond, 1997, p. 4); and that it can be correctly modeled by a serial process.

Emond (1997, 1999) mentions that in anaphoric as well as simple sentences pronouns activate all potential referent candidates in the discourse model and identify the single one that best meets the requirements imposed by the predicate structure. Mental spaces (Fauconnier, 1985, 1997, 1999) discussed in Chapter 2 are discourse models; consequently, construction of a discourse model as an interpretation of the sentence and its usage as a search space for anaphora resolution coincides with the ideas from Cognitive Linguistics.

3.5.2 Metaphor and analogy

Models described in this subsection are primarily focused on metaphor (Budiu, 2001; Budiu & Anderson, 2000, 2001, 2002, 2003, 2004) and analogy (Salvucci & Anderson, 2001). Similar to models described in the previous subsection, models described here are guided by some assumptions with respect to syntax and semantics.

Although there are several published models focused on metaphor, modeling of the phenomenon is discussed with respect to Budiu and Anderson’s (2004) “Interpretation-based Processing” theory of sentence comprehension, since prior publications by the same authors present certain portions of the same theory. According to the theory, sentence comprehension depends primarily on prior knowledge, and understanding a sentence is the same with finding a similar fact (or facts) in the declarative memory and relating the sentence to it. This sentence comprehension idea is very similar to the usage-based model of Langacker (1999a). In Chapter 2 it was mentioned that linguistic units sanction the usage of novel structures; consequently, the interpretation of the sentence is the same as the selection of a set of categorizing structures. Despite this similarity, there is no reference to Langacker in Budiu & Anderson (2004). It is possible to posit that there is an undocumented closeness between their ACT-R model's assumptions and Cognitive Linguistic assumptions.

The developed models successfully[18] account not just for the metaphor, but also for the sentence with a literal meaning and sentences containing semantic illusions[19] (Budiu & Anderson, 2004). There are two levels of representation (syntactic and semantic parse trees) that are constructed in parallel after each new word. After each content word (noun phrases, verbs, adverbs, etc.) the model looks for the best match for what it read based on semantic similarity and controls the selected meaning with subsequent semantic units; abandons it for another one if there is a mismatch (Budiu & Anderson, 2004). Syntactic representation is a simplified version of X-bar syntax of Jackendoff (1977) (cited in Budiu & Anderson, 2004, p. 39). Both syntactic and semantic representations are based on the fragmented representation of declarative knowledge introduced in Anderson, Bothell, Lebiere & Matessa (1998) and developed in Anderson, Budiu & Reder (2001) and Salvucci &

[18] The success of the model is judged with respect to how close is the model's performance in terms of speed and errors to the performance of real subjects. It is measured in terms of correlation and mean deviation between the results obtained from the simulation and the real experiment.

[19] Semantic (or Moses) illusion is the name for the phenomenon when people fail to find distortion in sentences like "How many animals of each kind did Moses take on the ark?" despite their knowledge of the fact that actually Noah, not Moses, took animals on the ark (Budiu & Anderson, 2004).

Anderson (2001). The knowledge representation is similar to the representation used in list memory: items are grouped together and groups are linked to the list (e.g. in the number 571,234,568 the digits are grouped in groups of three for better memory). Information in this kind of representation is held in the connections between items. Figure 3.4 illustrates the semantic representation of the sentence "the college students were taught by professors of good reputation" from the model:

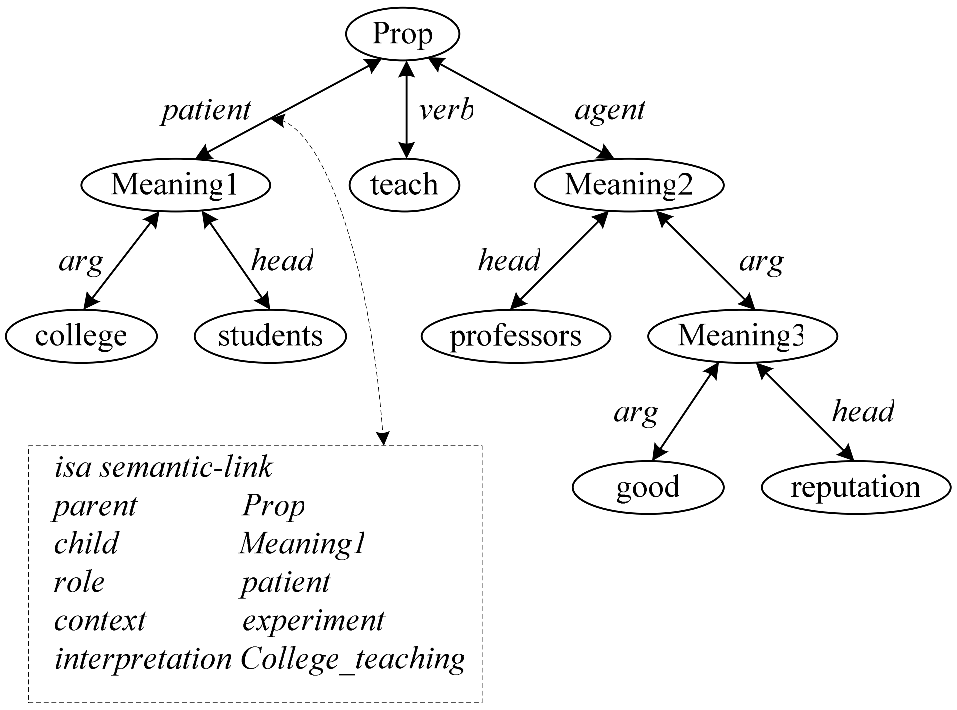


Figure 3.4: Fragmented representation of semantic structure (Budiu & Anderson, 2004, p. 5)

Representations of lexical items contain information about their orthography, syntactic category and meaning. Meaning of an item is not elaborated, that is, there is no information about its features; however, semantic similarity is used for the selection of a candidate interpretation. Since similarity between two chunks is determined by the number of slot values they share, and meaning in the model is represented by a chunk without any slots, ACT-R's matching mechanism cannot detect the similarity between two meanings. Thus, they have used Latent Semantic Analysis (Landauer & Dumais, 1997) to calculate semantic similarity values.

Another important contribution of Budiu's work is a model of word learning in context (Budiu & Anderson, 2001). The model's primary concern is the learning of metaphorical meanings of words and meanings of new artificial words. The experiment and the model supported that new words are understood and learned using context matching (i.e. checking what else occurred in the same context). Moreover, it showed that word learning is incremental and it is abstracted from a number of examples. Both findings are consistent with the process of schematization discussed in Chapter 2.

Salvucci and Anderson (2001) describe "the path-mapping theory of how humans integrate analogical mapping and general problem solving" (p. 67). The theory is based on the previous work by the same authors (Salvucci & Anderson, 1998), where they showed that complex problems are solved by decomposing them into components and using analogy to solve new problems. The representation used is similar to Budiu and Anderson (2004) described above, that is, the meaning of a single proposition is encoded using several chunks. The concept holds information about the relation it participates in and its role in the relation. The theory was tested using several models, such as similarities between the solar system and an atom, and provides an excellent fit to the experiments presented in the article.

The path-mapping theory fits in the cognitive semantics framework. It is quite similar to the theories of Sweetser (1990) and Lakoff (1987), where abstract domains were structured using basic domains. According to the path-mapping theory, concepts are represented using their roles within the structures (processes and relations), and analogy is made mapping these roles: source domain is modified to fit the requirements of the target domain. The analogy was proposed as an alternative to the rules for linguistic regularity and language production (Bloomfield, 1933 in Langacker, 1999a). In terms of Cognitive Grammar the analogy can be seen as a categorization by low-level schema, in cases when high-level schema has not yet achieved unit status.

3.5.3 Language learning

Models presented in this subsections show the widest range of application of ACT-R to language domain. ACT-R's learning mechanism was tested on the tasks like role assignment (Matessa & Anderson, 1997), communication (Matessa & Anderson, 2000), learning constraint ranking in Optimality Theory (Prince & Smolensky, 1997) with respect to syllabification and past tense learning (Misker & Anderson, 2003), past tense form learning in German and English (Taatgen, 2001; Taatgen and Anderson, 2002; Taatgen & Dijkstra, 2003) and learning functions of determiners (Zondervan & Taatgen, 2003). Role assignments and communication models are presented first, then comes Optimality Theory model, and the rest of the models are discussed together with respect to U-shaped learning[20].

Matessa and Anderson (1997) addressed the question whether ACT-R's learning mechanism can be used in modeling language acquisition. They developed a language comprehension model where the system had to assign linguistic roles

[20] Learning of past-tense in English is traditionally divided into three stages. In the first stage when children just start using past tense they use irregular verbs correctly. Then, in the second stage, the regular past tense rule is discovered, and children occasionally overgeneralize the rule and say 'goed' instead of 'went'. In the third stage, the overgeneralization of the rule is abandoned and irregular verbs are used correctly again. "Since performance on irregular verbs is worst in the second stage, the performance curve has a U-shape, hence the name of the phenomenon" (Taatgen & Anderson, 2002, p. 124).

(theta-roles) to nouns presented in a sentence. The role assignment is based on linguistic cues such as case markings, word order and semantic properties of the noun (animacy) (Matessa & Anderson, 1997). However, cues in different sentences may be contradictory (e.g. in one sentence first noun is an agent and in another sentence with the same structure first noun is an instrument). It is suggested that these contradictions are resolved by the cue dominance hierarchy of the language, where the validity of the cue is calculated using two statistics – how often the cue is present in the sentence (availability) and how often it assigns the role correctly (reliability) (Matessa & Anderson, 1997). The authors modeled the order of application of cues in a role assignment task and suggested that ACT-R might be useful for predictions in language domain (Matessa & Anderson, 1997).

Matessa and Anderson (2000) developed a communication model for dialogue acts. It is suggested that communication presupposes establishing common ground, which is similar to the term *ground* used in Cognitive Grammar to refer to the context of the speech act that includes previous discourse, time, place and any other relevant information shared by the speech-act participants, but has a narrower scope. In the model common ground is expanded by the semantic and syntactic representations of the conveyed information. These representations are used for generation of new utterances (Matessa & Anderson, 2000). With the model they showed that ACT-R's mechanism of storing accomplished goals as declarative memory chunks can be used for creation of a common ground.

Misker and Anderson (2003) model is an attempt to unite the Optimality theory (Prince & Smolensky, 1997) with ACT-R in order to provide the ACT-R with linguistic module. They addressed two linguistic phenomena – syllabification and past tense formation – in terms of constraint learning.[21] ACT-R's learning mechanism was successful in learning the constraint ranking but the model's performance can be compared only to early stages of language acquisition.

Past tense learning is discussed with respect to the Taatgen and Anderson (2002) model since Taatgen (2001) and Taatgen and Dijkstra (2003) are extensions to the English past tense learning model developed in Taatgen and Anderson (2002) applied to German past tense formation and error pattern in irregularization process respectively. German past tense forms similar to English ones exhibit the same U-shaped learning. The low error rate with irregular past tense forms of English verbs (i.e. why mistakes like "goed" are common, but like "brang" (as a past tense of "bring") are not) is explained by the lack of facilitation and generality to establish a separate rule and overgeneralize it (Taatgen & Dijkstra, 2003).

Past tense learning model of Taatgen and Anderson (2002) is particularly interesting because it explains the U-shaped learning function of English irregular verbs in very basic terms; whereas other "cognitive models often rely on a sudden increase in vocabulary, a high token-frequency of regular verbs, and complicated schemes of feedback in order to model this phenomenon" (Taatgen & Anderson, 2002, p. 123). Their model's learning depends on the frequency of usage of the verb: very frequent verbs are irregular since it is cheaper (in terms of time) to memorize the form for the fast retrieval; less frequent verbs are regular since to memorize different forms for few occasions is not economic (Taatgen & Anderson, 2002). In ACT-R's activation mechanism (base-level learning) more frequent words have

[21] Hypothesis of Optimality theory is that "a grammar consists entirely of constraints arranged in a strict domination hierarchy, in which each constraint is strictly more important than – takes absolute priority over – all the constraints in the hierarchy" (Prince & Smolensky, 1997, p. 318). This ranking of the constraints determines the well-formedness of the linguistic structures.

higher activation and are retrieved faster, and U-shaped learning will be due to “a temporary imbalance between retrieving examples and using the rule, at a moment that the learning of the examples hasn’t properly settled on the eventual activation values of the examples” (Taatgen & Anderson, 2002, p. 133).

Zondervan and Taatgen (2003) describe the model of children’s acquisition of determiners with respect to Representational Redescription theory (Karmiloff-Smith, 1992 in Zondervan & Taatgen, 2003). The Representational Redescription theory describes “cognitive development as a process in which knowledge becomes more and more explicit” (Zondervan & Taatgen, 2003, p. 225). In their model they displayed “how humans can make use of general, redescribed knowledge to specialize their general strategy of searching for regularities in their environment” (i.e. make domain specific knowledge more explicit and available for other tasks) (Zondervan & Taatgen, 2003, p. 230).

Models in this subsection have dealt with quite narrow linguistic phenomena and it is difficult to state their perspective with respect to Cognitive Linguistics. But, past tense learning model could be integrated in the framework of Cognitive Grammar easily, because it is based on the same premises, that is, the abstraction of recurring commonality and its application for categorization of other expressions. Modeling language acquisition is very important for both ACT-R and linguistic theories. ACT-R can simulate time passage (e.g. Zondervan & Taatgen (2003) simulated nine years of development) and reveal a lot about early language acquisition.

3.5.4 Sentence memory

Sentence memory is exemplified with just one publication – Anderson, Budiu and Reder (2001). Most of the information given in this article has already been mentioned in subsections on parsing and metaphor and analogy. The hypothesis of their model was that there is nothing special about sentence memory with respect to other types of memory (e.g. list memory) and that there is a single retention function for all levels of sentence memory (i.e. the exact words (surface form and syntax), textbase (propositions), and situation model (inferences from long term memory) have the same rate of forgetting), and the reason why meaning is remembered better is its being represented by fewer chunks. The representations used in the model are exactly the same with Budiu and Anderson (2004) and Salvucci and Anderson (2001). The sentence is parsed and syntactic parse tree and semantic representations are constructed. The representation is fragmented, as was mentioned above, and it was tested on various sentence memory tasks like sentence verification and recall. The model provides good data fits with real experiments with respect to latency, forgetting and errors; and supports their hypothesis that the number of chunks used in the representation is the reason for better memory for propositions.

All the models published under the category of language processing adopted one or another theory somehow related to language processing. The major models are Anderson, Budiu and Reder (2001), Budiu and Anderson (2004), Salvucci and Anderson (2001) and Taatgen and Anderson (2002) dealing with sentence memory, metaphor, analogy and language learning respectively. The models from above that do sentence processing are first and second, both using almost the same representation and processing. The syntactic and semantic processing is assumed to be done in parallel and syntactic and propositional representations are fragmented, that is, encoded in a number of chunks. There is a great deal of similarity between representations and techniques used in the above models and the proposals of Jackendoff (2002) and Langacker (1987, 1991, 1999a). It is observed that ACT-R community has not yet directed their attention to Cognitive Linguistics. It is one of the aims of this thesis to start an evaluative framework that involves models of Cognitive Linguistics, in particular Cognitive Grammar, against the background of ACT-R as a cognitive architecture.

This chapter presented the ACT-R architecture together with its assumptions about neural mapping of the architecture, representational commitments and subsymbolic processes that control memory phenomena. Description of the language related models published by the ACT-R community was intended to provide an idea about the points of view modelers have on linguistics, that is, the theories they adopt.

CHAPTER 4

A Cognitive Grammar Based ACT-R Model of Sentence Memory

The construction of a cognitive model can be divided into a number of steps. First of all, some task is selected and analyzed in terms of the knowledge necessary to perform the task and the way how the task is performed. Second, some cognitive architecture is selected and the required knowledge and an algorithm of the task are mapped onto the architecture, following all its constraints and conventions. Then, predictions are made about what will be the performance of the model on the task. Finally, the model's performance is evaluated with respect to the real data obtained from a psychological experiment on that task.

Section 4.1 describes the experiment of Anderson (1974) that was selected as a specific task for the model. Previous ACT-R models of the experiment are discussed in Section 4.2 with respect to the representations and processing. ACT-R was presented in Chapter 3 as the cognitive architecture selected for the model. The processes and representations required for the task from the standpoint of Cognitive Grammar (the theory of language that was presented in Chapter 2), as well as their implementations on ACT-R are discussed in Section 4.3. The performance of the model in terms of whether it can simulate the human data in terms of time and precision is evaluated in Section 4.4.

4.1 *The Task - Anderson (1974)*

The selected task is one of the experiments in Anderson (1974).[22] The author reports two experiments that were conducted to distinguish between immediate and long-term memory for sentential information. The research was motivated by the fact that the memory for the meaning of a sentence is much better than the memory for the form. Anderson assumed that long-term memories are "semantic interpretations (i.e. propositions) of sensory experiences" that are "similar to the active-like deep structure of Chomsky (1965)", whereas immediate memory is a verbatim image (Anderson, 1974, p.149). The two hypotheses were contrasted - the Auxiliary Encoding Hypothesis and the Verbatim Hypothesis. The former claims that "the information about the form of the sentence is held in a structure auxiliary to that structure that contains information about the sentence's semantic content" (e.g. *The sentence X was in the passive voice*) (Anderson, 1974, p. 150). The latter, on the other hand, claims that there is a probability that verbatim images of the sentences can exist in long term memory. This section overviews the method and the results of the experiments that aim to discriminate between these two hypotheses.

4.1.1 Method

The discrimination between the Auxiliary Encoding and the Verbatim Hypotheses is "crucial for evaluating the claim that all information in long-term memory is propositional" (Anderson, 1974, p. 150). The verbatim form of the sentence is different from its meaning; thus, the sentence memory experiment is suitable for the evaluation of this claim. Consequently, Anderson (1974) selected a

[22] This experiment was initially selected, because it was used by Anderson, Budiu, and Reder (2001) to test their representation. However, as it is discussed in Section 4.3.1, the simulation of this particular experiment is a good test for the representations and processing.

sentence verification task, in which reaction times might help to distinguish between these hypotheses about the memory for the form.

In sentence verification tasks, subjects are presented with a set of sentences to be remembered, and afterwards they are tested with another set of sentences, some of which are constructed by changing the voice of the input sentences. The subjects have to judge whether the probe sentence was implied by the input sentence. In the tests of the immediate memory for sentences, it has been found that subjects verify the sentences faster if their voices match the voice of the input sentence. However, on the assumption that sentences are represented in the form of propositions (neutral with respect to voice) in long-term memory, the voice of the input sentence should not affect verification time when the information for the input sentence is retrieved from long-term memory.

In both experiments, three factors were manipulated – voice of the input, voice of the probe, and truth condition; which result in eight conditions. The subjects studied a set of eight sentences – four in active and four in passive voice, and then they were tested with another set of eight sentences that was constructed by switching the voice of the half of the studied sentences. False sentences were constructed by switching the logical subject and the logical object of the study sentences. The subjects had to verify the truth of test sentences with respect to study sentences.

For example, the sentence (1) *the painter visited the missionary* and its passive variant (2) *the missionary was visited by the painter* imply the same thing (i.e. their propositional representations are the same), whereas the propositional representations of the sentences (3) *the missionary visited the painter* and (4) *the painter was visited* *by the missionary* are different from those of the sentences (1) and (2). If sentence (1) was presented during study phase, then in the true condition it would be tested with sentence (2), and in the false condition with either sentence (3), or sentence (4). In the first experiment, the subjects studied sentences within a short story to reflect the natural comprehension of sentences. In the second experiment, they studied isolated sentences that are usually used in verification tasks.

Anderson (1974) tested subjects under two conditions – immediate and delayed. The reaction times in the immediate condition would reflect the time it takes to read the sentence and match its form to the verbatim trace of the input sentence in short-term memory. The reaction times in the delayed condition would reflect the time it takes to read the sentence, construct its propositional form, and compare this form to the propositional form of the input sentence. According to the Auxiliary Encoding hypothesis, there is no effect of the voice of the input sentence in the delayed condition (because the information about the form is encoded in another proposition, which is not necessary for the truth judgments). However, the Verbatim Hypothesis expects the voice of the input sentence to affect the verification time, because the verbatim trace can also be retrieved and used for judgments.

The first experiment was conducted two test these two hypotheses. In the immediate condition of the first experiment, the subjects listened to a story on a tape-recorder, and after having heard the input sentences, the tape-recorder was stopped and the probe sentence was presented (flashed on the screen). The subjects had to judge whether the probe sentence was true or not, in case it was true, they also had to judge whether it was in the same voice. In the delayed condition of the same experiment, the subjects heard the entire story (which takes about two minutes), and then were tested. The probe sentences occurred in the same order as they appear in the story, thus, each input sentence was tested with a two minute delay. Similar to the immediate condition, the subjects had to judge the truth of the sentences presented on the screen, and in case of true sentences they also had to verify the voice.

The second experiment tested isolated sentences, rather than sentences in a story. Memory for the form in the delayed condition of the first experiment was rather poor (56%);[23] thus, it was not clear whether the Auxiliary Encoding or the Verbatim hypothesis was favored. It was assumed that "subjects are freed from processing the theme of the story" in the task with isolated sentences; consequently, they can "devote more capacity to encoding deliberate propositions about sentence from"; thus, exhibit better memory for form (Anderson, 1974, p. 155). In this experiment, the Auxiliary Encoding Hypothesis predicts better memory for form in the delayed condition, and no interaction between the voice of the input and the voice of the probe in verification times. The Verbatim hypothesis, on the other hand, predicts better memory for form because there are fewer sentences, and greater interaction between the voice of the input and the voice of the probe in verification times.

The second experiment was intended to be as close to the first experiment as possible. In the immediate condition, a subject studied an input sentence for 15 seconds, and was presented with a probe sentence that was flashed on the screen by a slide projector (input sentences were written on cards). In the delayed condition, the subjects first studied all eight sentences, and then were presented with probe sentences. The order of probe sentences in the delayed condition was different from

[23] The results of the both experiments are presented in the following section.

the order of input sentences, to prevent the subjects from guessing sentences to follow. The delay between an input sentence and a probe was again two minutes. Similar to the first experiment, in both conditions the subjects had to judge the truth of the probe sentences, and if the probe is true, also verify the voice.

The sentences used in both experiments are the same. The example sets are given below:

Study set:

The painter visited the missionary.
The missionary refused the painter.
The painter was chased by the missionary.
The painter was protected by the sailor.
The missionary shot the sailor.
The cannibal questioned the painter.
The missionary was accused by the painter.
The cannibal was feared by the missionary.

Test set:

The painter visited the missionary.
The painter was refused by the missionary.
The missionary chased the painter.
The painter was protected by the sailor.
The sailor shot the missionary.
The cannibal was questioned by the painter.
The missionary accused the painter.
The missionary was feared by the cannibal.

4.1.2 Results

In the immediate condition of the first experiment there were almost no errors in both truth and form judgments (98% and 99% correct respectively). In the delayed condition, the subjects correctly verified 96% of the probes (truth judgments), but the memory for form was only 56% correct. In the second experiment, which tested memory for isolated sentences, the subjects correctly verified the truth of 93% and 90% of the probes in immediate and delayed conditions, respectively. However, as it was predicted, the memory for form in the delayed condition of the second experiment was better than in the first experiment (78% correct).

In both experiments Anderson (1974) measured reaction times for the probe sentences in each condition (errors were excluded from mean calculations). The results of the experiments showed that in immediate condition participants were faster in truth judgments if voices match, but in delayed condition it took longer to judge sentences presented in the passive voice (because the passives contain more words). The mean verification times for both experiments are shown in the tables below.[24]

Table 4.1: Mean verification times in the first experiment in Anderson (1974)

Input Probe	Active Active	Active Passive	Passive Active	Passive Passive
DELAYED				
True False	2.25 2.55	2.80 2.95	2.30 2.55	2.75 2.95
IMMEDIATE				
True False	1.20 1.55	1.80 1.75	1.80 1.80	1.45 1.75

Table 4.2: Mean verification times in the second experiment in Anderson (1974)

Input Probe	Active Active	Active Passive	Passive Active	Passive Passive
DELAYED				
True False	2.10 2.45	2.75 2.77	2.45 2.45	2.70 2.75
IMMEDIATE				
True False	1.20 1.55	1.75 1.75	1.90 1.75	1.45 1.80

[24] The mean reaction times for the immediate condition of the first experiment and the delayed condition of the second experiment are not precise, because they were taken from the graphs in Anderson (1974). However, the mean verification times for other conditions are precise, because they were relisted in the model of Anderson et al. (2001).

The results do not reject the Auxiliary Encoding hypothesis, but they support the Verbatim hypothesis that claims that verbatim images for sentences can be stored in long term memory and used for form judgments. Furthermore, the slow verification times for passives in the delayed condition support the idea that the propositional representation has an active-like structure. The reaction times in both experiments are close to each other for both conditions, but there are slight differences. For the immediate condition, the mean verification times are almost the same, which shows that the verification time is affected only by the truth of the probe and its voice. For the delayed condition, in the first experiment the subjects were about 0.2 second slower than in the second experiment.

As an explanation, the author proposed the use of two strategies – verbatim and propositional. Subjects first read a probe sentence. After that, if the input sentence is in short-term memory, they begin the execution of the verbatim strategy. If the representation of the input sentence is not in short-term memory, then subjects first might attempt to retrieve it from long-term memory; and, if successful, they will execute the verbatim strategy. The propositional strategy is deployed when the input sentence in not in short term memory. The subjects might deploy it after they fail to retrieve the input sentence from long-term memory, or they might not even attempt to retrieve it.

In the verbatim strategy (Olson & Filby, 1972; Garrod and Trabasso, 1973 in Anderson, 1974), "the verbatim sentence is entered into a comparison buffer where it is compared with the probe sentence the subject is viewing" (Anderson, 1974, p. 157). Anderson assumes that the comparison operation is purely syntactic, because in order to match the words of the input sentence to words of the probe sentence, subjects do not need to access the semantics of words. Another assumption is that "a response index is used to keep track of the number of mismatches" between the input and probe sentences (Anderson, 1974, p. 158). This index is initially set to true (i.e. subjects expect that two sentences will match), and it is changed after each mismatch. During comparison, subjects go serially through the input sentence and match each word to the probe sentence. If the subjects of two sentences mismatch, the response index is set to false. Since all the false sentences in the experiment were constructed by switching subjects with objects, Anderson assumes that there is no comparison done after the verb of the sentence (because the truth of the sentence can be decided without matching any further), and the response may be executed.

In the propositional strategy (Anderson & Bower, 1973), the probe sentence is first transformed into propositional format, and then matched to the information in long-term memory. If the propositional representation of the probe matches any proposition in long term memory, it is regarded as true; otherwise it is regarded as false. Similar to the verbatim strategy, a response index, which is initially set to true, is used; and after the matching operation, which can change the response index if there is a mismatch, the response in executed.

In the immediate condition, the probability of selection of the verbatim strategy is higher; thus, Anderson (1974) assumed that the subjects always used it in this condition. Whereas in the delayed condition, the probability of choosing the verbatim strategy is lower (but possible) than the probability of choosing the propositional strategy; thus, the subjects mainly used the propositional strategy, but sometimes used the verbatim strategy in this condition. The difference between the results in the delayed condition of the experiments does not suggest that paragraphs are processed differently from isolated sentences; rather it reflects the "proportion of verbatim images available in the delayed condition" (Anderson, 1974, p. 162).

For the model developed in this thesis the delayed condition of the second experiment was selected, and the propositional strategy was modeled. The reasons for the selection of this experiment and the delayed condition are discussed in Section 4.3.1.

4.2 *Previous models of Anderson (1974)*

There are two existing ACT-R models of Anderson (1974) – Anderson, Budiu, and Reder (2001), which was mentioned in Section 3.5, and Lebiere (2002), which was presented in 24th Annual Conference of the Cognitive Science Society as a part of the introduction to ACT-R 5.0. Anderson et al. (2001) modeled both conditions, immediate and delayed, whereas Lebiere (2002), similar to the model developed in this thesis, considered just the delayed condition of the experiment. This section discusses the models of Anderson et al. (2001) and Lebiere (2002) with respect to the representation and the processing that they used.

4.2.1 Anderson et al. (2001)

Anderson, Budiu, and Reder (2001) described a theory of sentence memory as a general theory of memory, and developed an ACT-R model for it. Among other experiments, the experiment of Anderson (1974) was also modeled. This subsection overviews lexical item and sentential representations they used, and the processing they proposed for the experiment of Anderson (1974).

Lexical items, in the model of Anderson et al. (2001) are represented by chunks like:

```
The-waiter-n isa lex-entry
     type    noun
     word    the-waiter
     meaning *waiter*
```

Meaning of a lexical item, on the other hand, is represented by a chunk like:

```
*waiter* isa chunk
```

Such kind of representation is referred to as *atomic representation*, since both lexical items and their meanings are represented by a single chunk. Noun phrases like *the waiter* are treated as a single unit; consequently, there is no processing for determiners.

As it was mentioned in Section 3.5, Anderson et al. (2001) have used the *fragmented representation* for the sentential information (i.e. the representation of a sentence is distributed among several chunks). The semantic representation they have used is shown in Figure 4.1.

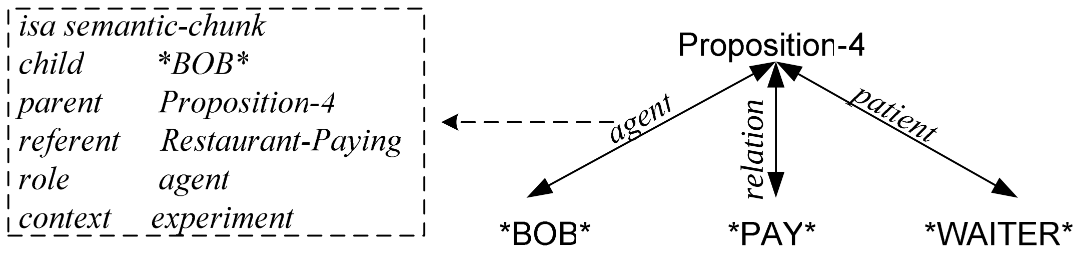


Figure 4.1: Propositional encoding (Anderson et al., 2001, p. 340)

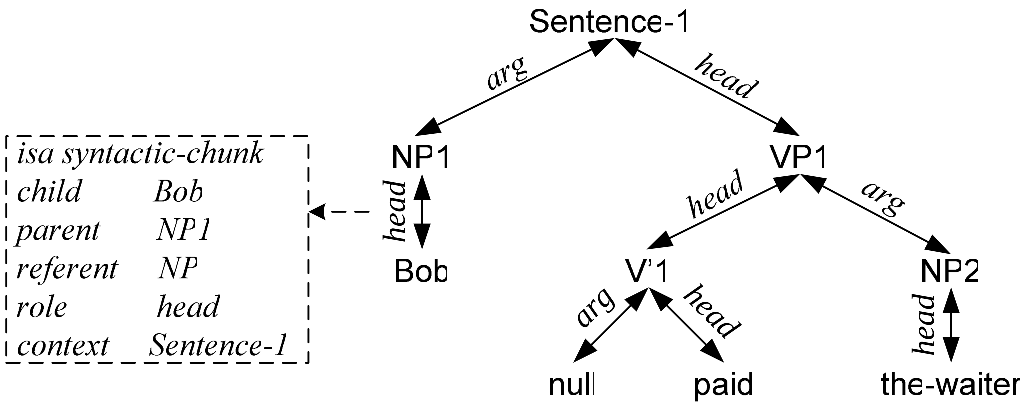


Figure 4.2: Syntactic encoding (Anderson et al., 2001, p.340)

Since they have also modeled the immediate condition, there is also a syntactic encoding, which is similar to the propositional encoding, but also contains syntactic hierarchy information. For the same sentence *Bob paid the waiter*, the syntactic representation is the one that can be seen on Figure 4.2. Their model constructs the semantic and the syntactic representations in parallel; it adds a new link both to the propositional and to the syntactic trees after reading a word. These representations are used later, in either verbatim or propositional strategies, which were discussed in Section 4.1.2. The role assignment is done with respect to syntactic processing, after having read a word the model retrieves its syntactic category and constructs a part of the syntactic parse tree with respect to phrase structure rules implemented as productions. The model is biased to believe that first noun is an agent of the proposition; consequently, the role of an agent is assigned to the first noun, without waiting for a verb.[25] In the case of passives, when the model reads an auxiliary, it retrieves the link that encodes that the first noun was an agent, and changes the role to patient.

According to Anderson (1974) and Anderson et al. (2001), in the delayed condition the subjects mostly used the propositional strategy. The processing they proposed for the propositional strategy of the experiment in Anderson (1974) is schematically represented in Figure 4.3:

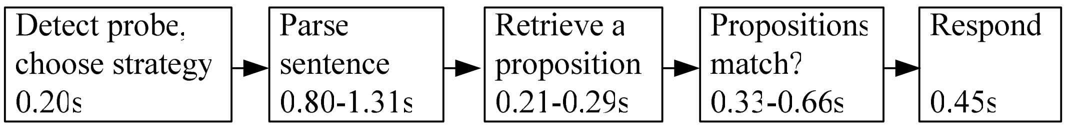

Detect probe, choose strategy 0.20s
Parse sentence 0.80-1.31s
Retrieve a proposition 0.21-0.29s
Propositions match? 0.33-0.66s
Respond 0.45s

Figure 4.3: Propositional strategy (Anderson et al., 2001, p. 350)

[25] Bias here does not mean that there are two options for the role assignment and one of them is selected with respect to some probability; rather it means that subjects automatically assigned the role of an agent to the first noun phrase.

The figure represents actions of the model in Anderson et al. (2001), when the virtual subject[26] chooses the propositional strategy. It also contains the range of times for each step; the time varies with respect to the delay and the voice of the probe sentence. The performance of this model with respect to the delayed condition is discussed later, in Section 4.4.

4.2.2 Lebiere (2002)

Lebiere (2002) considered only the delayed condition of the experiment and used atomic representation for both, lexical items and propositions. Lexical items were represented by chunks like:

```
painter isa  meaning
        word "painter"
```

The model of Lebiere (2002) does not construct syntactic representation; consequently, such kind of chunks does not contain lexical class information like in the model of Anderson et al. (2001). The meaning of a sentence is represented by chunks like:

```
Goal1 isa     comprehend-sentence
      agent   painter
      action  visit
      object  missionary
      purpose
      word
      state
```

The model of Lebiere (2002) was intended to be very simple. For example, for the sentence *the painter visited the missionary*, it skips articles, and the first word (which is not *the*) is automatically assigned the agent role. Similarly, the second and the third content words are assigned to the slots *action* and *object*, respectively. In

[26] The term virtual subject was used to differentiate between real subject (humans) who actually attended experiments, and subjects that were simulated on ACT-R.

case of passives, when the model reads words *was* or *by*, it exchanges the values of the agent and object slots. The slot *word* is filled by the character string of the word that was read last. The slots *purpose* and *state* are used for the control of processing; the former contains the information about the phase of the experiment (i.e. whether the sentences presented on the screen are to be studied or to be verified), the latter keeps track of the current state of the goal (i.e. keeps information like "the word was read" and "the sentence is done").

The propositional strategy for the delayed condition is exactly the same with Anderson et al. (2001), which can be seen on Figure 4.3. However, since Lebiere (2002) used atomic representation for sentential information, unlike Anderson et al. (2001), his model retrieves a single chunk for matching. This significantly speeds up the model. The performance of this model is also discussed in Section 4.4 together with the model developed in this thesis and the model of Anderson et al. (2001).

The models are similar in that they both use atomic lexical item representations; and they differ in that the model of Anderson et al. (2001) uses fragmented propositional representation, whereas the model of Lebiere (2002) uses atomic propositional representation. Another similarity is their implementation of the propositional strategy for the experiment of Anderson (1974).

Anderson et al. (2001) and Lebiere (2002) both modeled the sentence verification task with isolated sentences (the second experiment of Anderson (1974)). However, the mean verification times they used for the comparison with simulation results for the delayed condition of their models are the results of the delayed condition of the first experiment. This is justified by the fact that in the delayed condition of the first experiment the propositional strategy was used more often (almost always) than in the delayed condition of the second experiment. Consequently, the mean verification times of the delayed condition of the first experiment reflect the timing of the propositional strategy clearer.

The model developed in this thesis resembles the models of Anderson et al. (2001) and Lebiere (2002) in that it also uses the mean verification times of the delayed condition of the first experiment to simulate the propositional strategy of the second experiment. Furthermore, the model of this thesis simulates some perceptual-motor actions required for the experiment (e.g. reading), similar to the model of Lebiere (2002). Another similarity between the model of this thesis and the model of Lebiere (2002) is that only the delayed condition was considered.

4.3 Model Design

In this thesis the experiment of Anderson (1974) was implemented on ACT-R cognitive architecture, and the model was compared to the previous implementations of Anderson et al. (2001) and Lebiere (2002). The scope and the aim of the model with respect to the experiment of Anderson (1974) are discussed in Section 4.3.1.

In order to simulate the experiment, some functions have to be defined for the presentation of the sentences and the collection of virtual subjects' responses. The functions defined for the model are presented in Section 4.3.2. Furthermore, since the sentences in the original experiment were presented visually, there are production rules that simulate the reading process. The responses of the virtual subjects are key presses, which are also simulated by production rules. The discussion of the production rules that simulate these actions is presented in Section 4.3.3.

In order to model the experiment of Anderson (1974), like the models of Anderson et al. (2001) and Lebiere (2002), the representations for lexical items and sentential information have to be developed. Furthermore, there have to be some rules for sentence comprehension (i.e. in order to construct a propositional representation, an agent and a patient of the sentence have to be identified). The structures used for the representations of lexical items in the model are presented in Section 4.3.4. How these structures are processed to yield understanding is explained in Section 4.3.6. Section 4.3.8 presents the representation of sentential information that emerges as the result of processing using the designed representation.

Since Cognitive Grammar makes an emphasis on the notions of *ground* and *mental space* (see Section 2.1.5), in the model they are used to represent speech act settings and the current discourse, respectively.[27] The representation of these notions and their processing are presented in Sections 4.3.5 and 4.3.7, respectively.

4.3.1 The aim and the scope of the model

The primary aim of this thesis is to show that the structures provided by ACT-R for the representation of cognitive abilities meet the requirements of Cognitive Grammar. Another aim of our model is to show that the representation and processing developed using these structures can be used to simulate human behavior on the sentence verification task of Anderson (1974) (i.e. the reaction times from the ACT-R simulation of the experiment are comparable to the reaction times obtained by Anderson (1974)).

Cognitive Grammar differs from the linguistic theories that are usually used for the interpretation of the results of psychological experiments. Consequently, it is necessary to interpret the results of the experiment of Anderson (1974) in the light of

[27] Speech act here means any kind of situation in which some linguistic input was received, and speech act setting means the general context in which this input was received, i.e. the speakers, the time, and the place. Discourse here means the topic of the input, i.e. linguistic context.

processes and structures proposed by Cognitive Grammar. The following paragraphs present the interpretations of the key results of the experiment and the assumptions these interpretations lead to. They are necessary to clarify the reasons for the selection of the minor goals and the scope of the model.

As far as I know, there was not any attempt to interpret the results of the experiment of Anderson (1974) from the standpoint of Cognitive Grammar. Even the fact that the memory for meaning of a sentence is much better than the memory for form should be clarified. Anderson (1974) and Anderson et al. (2001) assumed that the propositional representation of passive sentences is the same with the propositional representation of active sentences. According to Langacker (1987, 1991, 1999a), proposition is a content evoked by an expression; thus, active and passive sentences evoke the same content (i.e. designate the same process). However, in Cognitive Grammar, the meaning of an expression is not only the content an expression evokes, but also the construal of this content (see Section 2.1.2). Consequently, meanings of active and passive sentences are not the same; they differ with respect to the trajector/landmark organization the sentences impose on the process they designate.

If memory for meaning presupposes all semantically relevant information, then, from the standpoint of Cognitive Grammar, the figure/ground organization[28] should be remembered as well as the content (i.e. proposition). However, the fact that in Anderson (1974) the subjects correctly verified the form of 56% and 78% of sentences in the delayed condition of the first and the second experiments,

[28] Trajector/landmark organization is a kind of figure/ground organization. Relations (processes) generally have one or more focal elements. The trajector is primary figure (focal element) and the landmark is the secondary figure of the relation.

respectively, while they were almost always correct in the truth judgments, clearly illustrates that the figure/ground organization, which is the difference between active and passive sentences, is not remembered as well as the content.

In Section 2.1.2, it was stated that lexical items evoke some conceptual content (the base of the lexical item) and impose certain figure/ground organization on it. Consequently, in order to accommodate the results of Anderson’s (1974) experiment, we had to make the following assumption: The final representation of a sentence in long-term memory contains no trajector/landmark organization; rather it is used to construct this final representation. This assumption leads to the prediction that the groups of sentences like the ones shown below will show the same effect as active/passive pairs.

1. (a) The picture is above the clock.
   (b) The clock is below the picture.

2. (a) The man gave the book to me.
   (b) The man gave me the book.
   (c) I received the book from the man.

The sentences in these groups profile the same conceptions, and the only difference between the sentences in each group is the figure/ground organization.

Consequently, the model should construct a sentential representation that is independent from construal. However, construal should be evident in processing. (Although the prediction was not tested in this thesis, the model was designed to accommodate it.)

The immediate condition in the experiments of Anderson (1974) showed that the memory for form is perfect without a delay. The nature of the verbatim trace, which is assumed to be the immediate memory for sentential information, is not explicit in Cognitive Grammar. The verbatim strategy, which was proposed to be used in the immediate condition by Anderson (1974), is assumed to be purely syntactic. Thus, the subjects do not access meanings of words in this strategy. However, according to Cognitive Grammar this could not be the case, because when a sentence is read, people necessarily access its meaning. The lack of clarity from the standpoint of Cognitive Grammar on this issue, and the unwillingness to rebuild the theory to accommodate these results are the reasons for leaving the immediate condition (thus, the Verbatim strategy) out of the scope of our model.

The delayed condition of the second experiment was modeled, and following the assumption of Anderson et al. (2001), the results of the delayed condition of the first experiment were selected as reaction times that reflect the propositional strategy. However, we assumed that the propositional strategy does not require the construction of the propositional representation of the probe sentence prior to the search for the matching proposition in long-term memory (as it is the case in the models of Anderson et al. (2001) and Lebiere (2002)); rather the subjects search for the candidate proposition after having read each content word. This assumption makes the propositional strategy more similar to the natural reading process, and it is justified by the fact that the primary goal is not to encode the sentence, but rather to test whether it was implied before or not.

To sum up, the model developed in this thesis aims to design the representation for lexical items and sentence processing that will meet the requirements of Cognitive Grammar and ACT-R. Moreover, the model aims to show that the designed representation and processing can be used to simulate human behavior on the sentence verification task of Anderson (1974). The model is also intended to accommodate the assumptions that construal is used in processing but it is not part of the final representation; and that in the propositional strategy the subjects do not construct another representation of the probe sentences, but rather they try to find the matching proposition after each content word.

The scope of the model of this thesis is limited to the delayed condition of the second experiment; and since only the propositional strategy was considered, the mean verification times of the delayed condition of the first experiment of Anderson (1974), which reflects the timing of this strategy clearer, were selected as the data to be matched.

4.3.2 Simulation of the experiment: functions

Anderson's (1974) experiment was simulated using the standalone version of the ACT-R environment that was presented in Section 3.4. As it was mentioned in that section, the basic functions for the design of simple experiments come together with the environment. These basic Common Lisp functions were used to simulate the way sentences (the study and test sets on page 72) were presented to subjects and the way subjects' responses were collected. This section describes the main functions that were defined for the model (i.e. those not predefined by ACT-R) with respect to what they do.[29]

The function *study-sentence* is used to display a single sentence in the window it creates. It takes two arguments – the sentence to be displayed and the time to keep the sentence in the window. All the functions used by this function are predefined by ACT-R. The other function *test-sentence* is very similar to the function *study-sentence*, but it also collects the responses of the virtual subjects. The functions

[29] All functions are available in Appendix.

study-sentences and *test-sentences* use functions *study-sentence* and *test-sentence* to present the whole the study and the test sets of sentences, respectively.

The whole experiment is simulated by the function *do-delayed*, which first presents the sentences of the study set by the function *study-sentences*, and then presents the word "test" by the function *study-sentence.* The word "test" is presented to make the virtual subjects change the purpose of reading from study to test (the goal chunk, which is discussed in the following section, contains the slot *purpose*). After the word "test", the test set is presented by the function *test-sentences*, which is the argument of another function *report-data.* The function *report-data* analyzes the responses of the virtual subjects with respect to the data obtained by Anderson (1974) from the original experiment (the original data is defined as a constant). As it was mentioned in Section 3.4, the results are analyzed in terms of the correlation and the mean deviation between two sets of values.

4.3.3 Perceptual-motor actions in the model

The simulation of basic perceptual-motor actions is necessary because they make the model more precise. ACT-R's perceptual-motor component provides basic actions like the movement of attention and key pressing; furthermore, it specifies the time it takes to perform these actions. The time for each action is an estimation drawn from the psychology literature. Since the model aims to simulate the human performance on the sentence verification task as close as possible, basic processes of reading and key-pressing were simulated.

Besides providing the basic perceptual-motor actions and their timing, ACT-R specifies how they must be used in modeling; consequently, the processes of reading and key-pressing are analogous across all ACT-R models. Thus, the basic reading process simulated in the model of this thesis is similar to the reading process used by Lebiere (2002).

The sentences are presented in the window created by the function provided by the ACT-R environment. This window is used to simulate both the cards used for the presentation of sentences during the study phase in the original experiment by Anderson (1974), and the visual screen on which the probe sentences were flashed out.

The virtual subjects[30] start the experiment with the goal buffer containing the chunk presented below.[31]

```
goal isa lwm
   context MS          ; Mental Space (Experiment)
   purpose "study"
   state   "attending"
```

It is required by the ACT-R architecture for the goal buffer to contain some chunk, which represents the current goal of the model (i.e. goal of the virtual subject), and the initial production is selected with respect to this chunk. The chunk above represents that virtual subjects are aware of the fact that they are participating in an experiment and their purpose is to study sentences that will appear in the window. Although there are other slots in the goal chunk, they are initially empty (i.e. they have the value *nil*).[32] The virtual subjects will attend to the leftmost word of the

[30] As it was stated earlier, the term virtual subject is used to refer to the subjects that were simulated in ACT-R. Declarative memory chunks and production rules are contents of the declarative and procedural memory of virtual subjects, respectively. Thus, a model run is a simulation of a virtual subject's behavior on the experiment.

[31] The chunk type is named *lwm* to stand for *linguistic working memory,* which is the term used by Jackendoff (2002) to refer to the part of memory where the linguistic processing is going on. However, the name does not have any connection to Cognitive Grammar. It was just assumed that the name will be appropriate for a goal chunk, because it is used to construct meaning.

[32] The other slots of the goal chunk are discussed later in the relevant sections.

sentence by default. This reflects the fact that subjects know where the sentence begins. This is achieved by the command:[33]

```
(pm-set-visloc-default :attended new :screen-x lowest)
```

The area of attention is represented by a visual-location chunk, which is specified by the perceptual-motor component of ACT-R. This chunk is placed into the visual-location buffer (see Section 3.1.3).

After the sentence is displayed in the window, the virtual subject begins reading. The process of reading is simplified; it is totally bottom-up and does not account for any top-down processes. For the first word in a sentence the virtual subjects apply the production rule[34] given below.

```
(p new-word
   =goal>
      isa        lwm
      state      "attending"
   =visual-location>
      isa        visual-location
   =visual-state>
      isa        module-state
      modality   free
==>
   -manual>
   =goal>
      isa        lwm
      state      "processing"
   +visual>
      isa        visual-object
      screen-pos =visual-location)
```

It is the only production rule that matches the initial contents of the buffers (visual-location and goal buffers). The production rule requests the identification of the object on the screen at the location specified by the visual-location chunk (i.e. requests the chunk that encodes that object to be placed into the visual buffer).

[33] Commands are also basic Common Lisp functions that come together with the environment. They are referred to as commands in order to avoid the confusion: functions are used for the simulation of the experiment, whereas commands are used to specify the state of the virtual subject.

[34] The syntax of production rules is presented in Section 3.2.2.

The next production rule to apply is `read-word`.[35]

```
(p read-word
   =goal>
      isa      lwm
      state    "processing"
   =visual>
      isa      text
      value    =word
==>
   -visual-location>
   =goal>
      isa      lwm
      state    "encoding"
   +retrieval>
      isa      s-link
      orth     =word)
```

It was mentioned in Section 3.1.3 that ACT-R's visual system encodes objects on the screen in form of chunks that represent their features. Visual objects of the type *text* are words. There are two possibilities: in the first case, a word is represented by a single chunk (feature); in the second case, a letter is represented by a number of features. The model makes use of the first case, because it is assumed that proficient readers do not attend to specific features of letters to identify them, but rather attend to words as a whole; consequently, the visual object that will be placed into the visual buffer is the first word of a sentence. The chunk is of the type text, and its value is a string of letters. The action side of this production rule requests the retrieval of a chunk with respect to this string. The chunk to be retrieved is the part of the representation that was developed for lexical items, and it is discussed in the following Section 4.3.4.1.

The first word of a sentence is processed differently from the other words, because the visual location of the word is specified without reference to any other object on the screen. However, the locations of the other words are specified with

[35] Recall that the words preceded by a '=' sign are variables.

respect to the location of the preceding word. Consequently, there is a need for another production that processes non-first words, which is `find-next`:

```
(p find-next
   =goal>
      isa      lwm
      state    "find-next"
==>
   +visual-location>
      isa      visual-location
      screen-x greater-than-current
      nearest  current
   =goal>
      isa      lwm
      state    "attending")
```

After the encoding of the first word, the virtual subject requests the location of the next word. When the visual system provides that location, the virtual subject attends to it. The contents of the buffers after the execution of this production rule will be the same that allow the selection of the production rule `new-word`; consequently, it will be applied next. These three production rules simulate the basic reading process.

Besides reading the sentences, the virtual subjects also press keys to denote whether a sentence presented during the test phase is true or false (i.e. whether it was implied by a study sentence or not). If the sentence is true, the virtual subjects press the key “k” on the keyboard, if it is false, they press the key “d”. The key presses are simulated by the production rules `respond-yes` and `respond-no`. The key that is pressed depends on whether the virtual subjects detected any error or not. The process of sentence verification and types of errors virtual subjects may detect are discussed in Section 4.3.9. Since the only difference between two productions is the pressed key and the error index, only `respond-yes` production rule is illustrated here.

```
(p respond-yes
   =goal>
      isa            lwm
      context        =ctx
      error          nil
      state          "answer"
   =manual-state>
      isa            module-state
      modality       free
==>
   +manual>
      isa            press-key
      key            "k"
   +goal>
      isa            lwm
      context        =ctx
      purpose        "test"
      state          "attending")
```

The production rule applies if the motor module is not busy (i.e. not performing another action). As it was mentioned in Section 3.1.3, some motor actions like key-presses and mouse click are already defined in ACT-R; consequently, it is sufficient to request the motor module to press a key and specify the key, which are done in the `+manual>` request. The time of the key press and its value (i.e. whether it was "d" or "k") are collected by the predefined function *rpm-window-key-event-handler* for the analysis later.

After pressing the key, the virtual subject prepares for the next sentence (i.e. introduces a new goal by `+goal>`): moves the attention to the place where the first word of the next sentence will appear, and leaves (practically copies) the context and the purpose unchanged.

These basic perceptual-motor actions are necessary for better simulation, because every action like the movement of attention and key pressing take certain amount of time, which is specified by the ACT-R and an estimation drawn from the psychology literature. These times are included in the reaction times measured by

Anderson (1974); consequently, the modeling of these processes forces the model to be more precise.

4.3.4 Representation of lexical items

The representation used for lexical items and grammatical rules affects the execution pattern of any model. In case of Cognitive Grammar, the representation is the main part of the model, since the grammar of a language is seen as an inventory of linguistic units (Langacker, 1987). In this case, language production is simply the search in the inventory for the appropriate linguistic units that best express the intended idea and their assembly according to some established pattern, which itself is a linguistic unit.

The lexicon of the model developed in this thesis consists of the words that appear in the sentences (see page 72) used by Anderson (1974) in the experiment. Thus, the model contains only the chunks that are necessary for the characterization of these words. The lexical item representations used by Anderson et al. (2001) and Lebiere (2002) do not meet the requirements of Cognitive Grammar, because, as discussed in Section 4.2, in the model of Anderson et al. (2001), lexical items contain syntactic category information, and neither of the models considers the assumption of Cognitive Grammar that the meaning of a lexical item is in the relation between its profile and base (see Section 2.1.2). Consequently, a new representation has to be developed. This section discusses the structures developed for the characterization of lexical items that will meet the requirements of both Cognitive Grammar and ACT-R. How these structures are used in the processing is discussed in Section 4.3.6.

According to Langacker (1987, 1991, 1999a) there are three types of linguistic units: phonological (or orthographic), semantic (concepts) and symbolic, which is an association between previous two structures. And there are three kinds of relationships between these units: *symbolization* – a relationship between phonological and semantic structures, *categorization* (or coding) – a relationship between two units (phonological or semantic) in which one structure is characterized as being an instance of another structure, and *designation* – a relationship between two semantic structures in which one structure functions as a knowledge base (base or domain) for characterization of another one. All these relationships are special cases of general cognitive abilities of association, categorization and foreground/background organization, respectively, which were discussed in Chapter 2.

Besides the theoretical requirements of Cognitive Grammar and ACT-R's representation of declarative and procedural units of knowledge in terms of chunks and productions, there are constrains imposed by subsymbolic level of ACT-R like the fact that two chunks are associated and activate each other only if they occur together in another chunk. Furthermore, over time ACT-R's community developed representational conventions that account for some phenomena such as errors of omission and commission.[36] The theory of declarative representation of knowledge developed by Anderson et al (1998) requires the representation to be fragmented, that is, divided into several chunks. The representation that meets all the constraints is

[36] Errors of omission arise due to the inability to retrieve any chunk that meets retrieval criteria, and errors of commission arise due to the retrieval of the wrong chunk, since the correct one is below the threshold of activation. With respect to sentence processing the inability to recall a participant in a proposition would be an error of omission and the recall of wrong participant would be an error of commission.

presented in terms of relationships that can exist between linguistic units – symbolization, categorization and designation.[37]

4.3.4.1 Symbolization

Symbolization is the relationship between phonological and semantic (or conceptual) units. In the model it is retrieved to allow the access from written marks to concepts they symbolize. The relationship is represented by a chunk such as:[38]

```
s0001
   isa    s-link   ; symbolic link
   orth   "knife"  ; orthography / phonology
   sem    knife*   ; semantics
```

The only information contained in this type of chunk are orthographic or phonological structures and the concept they symbolize. Although it is conventional to put grammatical category information in such kind of chunk (e.g. Anderson et al. (2001)), it should contain no such information because the classification of entities into things and relations (nouns and verbs) is semantic.

4.3.4.2 Categorization

Another kind of relationship that can be held between two linguistic units is the categorization relationship. The process of categorization was discussed in Chapter 2, and requires two entities – standard and target of categorization. The relationship is equivalent to type-token and schema-instance relationships and can be held between all kinds of linguistic units – phonological, semantic and symbolic. It is represented by a chunk such as:

[37] Although there is also a composition relationship in the semantic network that is held between symbolic units that are used in the one structure, this relationship is not considered here because the model makes use of the part-whole links, which are discussed bellow, and they suffice to represent the composition relationship.

[38] The sign '*' after a word represents the concept (profile of the expression), whereas words between two such signs represents bases of concept (i.e. abstract domains).

```
c0001
   isa    c-link   ; categorization link
   inst   knife*   ; target of categorization
   type   p-obj*   ; standard of categorization
   ctx    world    ; context of categorization
```

This chunk encodes that *knife* is classified as a *physical object* in the context of *world* (i.e. world knowledge). There is a possibility of multiple category memberships, and categorization is done in some context (Medin, 1989); consequently, the context slot is used to allow this possibility. There are two types of categorization relationships – between units in long-term memory and between a unit in long-term memory and a new event. The former one is a classification, which is a part of lexical meaning. the latter one appears when, for example, we actually see some object and identify that object as an instance of some category. The context of categorization in this case is actual usage instance. In the model the categorization relationship is used to structure the lexicon, as well as for the representation of the type-token relationship between the entities in mental space (discussed in Section 4.3.5) and the concepts in long-term memory.

4.3.4.3 Designation

Lexical items designate some parts of the bigger knowledge structures; in other words, lexical items *profile* some part within its *base*.[39] In the model developed in this thesis the designation relationships are divided into two types – *domain specification* and *part-whole* relationships, which are represented by different chunk types. The representation and the function of each relationship in the model is presented below.

[39] The profile is the conceptual referent of the expression, whereas the base is the conceptual content evoked by the expression.

Domain specification relationships are used to specify the properties of an entity in some basic domain. For example, the chunk given below specifies the value of a knife in the basic domain of color.

```
d0001
   isa    d-link    ; domain specification link
   ent    knife*    ; entity
   domain COLOR     ; domain
   value  black     ; value in the domain
```

An entity is usually specified in several basic domains simultaneously. For example, *knife* is also specified in the domain of space for its shape. The domain specification relationships form a network centered on the entity that is specified.

Although there are chunks that represent domain specifications in the model of this thesis, they are not used in the processing. With respect to the developed representation, the similarity between two entities is the number of the domain specification links that has similar values in the slots *domain* and *value*. However, the matching mechanism of ACT-R can only calculate the similarity between two chunks. Consequently, because the model developed in this thesis does not require precise similarity, the categorization relationship is used. In this case, two entities are similar if they have the same superordinate (i.e. the same value in the slot *type* of the categorization link chunk).

Part-whole relationships are used to represent the relationship between a profile and the base of a lexical item (i.e. partonymy between two entities). It is the most used relationship in the model, because for every lexical item a part-whole link is retrieved. The chunk given below represents the part-whole relationship between concepts *blade* and *knife*:

```
pw0001
   isa    pw-link   ; part-whole link
   part   blade*    ; part
   whole  knife*    ; whole
   cfg    ?         ; configuration or role
```

Configuration is a relative assembly between a part and a whole. With respect to the blade-knife example, it is the way the blade is attached to the handle.

In the example above, *knife* functions as an abstract domain for *blade*. Since any knowledge structure can function as an abstract domain, there is a network formed by part-whole relationships in the lexicon. For example, the wholes for *knife* are *cut* and *silverware*. *Cut*, which is the process, is also represented by part-whole links. Parts of the thing, like *knife*, are things themselves, whereas parts of the process, like *cut*, are its participants. The process of cutting is represented by chunks shown below:

```
pw0034              pw0035              pw0036
   isa   pw-link       isa   pw-link       isa   pw-link
   part  cut-AG*       part  knife*        part  cut-PAT*
   whole *cut*         whole *cut*         whole *cut*
   cfg   AG            cfg   INSTR         cfg   PAT
```

The chunks encode that the process of cutting has three participants which are *cut-AG**, *cut-PAT**, and *knife** (although it could be any sharp entity, the *knife* is assumed to be the default instrument). Their relative assembly is that *cut-AG** is an agent, *cut-PAT* *is a patient, and *knife** is an instrument of the process.

4.3.4.4 <u>Domains</u>

Concepts are represented by schemas formed by part-whole links. Since one concept can function as a domain for another, abstract domains for some concepts are represented by a small number of part-whole links (e.g. cut). However, the representations for other abstract domains like politics and religion were not explored in the model, because, although they can be represented by part-whole links, the number of links will be too big to be included in the scope of this thesis. Consequently, only some links of these domains are specified.

Although basic domains can be uniformly analyzed in terms of dimensions (a number of axes needed for the representation of a concept in a domain, e.g. color has three dimensions – brightness, hue, and saturation) they have, there is no uniform representation for them. Since they are basic experiences and do not exhibit declarative knowledge features, they cannot be represented by schemas like abstract domains. The only way to implement dimensions of basic domains is to integrate them into ACT-R's perceptual-motor extension. A number of them is already integrated; for example: ACT-R's visual module processes visual field and stores features like color and location in the form of declarative memory chunks, and the chunk representing color features, for instance, specifies color without specifying its dimensional values. Furthermore, the knowledge of dimensional values of the color domain is not part of common knowledge (e.g. what are hue, saturation and brightness values of *red*?). However, the domain is partitioned into regions with respect to dimensions, and these regions have labels; consequently, it is reasonable to use these labels as domain values instead of specifying dimensional values.

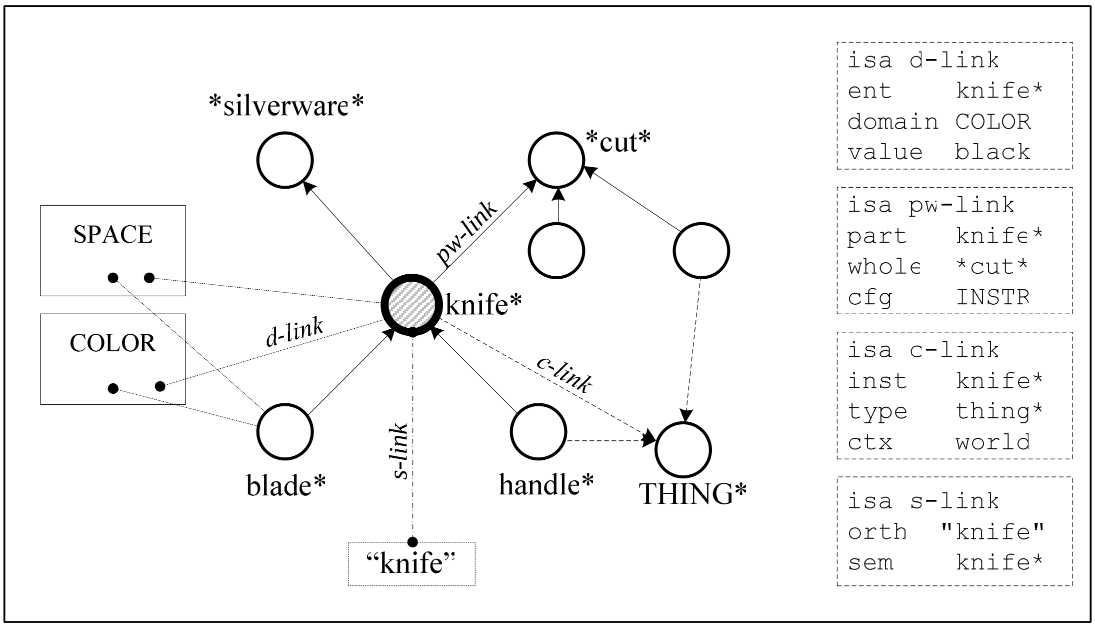
There was no attempt in this thesis to integrate dimensions into ACT-R. Instead, basic domains are represented by chunks such as:

```
TIME  isa domain
SPACE isa domain
COLOR isa domain
```

4.3.4.5 Ordering of relationships

The subsymbolic level of ACT-R adds another aspect of meaning proposed by Langacker (1987, 1991, 1999a) – ordering of domains according to their saliences. The centrality of the domain depends on the frequency of its activation together with the profile. The frequency of use determines the base-level activation of the chunk in ACT-R; consequently, the most central domain (see Section 2.1.2) (i.e. chunk representing domain specification or part-whole relationship) will have the highest activation and activated in most of the cases.

Since one schema can function as a part in a number of other schemas, such a representation will eventually end up in a multilevel semantic network, with a concept being an access node to it (since it is the only element that participates in all the relationships, and having accessed this element it is possible to reach any structure in the network). There are taxonomy, part-whole, domain specification and symbolization levels created by corresponding relationships. The network created for a single lexical item by these relationships is exemplified in Figure 4.4:



```
isa d-link
ent    knife*
domain COLOR
value  black

isa pw-link
part   knife*
whole  *cut*
cfg    INSTR

isa c-link
inst   knife*
type   thing*
ctx    world

isa s-link
orth   "knife"
sem    knife*
```

Figure 4.4: Types of relationships between linguistic units

In Figure 4.4 the profile is a thick circle at the center labeled *knife**, which stands for *knife* schema. It functions as an access node to the network of related concepts. It is connected to its base, *cut* schema (circle labeled **cut**), by the part-whole link (pw-link). All the schemas are specified in several basic domain (rectangles labeled COLOR and SPACE.), and these specifications are represented by domain specification links (d-links). In the figure, *knife* is categorized as an instance of a *thing* schema (circle labeled *THING**), these relationships are represented by categorization links (c-links). The symbolic link between concept *knife* and the string of letters “knife” is represented by symbolic link (s-link). All the links connected to *knife* schema are activated when it becomes a slot value of the goal chunk.

The network created by relationships of symbolization, categorization, part-whole, and domain specification and enhanced by ACT-R’s activation calculus explains linguistic phenomena like semantic closeness, taxonomy, vagueness, and polysemy (although they are not in the scope of this thesis; thus, they were not explored). The network is easily updated; it is dynamic in the sense that meanings can change and new meanings can be added.

As already mentioned, the model developed in this thesis is limited to processing of the sentences used in the experiment of Anderson (1974). Thus, symbolization, categorization, domain specification, and part-whole links are specified only for the words that appear in Anderson (1974). There are four things: *painter*, *missionary*, *sailor*, and *cannibal*; and eight processes: *visit*, *refuse*, *chase*, *protect*, *shoot*, *question*, *accuse*, and *fear*. Besides these, there are words *was*, *the*, and *by*; their functions are discussed in Section 4.3.6.4.

4.3.5 Representations of Ground and Mental Space

The notions of *ground* and *mental space* were presented in Section 2.1.5. In the model they are used to represent speech act settings and the linguistic context, respectively. These notions make the model applicable to the tasks other than processing of isolated sentences, and reflect the process of natural text or speech comprehension. Each of the notions is represented by a number of part-whole and domain specification links. Their schematic representation can be seen on Figure 4.5, where the dashed lines represent part-whole links, and the straight lines represent domain specification links.

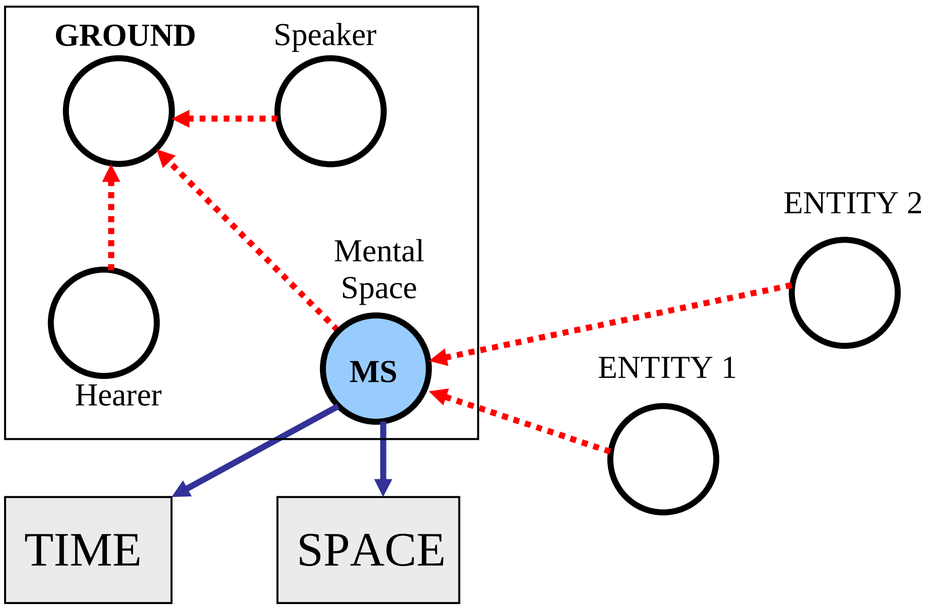


Figure 4.5: Ground and mental space

The direction of arrows represents the structure of the chunks that encode the relations between the elements of these structures. For example, the part-whole relationship between the ground and mental space is represented by the chunk given below.

```
Gr003
   isa   pw-link
   part  MS          ; Mental Space
   whole Ground
```

The direction of the arrow is from a part to a whole, and from an entity to a domain.

Since ground is speech act setting, it is specified in the domains of time (time of speech) and space (where it takes place). Moreover, both speech act participants –

speaker and hearer – are connected to the ground by part-whole links. Other part-whole links connect the objects in the environment that both speaker and hearer are aware of and that have any relevance to the dialogue.

Mental space is part of the ground as well; thus there is a part-whole link that represents their relationship. Mental Space is very similar to the ground: it also has time and setting, which may be different from the time and location of the ground (e.g. in a novel), and it is populated by the objects and relations between these objects, which the dialogue or text is about. Since the representation of mental space in the model is fragmented, it can be updated and modified during a dialog. One of the major roles of mental space is to provide the search space for possible referents of an expression.

The use of notions of ground and mental space makes the model applicable to the tasks of text comprehension and dialogue modeling, even though these tasks are out of scope of this thesis and the design was not concerned with them.

4.3.6 Encoding

The previous subsection described the representation of lexical items that is appropriate for both ACT-R theory and Cognitive Grammar. This section discusses how this representation is used in our model and the linguistic phenomena it accounts for. Processing is limited to active and passive transitive clauses, since these are the only sentence types that were used by Anderson (1974) in his experiment. Processing also reflects the assumptions that construal (i.e. trajector/landmark organization) is not part of the final representation.

4.3.6.1 Selection of categorizing structures

The process of declarative memory retrieval of ACT-R presented in Section 3.3.1 is the same with the process of selection of categorizing structures proposed by Langacker (1987, 1991, 1999a). Both processes assume likelihood of activation, contextual priming and the degree of similarity to be the factors affecting the selection of the structure in long-term memory that will be retrieved for the categorization of a novel structure.

Although the representation developed in this thesis exhibits these properties, they are not required for the simulation of the experiment of Anderson (1974), because in his experiment conceptual priming was not investigated. However, the process was considered, and it is used as a default for language processing, particularly for the retrieval of meanings of lexical items.

It was mentioned in Section 4.3.3 that virtual subjects requested a retrieval of the symbolic link for the string of letters they see by the production rule `read-word`. When the symbolic link is retrieved, the production rule `encode-word` is applied.

```
 (p encode-word
       =goal>
          isa             lwm
          state           "encoding"
       =retrieval>
          isa             s-link
          sem             =con
==>
       =goal>
          isa             lwm
          schema          =con
          state           "reading"
       +retrieval>
          isa             pw-link
          part            =con)
```

It will register that the current schema (slot *schema* of the goal chunk)[40] is the concept associated with the string of letter, and request the retrieval of a part-whole link that represents the profile-base relationship. Although the selection of the next production depends on the words that were read before (e.g. *the*), their common action is to encode the whole of the retrieved part-whole link into the slot *base* of the goal chunk.

The part-whole chunk that will be retrieved is the one with the highest activation. The chunk with the highest activation will be the one used most frequently (entrenchment). Since the chunk, which is specified as a value of the *part* slot of the part-whole link to be retrieved, is a slot value of the goal chunk, it will spread some activation to the part-whole link it participates in (ACT-R feature for contextual priming). Consequently, the primary domain of the lexical item represented by that schema will be retrieved. The retrieved *base,* after it is encoded into the goal chunk, will activate its own set of relationships and will affect the retrieval of part-whole links of other lexical items.

4.3.6.2 Composition

Representation and retrieval of concepts alone are not enough to account for language processing. The process of combination of single concepts to form complex phrases as well as the process of segmenting these complex phrases into constituent parts have to be specified. In most theories of grammar (e.g. Jackendoff (2002)) these processes are handled by syntactic processors, and syntactic parse trees form a separate level of representation. However, according to Langacker (1987, 1991,

[40] Entities encoded in the goal chunk are also referred to as entities in the focus of attention.

1999a) segmentation of expression into constituents is done with respect to patterns for creating and understanding composite structures – *constructional schemas,* which are schematizations of linguistic expressions, as discussed in Section 2.1.5. Langacker (1987) claims that the notions of *profile determinacy* and *elaboration site (e-site)*[41], which affect the combination of two structures, are both integrated in constructional schemas. Constructional schemas required for the processing of the sentences like *the painter visited the missionary* and *the missionary was visited by the painter* are embodied in production rules and they are presented in the following subsections.

4.3.6.3 Nominals

According to Cognitive Grammar, expressions like *the painter* and *the missionary* profile grounded instances of the types *painter* and *missionary*, respectively (i.e. they are *grounded things* or *nominals*). However, these expressions can be analyzed as consisting of two parts. At the phonology level, the component structures of the phrase *the painter* are [*the* …][42] and *painter*. They are symbolizations of the grounded thing and the type *painter*, respectively. In this case, the definite article *the* profiles any grounded thing, which is represented by a chunk like:

```
E1 isa entity
```

[41] When two component structures are combined to form a composite structure, the profile of one of the component structures is selected to be the profile of the composite structure, and this component structure is called *profile determinant*. The other component structure usually elaborates a subpart of the profile determinant structure, and the elaborated subpart is called *elaboration site*.

[42] Square brackets are used to represent schemas. Note that the schema contains a variable (three dots), since it is the commonality observable across all such expressions, and the only recurrent part is *the*.

For an entity to be grounded, there has to be a part-whole link that connects this entity to the mental space. Consequently, there is also a chunk like:

```
Gre002
   isa    pw-link
   part   E1
   whole  MS         ; mental space
```

Since an entity is a thing, there is also a categorization link represented by a chunk:

```
Gre003
   isa    c-link
   inst   E1         ; instance
   type   thing*     ; type
   ctx    MS         ; context
```

The chunk specifies that an entity *E1* is an instance of the type *thing*, and that it was categorized as a thing in the context of *mental space*.

Painter, on the other hand, which profiles a type of thing, is represented by a part-whole link that also specifies its whole, which is assumed to be **paint**, like in the chunk below:

```
Pw004
   isa    pw-link
   part   painter*
   whole  *paint*
   cfg    AG
```

When two component structures ([*the* …] and *painter*) are combined, one of them elaborates a subpart of the other. In this case, *painter* elaborates the type specification of [*the* …], which is a profile determinant. The resulting structure profiles the grounded entity of type *painter*, and the type specification for the entity *E1* becomes *painter*, rather than *thing*, and the categorization link that is given below is added.

```
C0004
   isa   c-link
   inst  E1
   type  painter*
   ctx   MS
```

Constructional schemas are represented by a pair of production rules (the first production rule receives component structures, and the second combines them), based on the fact that the frequent use of a schema proceduralizes it. Recall from Section 4.3.6.1 that virtual subjects encoded concepts into the *schema* slot of the goal chunk. If the goal chunk already contains some chunk in that slot, the concept (i.e. part) in the retrieved part-whole link is first combined with the existing schema.

For example, the production rule `categorize-nominal-old`, which represents the constructional schema for the expressions like *the painter*, applies when the schema slot has the value *g-thing-old** (i.e. previously grounded thing or thing already in mental space, which is the meaning of [*the* …]) and the part is of type *thing*.

```
(p categorize-nominal-old
        =goal>
                isa             lwm
                tschema         g-thing-old*
                context         =ctx
                state           "reading"
        =retrieval>
                isa             pw-link
                part            =con
                whole           =base
==>
        =goal>
                isa             lwm
                tbase           =base
                ttype           =con
                state           "find-referent"
        +retrieval>
                isa             c-link
                type            =con
                ctx             =ctx)
```

What this production practically does is the retrieval request for the categorization link of the entity in the mental space (i.e. it checks whether there is an entity in the mental space that was categorized as an instance of the type specified by a part-whole link). Upon retrieval of the categorization link, virtual subjects apply the production rule `check-nominal-yes-TR`:

```
(p check-nominal-yes-TR
        =goal>
                isa             lwm
                context         =ctx
                state           "find-referent"
        =retrieval>
                isa             c-link
                inst            =ent
                type            =con
==>
        =goal>
                isa             lwm
                tentity         =ent
                tschema         g-thing*
                ttype           =con
                state           "find-next")
```

Since the result of the combination of [*the* …] with *painter*, is the entity in the mental space categorized as *painter*, the virtual subject focuses on that entity (i.e. it becomes a slot value of the goal chunk).

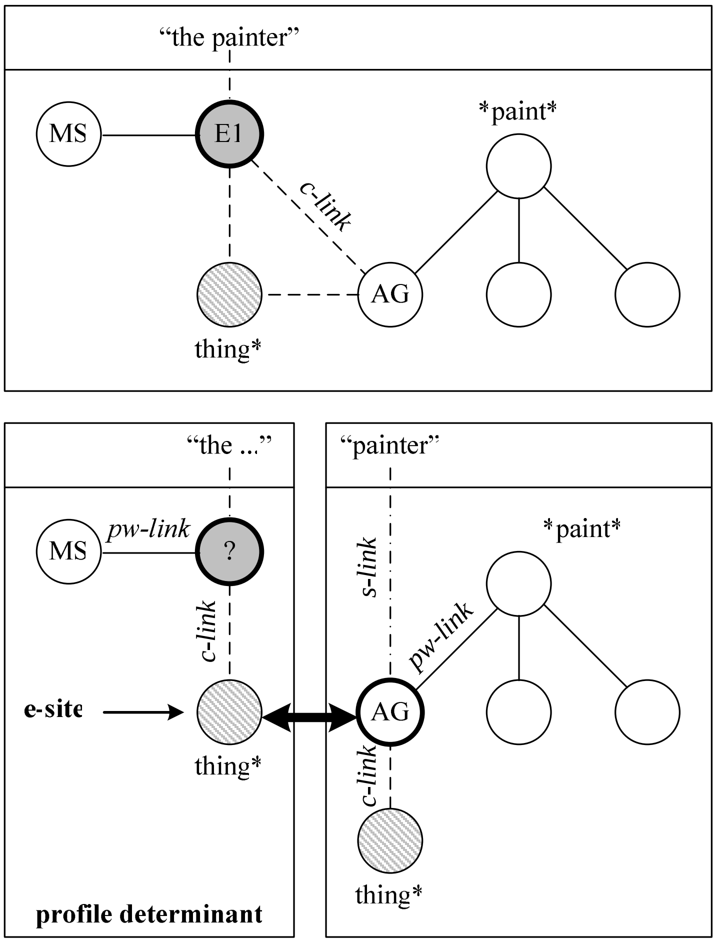


Figure 4.6: Nominal construction

The process of combination of [*the* …] with *painter* is schematically represented in Figure 4.6. In the figure, the upper picture represents the structure that results from the combination of two structures illustrated on the lower pictures.

The sentences used by Anderson (1974) in his experiment contain only definite articles. Consequently, there is no need to consider the constructional schema for expressions like *a painter*. However, the set of production rules for this constructional schema exists in the model, and it is discussed in Subsection 4.3.7 with respect to the mental space construction.

4.3.6.4 Clauses

Another constructional schema required for the simulation of the experiment of Anderson (1974), is the one that specifies the composition of nominals with processes. The previous subsection stated that [*the* …], rather than *the* is the component part of the nominal structure. The similar case is observed in the example sentence *the painter visited the missionary*, with respect to *visited*. Two component structures for the constructional schema that integrates a nominal with a process are *the painter* and [… *visited* …], rather than *the painter* and *visited*.

The meaning of *the painter* was discussed in the previous subsection, it profiled a grounded thing and *the* was the grounding predication. In the case of processes, grounding predications are tense and modals. Consequently, […*visited*…] profiles a grounded process of the type *visit*. Although, there should be constructional schema for the integration of [… *-ed*] and *visit*, the model assumes a word, rather than a morpheme to be a unit of analysis; consequently, the chunks that encode verbs in the model represent grounded processes.

The grounded process of the type *visit* is represented by a number of chunks:

```
P0004                   p0006                   p0005
   isa    pw-link          isa    c-link           isa   pw-link
   part   visit*           inst   E2               part  E2
   whole  *visit*          type   visit*           whole MS
   cfg    process          ctx    MS
```

First of all, [… *visit* …] profiles the part of a more complex structure **visit** (its conceptual base); therefore the chunk `p0004` is retrieved for the combination with the previous schema. Since [… *visited* …] profiles a grounded process (represented by the chunk `E2 isa entity`), which is grounded by the part-whole link `p0005` and categorized as an instance of the *visit* process by the categorization link `p0006`.

The production rule `categorize-and-ground-clause`, which represents a constructional schema for the nominal-process combination, applies when the retrieved part-whole link profiles a process:

```
(p categorize-and-ground-clause
        =goal>
                isa             lwm
                schema          g-thing*
                context         =ctx
                state           "reading"
        =retrieval>
                isa             pw-link
                part            =con
                whole           =base
                cfg             process*
==>
        =cl>
                isa             entity
        =pw-link>
                isa             pw-link
                part            =cl
                whole           =ctx
        =c-link>
                isa             c-link
                inst            =cl
                type            =con
                ctx             =ctx
        =goal>
                isa             lwm
                tschema         g-process*
                tentity         =cl
                ttype           =con
                state           "integrate-TR"
+retrieval>
                isa             pw-link
                whole           =base
                cfg             AG)
```

The production rule creates chunks `E2`, `p0005`, and `p0006`, which were already presented. Furthermore, since participants in a process are represented by part-whole links, it requests the retrieval of the link that specifies the agent (recall that elaboration sites are integrated in constructional schemas).

The production rule that applies next is `integrate-tr` (tr means trajector):

```
(p integrate-TR
        =goal>
                isa             lwm
                entity          =nom
                tentity         =cl
                context         =ctx
                state           "integrate-TR"
        =retrieval>
                isa             pw-link
                part            =esite
                base            =base
                cfg             =role
==>
        =c-link>
                isa             c-link
                inst            =nom
                type            =esite
                ctx             =ctx
        =pw-link>
                isa             pw-link
                part            =nom
                whole           =cl
                cfg             =role
        =goal>
                isa             lwm
                state           "find-next")
```

It categorizes the previous entity as an instance of the type provided by the retrieved part-whole link (i.e. categorizes as a participant in the process) by the addition of a categorization link, and specifies that the entity participates in the grounded process (E2) by the addition of the part-whole link. As the result of this pair of productions, the phrase *the painter visited* profiles a grounded process of the type *visit*, and specifies that its agent is the thing categorized as a *painter*.

There are similar production rule pairs that represent constructional schemas for the combinations process-thing, process-process, process-by, and by-thing that are enough to process active and passive sentences. The production pair process-process is used for the combination of *was* with *past participles* in passives. The meaning of *was* is the schematic process, which is elaborated by a part-participle. Since, *was* does not profile the whole process (e.g. for a two participant process, it leaves the agent unelaborated), the agent of the process is added into profile by the atemporal relation *by*.

The process of encoding of nominals and clauses is consistent with the requirements of Cognitive Grammar, since it is based on constructional schemas (represented by production rules) and the relationships of symbolization, categorization and designation (represented by s-links, c-links, and pw-links, respectively). It also accommodates the assumption that the trajector/landmark organization, which is used in production rules, is not part of the resulting structure. The representation that results from this encoding is discussed in Section 4.3.8.

4.3.7 Mental space construction

Although it is a fact that there are different mental spaces like politics and kinship relations (abstract domains), which constitute our world knowledge, they are not modeled, since the modeled experiment does not require the use of such knowledge. It is assumed that virtual subjects create new mental spaces for experiments and construct them in parallel with composition process. The construction of mental spaces is required from the standpoint of Cognitive Grammar, and it is integrated in the model developed in this thesis to meet this requirement, and is used as a search space for the referents of linguistic expressions, since created mental spaces are populated by entities and relations between them.

The previous sections already mentioned the meanings of definite article *the* and the past morpheme *–ed* (they profile grounded things and processes, respectively), and the production rules that process them. Besides those productions, the model contains a production pair that represents the constructional schema for phrases like *a painter*.

The first production rule of the pair will introduce an entity to mental space, that is, create and connect it to mental space with a part-whole link. The difference between definite and indefinite articles is in that the former will not add a new entity, but rather single out some existing entity in mental space; whereas indefinite article will simply add a new entity. The second production rule of the pair will categorize the entity introduced by the first production rule as an instance of the type specified by the following word, and note that the categorization was made in the context of current mental space. In the model, if the production pair for the phrase *the painter* fails to retrieve the associated entity, the virtual subjects apply the production rules similar to the production pair for new nominals. Consequently, the encoding of the sentences does not fail, in spite of the fact that initially there is no entity in the mental space, and all the sentences contain definite articles. Processes are introduced and categorized in the same way with nominals, that is, part-whole and categorization links are added, as it was exemplified in Subsection 4.3.6.4.

The construction of mental space makes the model applicable to the tasks of text comprehension and discourse modeling. Furthermore, it makes the model closer to the natural process of language comprehension.

4.3.8 Representation of sentential information

Figure 4.7 illustrates the representation resulting from processing of the sentences used in the experiment by Anderson (1974).

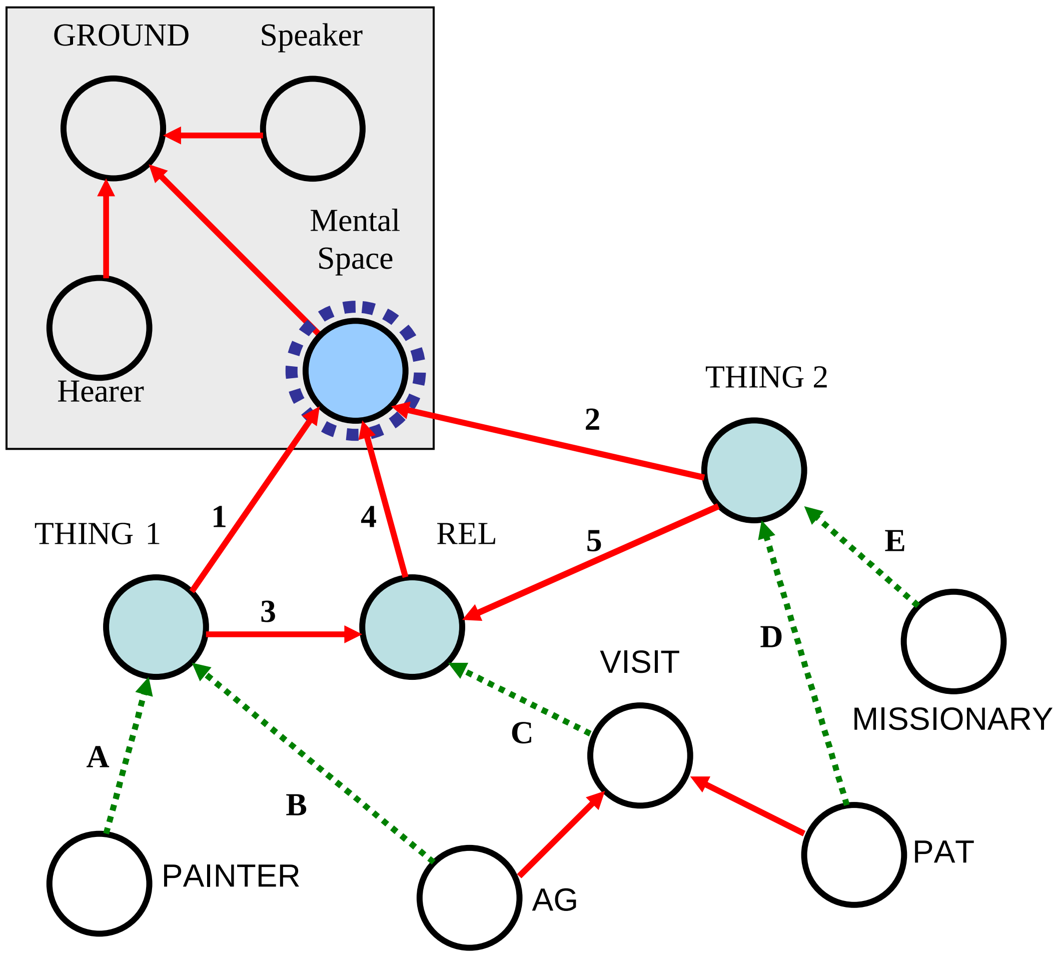


Figure 4.7: Representation of sentential information

In the figure the upper shaded area represents the ground and the mental space; parts of the ground – speaker and hearer. With respect to the sentence *the painter visited the missionary;* the mental space starts with a number of links (since *the painter* and *the missionary* profile previously grounded nominals). The entities (shaded circles) that exist in the mental space are labeled THING1 and THING2. White circles represent concepts (schemas), which are linked to THING1 and

THING2 by categorization links A and E. The existing part-whole links are straight lines labeled 1 and 2.

When the virtual subject processes *visited*, the new entity (REL) is added, and it is connected to the mental space by the part-whole link labeled 4, and categorized as an instance of the process type *visit* by the addition of the categorization link C. The categorization links B and D and the part-whole links 3 and 5 are added by the production pairs that represent the constructional schemas for the thing-process and process-thing combinations, respectively.

There are two categorization links for each thing, one represents the categorization of an entity before the process, and the other represents the categorization of the entity as the part of the process schema (i.e. elaboration site). Two part-whole links between the process (REL) and things (THING1) and (THING2), which are labeled 3 and 5, are used for exact specification of which thing participated in which process. For example, if there are several processes of the same type like in the sentence *John went to school, but Bill went to park*, the information that *John* and *Bill* are agents of the process *to go* is not enough, it has to be specified exactly who participated in each process.

As it is explained in the following (Section 4.3.9), the representation is suitable for the sentence verification task of Anderson (1974).

4.3.9 Retrieval of sentential information

In the experiment of Anderson (1974), the subjects had to judge whether the probe sentences were implied by the study sentences. Consequently, the production rules for the retrieval of sentential information (i.e. the retrieval of chunks that encode this information) have to be defined. The model developed in this thesis differs from the models of Anderson et al. (2001) and Lebiere (2002) with respect to the strategy used for sentence verification. Their assumption was that the subjects first parse the sentence, i.e. they construct the new representation, and then check whether it matches any proposition from the study set. However, in the model of this thesis it is assumed that the subjects try to find a matching structure without waiting for the sentence to end. Thus, at the end of the sentence they will have already selected a candidate match or failed to do so. The assumption is plausible since the primary goal is not to construct the representation, but to check whether there is a matching one. Moreover, this assumption makes the verification process more similar to the process of encoding.

For example, the sentence *the missionary was visited by the painter*, which yields the same representation with the sentence *the painter visited the missionary* that is shown on Figure 4.7, would be tested like follows:

1. Read “*the missionary*” and find its referent (i.e. retrieve the categorization link labeled A).
2. Read “*was visited*” and find its referent (i.e. retrieve the categorization link labeled C).
3. Check whether the entity THING1 participated in the process REL with the same role (i.e. retrieve the part-whole link labeled 3).
4. Read “*the painter*” and find its referent (i.e. retrieve the categorization link labeled E).
5. Check whether the entity THING2 participated in the process REL, and whether it participated with the same role (i.e. retrieve the part-whole link labeled 5).
6. Respond.

The production rules used for the reading and the combination of words are the same with those used for encoding words. The different production rules apply in steps 2, 3, and 5. Instead of the creation of categorization and part-whole links, these production rules retrieve them. The different production rules are selected, because the goal chunk contains the slot *purpose*, and in the test phase it is set to “test”. The condition part of these production rules contains the specification of this slot (i.e. they can apply only if the purpose it to test).

The virtual subjects respond with respect to the *error* slot in the goal chunk, which is initially set to “nil”. If in any step during verification the virtual subject fails to retrieve the required chunk, the error slot becomes “error”. In order to change the slot value, another production has to be applied (there are “error” production rules for each retrieval attempt in the model). Thus, this extra production rule accounts for the fact that in the original experiment it took about 0.15 - 0.30 seconds longer to respond to the false probe sentences.

For example, the production rule `check-TR-no` applies when the virtual subject fails to retrieve a chunk that encodes that the entity referred to by the first noun phrase participated in the process referred to by the verb.

```
(p check-TR-no
   =goal>
      isa            lwm
      purpose        "test"
      state          "check-TR"
   =retrieval>
      isa            error
==>
   =goal>
      isa            lwm
      error          "error"
      state          "find-next")
```

As it can be seen, the action side of the production rule changes the *error* slot to indicate that the virtual subject failed to retrieve the required information (i.e. detected an error).

The results of the simulation of the experiment are evaluated and discussed in the following section.

4.4 Evaluation of the Model

This section discusses the results of the ACT-R simulation of the experiment of Anderson (1974). Recall that the delayed condition of the second experiment was modeled, whereas the reaction times of the delayed condition of the first experiment were used for comparison. The reason for this is the fact that the delayed condition makes uses of the propositional strategy more often (almost always); thus, it reflects the timing of this strategy more clearly.

As it was mentioned in Chapter 3, ACT-R comes with two functions for data analysis. The reaction times of the virtual subjects that were collected by the function *report-data* (see Section 4.3.2) are analyzed in terms of the correlation and the mean deviation with the mean reaction times of the real subjects that participated in the experiment of Anderson (1974) (see Section 4.1). The results obtained by Anderson (1974) for the modeled condition were given in Section 4.1, and they are repeated in the table below.

Table 4.3: The mean reaction times in the delayed condition in Anderson (1974)

Input Probe	Active Active	Active Passive	Passive Active	Passive Passive
True	2.25	2.80	2.30	2.75
False	2.55	2.95	2.55	2.95

Recall that each cell in the table represents a different condition. For example, the cell in the first row of the first column represents the mean verification time for the true sentences presented and tested in active voice. The results collected on the ACT-R simulation of the experiment with the model developed in this thesis are given on Table 4.4.

Table 4.4: Reaction times obtained from the ACT-R simulation

Input Probe	Active Active	Active Passive	Passive Active	Passive Passive
True	2.29	2.95	2.29	2.95
False	2.59	3.06	2.59	3.06

The correlation of the results obtained from ACT-R simulation with the data obtained by Anderson (1974) is 0.988 and the mean deviation is 0.107. The correlation value shows that the processes proposed in the model of this thesis can account for the results of the original experiment with 98.8% accuracy. The mean deviation value, on the other hand, shows that the overall speed of processing was 0.107 seconds slower than the original experiment. The reason for this overall slowness could be the occasional use of the verbatim strategy in the delayed condition of the first experiment.

The comparison of the model developed in this thesis with the models of Anderson et al. (2001) and Lebiere (2002) in terms of the correlation and the mean deviation is shown on Table 4.5. As it can be seen from the table, our model is compatible with the models of Anderson et al. (2001) and Lebiere (2002).

Table 4.5: Comparison with other models in terms of the correlation and the mean deviation

	Anderson, et al. (2001)	Lebiere (2002)	The model
Correlation	0.996	0.978	0.988
Mean deviation	NA	0.072	0.107

Since Anderson et al. (2001) does not report mean deviation; correlation is the only value that can be compared. With respect to the models of Anderson et al. (2001) and Lebiere (2002), the structure of our model has a number of differences and similarities, which are shown on the table below:

Table 4.6: Comparison with other models in terms of representation and parameters

	Anderson, et al. (2001)	Lebiere (2002)	The model
Sentential repr.	Fragmented	Atomic	Fragmented
Lexical repr.	Atomic	Atomic	Both[43]
Parameters			
Decay	YES	NO	NO
Production specific	YES	YES	NO
Latency factor	0.3	0.15	0.15

The differences of our model from both models are the representation for lexical items and the verification strategy, which were presented in previous sections. With respect to the parameters used, the strong point is of our model is the absence of any production specific parameters. For example, production specific parameters like `:effort` affect the time it takes a production rule to complete; thus, it can affect reaction times.

[43] The representation of lexical items is atomic with respect to symbolic links, that represent the association between sound and meaning; it is fragmented in the sense that lexical item participates in a number of relations, that are part of its meaning.

The latency factor and decay also affect the time it takes to retrieve a chunk. It was set to 0.15 after the model of Lebiere (2002), and, although it is the most variable parameter across ACT-R models, it appears to be a reasonable value. The decay parameter reflects the speed of forgetting (i.e. chunks lose their base level activations with time). Decay was not used in the model, because the subjects in the original experiment of Anderson (1974) were almost always correct; consequently, the chunks should always be retrieved.

The model is not compared to the implementation of Heinze (1994) because it did not attempt to simulate human performance, but rather used ideas from Cognitive Grammar for computational purposes.

Although the implementation of Holmqvist (1993, 1998) is also not concerned with human performance, below we will compare his model with, since it was a major attempt to model Cognitive Grammar computationally.

The structure of the lexicon in our model resembles the one used by Holmqvist (1993, 1998). The lexicons in both implementations are formed by the relations that can exist between lexical items. However, a lexical item in Holmqvist (1993, 1998) includes only the matrix of domains ordered by centrality values (see Section 2.1.2), list of parts ordered by their saliences (i.e. their importance to the meaning), and list of wholes ordered by their saliences. In our model, on the other hand, there are also categorization relationships. Another difference between the structure of lexical items is that in our model a lexical item is represented by a number of chunks, whereas in Holmqvist (1993, 1998), a lexical item is represented in one structure. This fragmented representation, when enhanced with ACT-R’s activation calculus (see Section 3.3.1), allows to account for the ambiguity, vagueness, and change in the meaning of lexical items arising from frequency of use.

The proposed processes for encoding are entirely different. Since Holmqvist (1993, 1998) uses visual images as a meaning of lexical items, he does image superimposition (see Section 2.3); whereas in our model, meanings are represented by abstract symbols like *painter** (because ACT-R is underdeveloped to allow image manipulation, and Langacker (1987, 1991, 1999a) does not claim that image-schemas have such imagistic character). Furthermore, the constituency in Holmqvist (1993, 1998) is based on Behaghel's principle, which "claims a correlation between closeness of morphemes and closeness in valence relations" (Holmqvist, 1998, p. 167). Our model makes use of constructional schemas proposed by Langacker (1987, 1991, 1999a) for the same purpose, which are embodied in ACT-R production rules.

To summarize, in this thesis, the experiment of Anderson (1974) was implemented on ACT-R cognitive architecture considering the primitives of Cognitive Grammar. The representation for lexical items that meets the requirements of both theories was developed. It consists of symbolization, categorization, part-whole and domain specification links, which represent basic human cognitive abilities of association, categorization and designation, essential to the human linguistic ability according to Langacker (1987, 1991, 1999a).

The process of encoding of sentential information was also developed. The process makes use of constructional schemas embodied in production rules, which is consistent with Cognitive Grammar because it also considers proceduralization. The process of retrieval of sentential information for the delayed condition of the sentence verification task was proposed. The process is in line with the assumptions of Cognitive Grammar, and it is close to the natural process of sentence comprehension.

The results obtained from the ACT-R simulation of the experiment are comparable to the results of the original experiment of Anderson (1974). Thus, it is possible to conclude that the model is successful, and the processes proposed in the model could be used to account for the results of the original experiment.

CHAPTER 5

Conclusion

This chapter summarizes contributions of the thesis (Section 5.1), points out limitations of the model (Section 5.2), and presents some directions for the future development of the model (Section 5.3).

5.1 Contributions

The main objective was to design a language processing model based on Langacker's Cognitive Grammar (1987, 1991, 1999a) and implement it on ACT-R, a cognitive architecture developed by J. R. Anderson and colleagues (Anderson & Lebiere, 1998a; Anderson et al., 2004). The purpose of designing of the model was to show that ACT-R provides the minimal context of general human cognitive abilities necessary for Cognitive Grammar to be modeled. Although this is not the first attempt to computationally model Langacker's theory, it is the first model that was implemented on a cognitive architecture.

5.1.1 Representation

The representation was intended to represent image schemas of Langacker (1987, 1991, 1999a) and Lakoff (1987). It was inspired by the representation used in Holmqvist (1993), who also attempted to model Langacker's theory. However, it was substantially modified to suit the representational commitments of ACT-R.

The lexical items are represented in a form of schemas made up of four types of links: symbolic links that represent associations between sound and meaning; categorization links that represent taxonomic relations between concepts; designation or domain specification links that represent concepts with respect to human sensory abilities such as color, shape and so forth; and part-whole links that represent part-whole relations between concepts or their specifications in abstract knowledge spaces such as kinship, politics, and so forth. With the use of these kinds of links, any kind of semantic unit is represented in a uniform manner: verbs (processes), nouns (things), morphemes, grammar rules all have the form of schemas and constitute lexical items. Since schemas can function as parts or wholes of other schemas the representation resulted in a multilevel schematic (semantic) network which is easily updated: new links could be added as well as modified; consequently, learning can take place.

The designed representation can be used to explain various linguistic phenomena like lexical ambiguity and its resolution, vagueness, meaning change, contextual priming, and polysemy, even though they are not accounted in the model, since it is limited to the sentence verification task in the experiment of Anderson (1974). The same types of links are used for the representation of sentential information, discourse, and extra-linguistic knowledge such as personal experience (episodic memory). Because of the time limitations imposed by 0.050 seconds serial bottleneck of ACT-R[44] previous language processing models on ACT-R processed the language "in a rather sloppy and shallow way – not processing each word independently and fully" (Anderson et al., 2002, p. 33); in other words, there was not

[44] The minimum time it takes a production to execute.

enough time to elaborate meaning further than a single chunk. If we take into account the fact that the working memory in ACT-R is the set of declarative memory chunks that are above the threshold of activation, the representation presented in this thesis allows more complete activation of meaning, because a profile of a lexical item activates the relationships it participates in.

5.1.2 Processing

The designed language processing is reflection of the human cognitive abilities considered by Langacker as essential to linguistic abilities. Since in ACT-R production rules are units of procedural knowledge and according to Langacker human linguistic ability is mainly procedural, the processing follows the requirements of Cognitive Grammar. Production rules specified for the model could be seen as proceduralized schemas. Most of them could be seen as constructional schemas of Langacker that specify the way two schemas are combined with respect to notions of profile determinancy, selection of elaboration sites and detection of whether elaborating schema are an instance of elaboration site schema. It was shown that production rules combined with the representation are capable of producing human like performance in terms of speed and accuracy by testing the model on a psychological experiment data (Anderson, 1974). The model was compared with other language processing models in ACT-R and was found to lead to comparable results.

5.1.3 Applicability

Despites the simplicity, the representation and processing are applicable to modeling of a wide variety of tasks. It was shown that the model is capable of comprehension of isolated active and passive sentences. Moreover, in Chapter 4 it was mentioned that the processing and representation are applicable to dialogue modeling and text comprehension, since the notions of grounding and mental space, which are also represented similar to concepts, construct the context by default.

5.2 Limitations

The language processing model presented in this thesis neglects a number of key points of Cognitive Grammar and some psychological phenomena related to language processing. First of all, the number of implemented schemas in the model is relatively small and the only types of sentences the model can deal with is active transitive clauses and passives. The lexicon of the model is limited to the words that were used in the experiment by Anderson (1974). Limitations other than the scope of the model are presented below.

5.2.1 Phonology and morphology

In Cognitive Grammar phonological pole plays a role as important as semantic pole. However, this work was primarily concerned with semantics. The phonology, according to Langacker, exhibits the same properties as semantics, that is, there are categorization and schematicity relations as well. The processing should go in parallel in both semantic and phonological poles. The model treats words as units of analysis neglecting the fact that a morpheme is the unit of analysis in Cognitive Grammar. Consequently, there is no account for morphology. However, the representation and processing will become capable of this with the addition of extra productions (all of which will be constructional schemas) and schemas for the morphemes.

5.2.2 Reading

As it was mentioned before, the reading part of the model is not precise, it is simplified. Although there is no unitary vision on the reading process, the author of this thesis takes stance near to Crowder and Wagner (1992), who advocate that there is a subvocalization process necessarily going on while reading a sentence, since the proper understanding of the sentence requires the reconstruction of its prosodic structure. The model does not account for the verbatim (exact wording) memory for the sentence which is a well established phenomenon. It was assumed that the verbatim memory is the result of trace left by subvocalization; however, the process was not implemented and tested.

5.3 Future Development

The model needs to be developed further to be able to deal with broader range of linguistic data. Besides the expansion of lexicon and inclusion of all type of sentences into the model, the next step will be to eliminate all the limitations of the model as far as ACT-R architecture allows it. Unfortunately ACT-R is not developed enough to allow exact implementation of basic (sensory) domains; thus, the development of an approximation is one of the possibilities. Since the representation and processing are modifiable and expandable; and since ACT-R includes learning mechanisms the natural intention will be to construct a language acquisition model based on Cognitive Grammar. Langacker presents innate cognitive abilities required for language processing and literature on language acquisition is rich with respect to the stimuli children receive and their linguistic abilities at certain stages. A successful language acquisition model could contribute to the debates on innateness of language faculty and the psychological plausibility of Cognitive Grammar.

REFERENCES

Anderson, J. R. (1974). Verbatim and propositional representation of sentences in immediate and long-term memory. *Journal of Verbal Learning and Verbal Behavior, 13*, 149-162.

Anderson, J. R. (1976). *Language, memory, and thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R. (1990). *Cognitive psychology and its implications* (3rd ed.). New York, NY: W. H. Freeman and Company.

Anderson, J. R. (1991). Is human cognition adaptive? In T. A. Polk & C. M. Seifert (Eds.), (2002), *Cognitive Modeling*. Cambridge, MA: MIT Press. (Reprinted from *Behavioral and Brain Sciences, 14 (3)*)

Anderson, J. R. & Bothell, D. (2004). *The ACT-R 5.0 Tutorial*. Carnegie Mellon University, http://act-r.psy.cmu.edu/tutorials (24.06.2004).

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C. & Qin, Y. (2004). *An integrated theory of mind*. Carnegie Mellon University, http://act-r.psy.cmu.edu/.

Anderson, J. R., Bothell, D., Byrne, M. D., & Lebiere, C. (2002). *An integrated theory of mind*. Carnegie Mellon University, http://act-r.psy.cmu.edu/.

Anderson, J. R., Bothell, D., Lebiere, C. & Matessa, M. (1998). An integrated theory of list memory. *Journal of Memory and Language, 38*, 341–380.

Anderson, J. R. & Bower, G. H. (1973). *Human associative memory*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Budiu, R. & Reder, L. (2001). A theory of sentence memory as part of a general theory of memory. *Journal of Memory and Language, 45*, 337–367.

Anderson, J. R. & Lebiere, C. (Eds.) (1998a). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R. & Lebiere, C. (1998b). Introduction. In J. R. Anderson & C. Lebiere (Eds.) *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R. & Lebiere, C. (1998c). Knowledge representation. In J. R. Anderson & C. Lebiere (Eds.) *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R. & Lebiere, C. (1998d). Learning. In J. R. Anderson & C. Lebiere (Eds.) The atomic components of thought. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Anderson, J. R., Lebiere, C. & Lovett, M. (1998). Performance. In J. R. Anderson & C. Lebiere (Eds.) The atomic components of thought. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Bach, E. (1983). On the relationship between word-grammar and phrase grammar. Natural Language and Linguistic Theory, 1, 65-89.

Block, N. & Rey, G. (1998). Mind, computational theories of. In Routledge Encyclopedia of Philosophy, Version 1.0. London and New York: Routledge.

Bloomfield, L. (1933). Language. New York, NY: Holt.

Budiu, R. (2001). The role of background knowledge in sentence processing, Doctoral Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Budiu, R. & Anderson, J. R. (2000). Integration of background knowledge in sentence processing: A unified theory of metaphor understanding, semantic illusions, and text memory. Proceedings of the Third International Conference on Cognitive Modeling (pp. 50-57). Groningen, the Netherlands: Universal Press.

Budiu, R. & Anderson, J. R. (2001). Word learning in context: Metaphors and neologisms. (Tech. Rep. CMU-CS-01-147.) School of Computer Science, Carnegie Mellon University.

Budiu, R. & Anderson, J. R. (2002). Comprehending anaphoric metaphors. Memory & Cognition, 30, 158-165.

Budiu, R. & Anderson, J. R. (2003). Verification of sentences containing anaphoric metaphors. Proceedings of the Fifth International Conference on Cognitive Modeling. Bamberg, Germany.

Budiu, R. & Anderson, J. R. (2004). Interpretation-based processing: A unified theory of semantic sentence processing. Cognitive Science, 28, 1–44.

Byrne, M.D. (2004). ACT-R/PM Documentation. Rice University, http://www.chil.rice.edu/projects/RPM/index.html (10.04.2004).

Byrne, M. D. & Anderson, J. R. (1998). Perception and action. In J. R. Anderson & C. Lebiere (Eds.) The atomic components of thought. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Byrne, M. D. & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing. Psychological Review, 108, 847-869.

Chomsky, N. (1957). Syntactic structures. The Hague: Mouton.

Chomsky, N. (1965). Aspects of the theory of syntax. Cambridge, MA: MIT Press.

Cognitive Linguistics. (2004). Journal webpage. http://linguistics.uiuc.edu/cogling/ (20.08.2004)

Crowder, R. G. & Wagner, R. K. (1992). The psychology of reading: An introduction. New York, NY: Oxford University Press.

Darwin, C. J., Turvey, M. T., & Crowder, R. G. (1972). The auditory analogue of the Sperling partial report procedure: Evidence for brief auditory storage. Cognitive Psychology, 3, 255-267.

Davis, M. H. (2000). Lexical segmentation in spoken word recognition. Doctoral Dissertation, Birkbeck College, University of London.

Emond, B. (1997). Modeling natural language comprehension and anaphora resolution with ACT-R. Proceedings of the Fourth Annual ACT-R Workshop (pp. 1-8). Pittsburgh, PA: Department of Psychology, Carnegie Mellon University.

Emond, B. (1999). Estimating processing time of online semantic interpretation components. Proceedings of the Fifth Annual ACT-R Workshop (pp. 113-118). Fairfax, VA: Department of Psychology, George Mason University.

Fauconnier, G. (1985). Mental spaces: Aspects of meaning construction in natural language. Cambridge, MA: MIT Press.

Fauconnier, G. (1997). Mappings in thought and language. Cambridge, MA: Cambridge University Press.

Fauconnier, G. (1999). Methods and generalizations. In T. Janssen & G. Redeker (Eds.), Cognitive linguistics: Foundations, scope, and methodology. Berlin: Mouton de Gruyter.

Fauconnier, G. & Turner, M. (1998). Conceptual integration networks. Cognitive Science, 22, 133-87.

Fillmore, C. J. (1968). The case for case. In E. Bach & R. T. Harms (Eds.), Universals in linguistic theory. New York, NY: Holt, Rinehart & Winston.

Fodor, J. A. (1983). The modularity of the mind. Cambridge, MA: MIT Press.

Fodor, J. A. (2000). The mind doesn't work that way: The scopes and limits of computational psychology. Cambridge, MA: MIT Press.

Garrod, S. & Trabasso, T. (1973). A dual-memory information processing interpretation of sentence comprehension. Journal of Verbal Learning and Verbal Behavior, 13, 155-167.

Holmqvist, K. (1993). Implementing cognitive semantics. Doctoral Dissertation. Department of Cognitive Science, Lund University, Sweden.

Holmqvist, K. (1998). Conceptual engineering. In J. Allwood & P. Gardenfors (Eds.) Cognitive semantics: Meaning and cognition. Amsterdam, the Netherlands: John Benjamins Publishing Co.

Heinze, D. T. (1994). Computational cognitive linguistics, Doctoral Dissertation, Department of Industrial Engineering, Pennsylvania State University.

Jackendoff, R. (1977). X" syntax: A study of phrase structure. Cambridge, MA: MIT Press.

Jackendoff, R. (1990). Semantic Structures. Cambridge, MA: MIT Press.

Jackendoff, R. (2002). Foundations of language: brain, meaning, grammar, evolution. New York, NY: Oxford University Press.

Janssen, T. & Redeker, G. (Eds.) (1999). Cognitive linguistics: Foundations, scope, and methodology. Berlin: Mouton de Gruyter.

Johnson, M. (1987). The body in the mind: The bodily basis of meaning, imagination, and reason. Chicago: University of Chicago Press.

Johnson-Laird, P. N. (1983). Mental models. Cambridge, MA: Cambridge University Press.

Just, M. A., Carpenter, P. A. & Varma, S. (1999). Computational modeling of high-level cognition and brain function. Human Brain Mapping, 8,128–136.

Karmiloff-Smith, A. (1992). Beyond modularity: A developmental perspective on cognitive science. Cambridge, MA: MIT Press.

Kieras, D. E. & Meyer, D. E. (1998). The EPIC architecture: Principles of operation. University of Michigan, ftp://ftp.eecs.umich.edu/people/kieras/EPICarch.ps (12.06.2004).

Lakoff, G. (1987). Women, fire, and dangerous things. Chicago, IL: University of Chicago Press.

Lakoff, G. & Johnson, M. (1980). Metaphors we live by. Chicago: University of Chicago Press.

Landauer, T. K. & Dumais, S. T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction and representation of knowledge. Psychological Review, 105, 221–240.

Langacker, R. W. (1987). Foundations of Cognitive Grammar. Vol. 1. Theoretical prerequisites. Stanford, CA: Stanford University Press.

Langacker, R. W. (1991). Foundations of Cognitive Grammar. Vol. 2. Descriptive application. Stanford, CA: Stanford University Press.

Langacker, R. W. (1999a). Grammar and conceptualization. Berlin: Mouton de Gruyter.

Langacker, R. W. (1999b). Assessing the cognitive linguistics enterprise. In T. Janssen & G. Redeker (Eds.), Cognitive linguistics: Foundations, scope, and methodology. Berlin: Mouton de Gruyter.

Lebiere, C. (2002). Introduction to ACT-R 5.0: Tutorial. Presented at 24th Annual Conference of the Cognitive Science Society.

Lewis, R. L. (1999a). Cognitive modeling, symbolic. In R. Wilson and F. Keil (Eds.), The MIT Encyclopedia of the Cognitive Sciences. Cambridge, MA: MIT Press.

Lewis, R. L. (1999b). Attachment without competition: A race-based model of ambiguity resolution in a limited working memory. Presented at the CUNY Sentence Processing Conference, New York.

Lewis, R. L. (2001). Cognitive theory, SOAR. In International Encyclopedia of the Social and Behavioral Sciences. Amsterdam: Pergamon.

Matessa, M. & Anderson, J. R. (1997). Focused Learning in a Linguistic Environment. Proceedings of the 19th Annual Conference of the Cognitive Science Society (p. 990). Mahwah, NJ: Erlbaum.

Matessa, M. & Anderson, J. R. (2000). An ACT-R model of adaptive communication. Proceedings of the Third International Conference on Cognitive Modeling (pp. 210-217), University of Groningen, Netherlands.

Mazoyer, B. M., Dehaene, S., Tzourio, N., Frak, V., Murayama, N., Cohen, L., Levrier, O., Salamon, G., Syrota, A., & Mehler, J. (1993). The cortical representation of speech. Journal of Cognitive Neuroscience, 5, 467-479.

McLeod, P., Plunkett, K. & Rolls, E. T. (1998). Introduction to connectionist modelling of cognitive processes. New York, NY: Oxford University Press.

Medin, D. L. (1989). Concepts and conceptual structure. American Psychologist, 44, 1469-1481.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63, 81-97.

Misker, J. M. V. & Anderson, J. R. (2003). Combining optimality theory and a cognitive architecture. Proceedings of the Fifth International Conference on Cognitive Modeling (pp. 165-170), Bamberg, Germany.

Newell, A. (1990). Unified theories of cognition. Cambridge, MA: Cambridge University Press.

Olson, D. R. & Filby, N. (1972). On the comprehension of active and passive sentences. Cognitive Psychology, 3, 361-381.

Prince, A. & Smolensky, P. (1997). Optimality: From neural networks to universal grammar. In T. A. Polk & C. M. Seifert (Eds.), (2002), Cognitive Modeling. Cambridge, MA: MIT Press. (Reprinted from Science, 275 (5306))

Pylyshyn, Z. W. (1998). Cognitive Architecture. In Routledge Encyclopedia of Philosophy, Version 1.0. London and New York: Routledge.

Redeker, G. & Janssen, T. (1999). Introduction. In T. Janssen & G. Redeker (Eds.), Cognitive linguistics: Foundations, scope, and methodology. Berlin: Mouton de Gruyter.

Rosch, E. (1975). Universals and culture specifics in human categorization. In R. W. Brislin, S. Bochner & W. J. Lonner (Eds.), Cross-cultural Perspectives on Learning. New-York: Wiley.

Salvucci, D. D. & Anderson, J. R. (2001). Integrating analogical mapping and general problem solving: The path-mapping theory. Cognitive Science, 25, 67-110.

Shiffrin, R. M. & Schneider, W. (1977). Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. Psychological Review, 84, 127-190.

Sperling, G. A. (1960). The information available in brief visual perception. Psychological Monographs, 74, Whole No. 498.

Sternberg, R. J. (1999). Cognitive Psychology. (2nd ed.) Fort Worth, TX: Harcourt Brace College Publishers.

Sticht, T. G. (1978). The acquisition of literacy by children and adults. In F. B. Murray & J. J. Pikulski (Eds.), The acquisition of reading. Baltimore: University Park Press.

Sticht, T. G. (1979). Application of the aud-read model to reading evaluation and instruction. In L. B. Resnick & P. L. Weaver (Eds.), Theory and practice of early reading, 1. Hillsdale, NJ: Lawrence Erlbaum.

Sweetser, E. E. (1990). From etymology to pragmatics: Metaphorical and cultural aspects of semantic structure. UK: Cambridge University Press.

Taatgen, N. A. (1999). Learning without limits: From problem solving towards a unified theory of learning. Doctoral Dissertation, Department of Artificial Intelligence, University of Groningen, Groningen, the Netherlands.

Taatgen, N. A. (2001). Extending the Past-tense debate: A model of the German plural. In J. D. Moore and K. Stenning, Proceedings of the twenty-third annual conference of the cognitive science society (pp. 1018-1023). Mahwah, NJ: Erlbaum.

Taatgen, N.A. & Anderson, J.R. (2002). Why do children learn to say "broke"? A model of learning the past tense without feedback. Cognition, 86 (2), 123-155.

Taatgen, N.A. & Dijkstra, M. (2003). Constraints on Generalization: Why are past-tense irregularization errors so rare? Proceedings of the 25th annual conference of the cognitive science society (pp. 1146-1151). Mahwah, NJ: Erlbaum.

Talmy, L. (1988). Force dynamics in language and cognition. Cognitive Science, 12, 49-100.

Talmy, L. (2000). Toward a cognitive semantics (Vols. 1, 2).Cambridge, MA: MIT Press.

Taylor, J. R. (2002). Cognitive Grammar. New York, NY: Oxford University Press.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), Organization of memory. New York, NY: Academic Press.

Tulving, E. (1986). What kind of a hypothesis is the distinction between episodic and semantic memory? Journal of Experimental Psychology: Learning, Memory, & Cognition, 12 (2), 307-311.

Ungerer, F. & Schmid, H.-J. (1996). An introduction to cognitive linguistics. New York, NY: Addison Wesley Longman Inc.

Zondervan, K. & Taatgen, N.A. (2003). The determiners model: a cognitive model of macro development and U-shaped learning in a micro domain. In F. Detje, D Dörner and H. Schaub (Eds.), Proceedings of the Fifth International Conference on Cognitive Modeling (pp. 225-230). Bamberg, Germany: Universitätsverlag Bamberg.

APPENDIX

Model Listing

```
; Stepanov Evgueni A.
; 1132570
; Anderson (1974) based on Cognitive Grammar primitives
; Requires ACT-R 5.0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; VARIABLE & CONSTANT DEFINITIONS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; variables to record responses (the keys pressed)
; and reaction times (the times of the key presses)

(defvar *response* nil)
(defvar *response-time* nil)

; reaction times from the original experiment of Anderson (1974)
;
; +----------+----------+----------+----------+----------+
; |    input |  Active  |  Active  |  Passive |  Passive |
; |    probe |  Active  |  Passive |  Active  |  Passive |
; +----------+----------+----------+----------+----------+
; | True     |   2.25   |   2.80   |   2.30   |   2.75   |
; | False    |   2.55   |   2.95   |   2.55   |   2.95   |
; +----------+----------+----------+----------+----------+

(defconstant *a74-data* '(2.25 2.80 2.30 2.75 2.55 2.95 2.55 2.95))


; an example study set from Anderson (1974)

(defvar *a74-study*
   '("The painter visited the missionary"
     "The missionary refused the painter"
     "The painter was chased by the missionary"
     "The painter was protected by the sailor"
     "The missionary shot the sailor"
     "The cannibal questioned the painter"
     "The missionary was accused by the painter"
     "The cannibal was feared by the missionary"))
```

```
; the test set for the example study set
; ([condition] [correct response] [sentence])

(defvar *a74-test*
   '((1 "k" "The painter visited the missionary")
     (2 "k" "The painter was refused by the missionary")
     (3 "k" "The missionary chased the painter")
     (4 "k" "The painter was protected by the sailor")
     (5 "d" "The sailor shot the missionary")
     (6 "d" "The cannibal was questioned by the painter")
     (7 "d" "The missionary accused the painter")
     (8 "d" "The missionary was feared by the cannibal")))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; FUNCTION DEFINITIONS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; the function opens a window and displays a sentence in it,
; and runs the model for the specified amount of time

(defun study-sentence (text time)
  (let ((window (open-exp-window "Anderson 1974"
                                 :visible t
                                 :width 500)))
    (pm-install-device window)
    (add-text-to-exp-window :text text
                            :x 25 :y 150
                            :width 250)
    (pm-proc-display :clear t)
    (pm-run time :full-time t)))
;--------------------------------------------------------------------

; the function opens a window and displays a sentence in it,
; runs the model and records response and response time of a
; virtual subject, and puts them in a list, where the first
; element is the condition (e.g. Active-Active is 1), the second
; element is the response time, and the third element is either
; T or F to represent whether the response was correct or not.

(defun test-sentence (test)
  (let ((window (open-exp-window "Anderson 1974"
                                 :visible t
                                 :width 500)))
    (pm-install-device window)
    (add-text-to-exp-window :text (third test)
                            :x 25 :y 150
                            :width 250)
    (pm-proc-display :clear t)
    (setf *response-time* nil)
    (setf *response* nil)
    (let ((start-time (pm-get-time)))
      (pm-run 30)
      (setf *response-time* (if *response-time*
                                (- *response-time* start-time)
                              30000)))
    (list (first test) (/ *response-time* 1000.0)
          (if (string-equal *response* (second test))
              'T
            'F))))
```

```
; functions do-study and do-test run the functions study-sentence
; and test-sentence for all sentences of the study set and the test
; set, respectively. The function do-test also collects the
; responses of a virtual subject in a list, and sorts them with
; respect to condition

(defun do-study (set time)
  (dolist (x set)
    (study-sentence x time)))

(defun do-test (set)
  (let ((results nil))
    (dolist (x set)
      (push (test-sentence x) results))
    (mapcar #'cdr (sort results #'< :key #'first))))
;-------------------------------------------------------------------

; the function simulates the delayed condition of Anderson (1974):
; the study phase and the test phase, and reports the results

(defun do-delayed ()
  (reset)
  (do-study (permute-list *study-set*) 5)
  (study-sentence "test" 2)
  (report-results (do-test (permute-list *test-set*))))
;-------------------------------------------------------------------

; the function displays the results of the simulation:
; prints the reaction times, and the correlation and the mean
; deviation with the results of the original experiment (*a74-data*)

(defun report-results (L)
  (let ((results (mapcar #'first L)))
    (format t "~%~%")
    (format t "+----------+----------+----------+----------+----------+~%")
    (format t "|    input |  Active  |  Active  |  Passive |  Passive |~%")
    (format t "|    probe |  Active  |  Passive |  Active  |  Passive |~%")
    (format t "+----------+----------+----------+----------+----------+~%")
    (format t "|  True    |~:{ ~4,2F (~1s) |~}~%" (subseq L 0 4))
    (format t "|  False   |~:{ ~4,2F (~1s) |~}~%" (subseq L 4 8))
    (format t "+----------+----------+----------+----------+----------+~%")
    (format t "~%")

    (correlation results *a74-data*)
    (mean-deviation results *a74-data*)

    (format t "~%")))
;-------------------------------------------------------------------

; This method is automatically called by the system when a key press
; occurs in an experiment window. It is passed two parameters, the
; window in which the key press occurred and the character
; representing the key that was pressed.

(defmethod rpm-window-key-event-handler ((win rpm-window) key)
  (setf *response-time* (pm-get-time))
  (setf *response* (string key)))
```

```
; clears everything from the ACT-R state (must be put at the top
; of each model file)

(clear-all)

; This RPM command resets the Master Process to its initial state.

(pm-reset)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PARAMETERS & COMMANDS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; sets global parametners
(sgp :v t               ; displays the trace of processing
     :esc t             ; enables subsymbolic computation
     :lf 0.15           ; sets latency factor to 0.15
     :blc 10            ; sets the value added to the initial
)                       ; activation of the chunk

; sets the environment to show the current focus of attention
; (red circle on the currently processed word)

(pm-set-params :show-focus t)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; CHUNK TYPES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(chunk-type s-link              ; symbolic link
                orth            ; orthography
                sem)            ; semantics

(chunk-type c-link              ; categorization link
                inst            ; target
                type            ; standard
                ctx)            ; context of categorization

(chunk-type d-link              ; domain specification link
                ent             ; entity
                dom             ; domain
                val)            ; value

(chunk-type pw-link             ; part-whole-link
                part            ; part
                whole           ; whole
                cfg)            ; configuration
;------------------------------------------------------------------;
(chunk-type lwm                 ; linguistic working memory
                schema          ; composite schema
                entity          ; referent of composite schema
                type            ; type of referent
                base            ; conceptual content of expression
                tschema         ; temporary schema
                tentity         ; referent of temp. schema
                ttype           ; type of referent
                tbase           ; conceptual content of t. schema
                context         ; current mental space
                error           ; "error" if fail to verify
                purpose         ; "study" or "test"
                state)          ; state of processing
```

```
;------------------------------------------------------------------;
(chunk-type entity)             ; entity in Mental Space
(chunk-type domain)             ; basic domains
;------------------------------------------------------------------;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; DECLARATIVE CHUNKS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(add-dm
;==================================================================;
; Symbolic Links
;==================================================================;
; read as string of letters "..." symbolizes concept ...*

        (s0001 isa s-link orth "the"            sem g-thing-old*)
        (s0002 isa s-link orth "a"              sem g-thing-new*)
        (s0003 isa s-link orth "an"             sem g-thing-new*)
        (s0004 isa s-link orth "was"            sem g-p-past*)
        (s0005 isa s-link orth "by"             sem by*)

        (s0006 isa s-link orth "painter"        sem painter*)
        (s0007 isa s-link orth "missionary"     sem missionary*)
        (s0008 isa s-link orth "cannibal"       sem cannibal*)
        (s0009 isa s-link orth "sailor"         sem sailor*)

        (s0010 isa s-link orth "visited"        sem visit*)
        (s0011 isa s-link orth "refused"        sem refuse*)
        (s0012 isa s-link orth "chased"         sem chase*)
        (s0013 isa s-link orth "protected"      sem protect*)
        (s0014 isa s-link orth "shot"           sem shoot*)
        (s0015 isa s-link orth "questioned"     sem question*)
        (s0016 isa s-link orth "accused"        sem accuse*)
        (s0017 isa s-link orth "feared"         sem fear*)

        (stest isa s-link orth "test"           sem test*)

;==================================================================;
; Part-Whole Links
;==================================================================;
; read as concept ...* is the part of *...* schema
; e.g.: visitor* is the part of the *visit* schema, and it is an
; agent of the process

   (pw001 isa pw-link part painter*    whole *paint* cfg AG)
   (pw002 isa pw-link part sailor*     whole *sail*  cfg AG)
   (pw003 isa pw-link part missionary* whole *religion*)
   (pw004 isa pw-link part cannibal*   whole *Africa*)


    (pw005 isa pw-link part visit*    whole *visit* cfg process*)
   (pw0051 isa pw-link part visitor*  whole *visit* cfg AG)
   (pw0052 isa pw-link part visit-TH* whole *visit* cfg PAT)

    (pw006 isa pw-link part refuse*    whole *refuse* cfg process*)
   (pw0061 isa pw-link part refuse-AG* whole *refuse* cfg AG)
   (pw0062 isa pw-link part refuse-TH* whole *refuse* cfg PAT)
```

```
    (pw007 isa pw-link part chase*    whole *chase* cfg process*)
   (pw0071 isa pw-link part chasor*   whole *chase* cfg AG)
   (pw0072 isa pw-link part chase-TH* whole *chase* cfg PAT)

    (pw008 isa pw-link part protect*   whole *protect* cfg process*)
   (pw0081 isa pw-link part protector*  whole *protect* cfg AG)
   (pw0082 isa pw-link part protect-TH* whole *protect* cfg PAT)

    (pw009 isa pw-link part shoot*    whole *shoot* cfg process*)
   (pw0091 isa pw-link part shooter*  whole *shoot* cfg AG)
   (pw0092 isa pw-link part shoot-TH* whole *shoot* cfg PAT)

    (pw010 isa pw-link part question* whole *question* cfg process*)
   (pw0101 isa pw-link part question-AG* whole *question* cfg AG)
   (pw0102 isa pw-link part question-TH* whole *question* cfg PAT)

    (pw011 isa pw-link part accuse*    whole *accuse* cfg process*)
   (pw0111 isa pw-link part accuse-AG* whole *accuse* cfg AG)
   (pw0112 isa pw-link part accuse-TH* whole *accuse* cfg PAT)

    (pw012 isa pw-link part fear*    whole *fear* cfg process*)
   (pw0121 isa pw-link part fear-AG* whole *fear* cfg AG)
   (pw0122 isa pw-link part fear-TH* whole *fear* cfg PAT)

; "the" profiles the grounded thing and it is the part of the
; mental space

    (the01 isa pw-link part g-thing-old* whole MS)
    (a0001 isa pw-link part g-thing-new* whole MS)

; no whole is specified for by*

    (by001 isa pw-link part by*)

; "was" profiles the past grounded process, and it is also a part of
; the mental space

    (was01 isa pw-link part g-p-past*    whole MS cfg process*)

;==================================================================;
; Categorization Links
;==================================================================;
; read as concept ...* is an instance of the ...* schema

   (c0001 isa c-link inst painter*    type thing*)
   (c0002 isa c-link inst sailor*     type thing*)
   (c0003 isa c-link inst missionary* type thing*)
   (c0004 isa c-link inst cannibal*   type thing*)

   (c0010 isa c-link inst visit*    type process*)
   (c0011 isa c-link inst refuse*   type process*)
   (c0012 isa c-link inst chase*    type process*)
   (c0013 isa c-link inst protect*  type process*)
   (c0014 isa c-link inst shoot*    type process*)
   (c0015 isa c-link inst question* type process*)
   (c0016 isa c-link inst accuse*   type process*)
   (c0017 isa c-link inst fear*     type process*)

;==================================================================;
```

```
; Mental Space is the part of the ground, and it is specified in the
; domains of SPACE and TIME

   (g0001 isa pw-link part MS whole Ground)
   (g0002 isa d-link ent MS dom TIME  val now)
   (g0003 isa d-link ent MS dom SPACE val here)

; Basic Domains are represented by chunks:
   (TIME  isa domain)
   (SPACE isa domain)
   (COLOR isa domain)

;=================================================================;
; *cut* schema [not used in the experiment], given as an example

; parts of the *cut* schema are process, agent, patient and
; intrument (default balue knife*)
   (pwcut1 isa pw-link part cut*     whole *cut* cfg process*)
   (pwcut2 isa pw-link part cut-AG*  whole *cut* cfg AG)
   (pwcut3 isa pw-link part cut-PAT* whole *cut* cfg PAT)
   (pwcut4 isa pw-link part knife*   whole *cut* cfg INSTR)
; participants of the process are instances of human* and thing*
; schemas.
   (ccut01 isa c-link inst cut-AG*   type human*)
   (ccut02 isa c-link inst cut-PAT*  type thing*)
   (ccut03 isa c-link inst knife*    type thing*)
; knife* is specified in the domains of SPACE for its shape, and
; COLOR for its color.
   (dcut01 isa d-link ent knife* dom SPACE val knife-shape)
   (dcut02 isa d-link ent knife* dom COLOR val black)

;=================================================================;
; chunk representing initial state of the virtual subject
        (goal isa lwm context MS purpose "study" state "attending")
;=================================================================;
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; PRODUCTION RULES
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;=================================================================;
; Reading
;=================================================================;
; Production rule NEW-WORD applies for the first word in a sentence.
; It requests the identification of the object at the location
; specified by =visual-location chunk

(p new-word
"
IF the goal is to lwm
   and VS (virtual subject) is attending
   to the visual location X
   and the visual module is free
THEN
   process the screen
   and request the identification of the object
      at location X
"
```

```
   =goal>
      isa             lwm
      state           "attending"
   =visual-location>
      isa             visual-location
   =visual-state>
      isa             module-state
      modality        free
==>
   =goal>
      isa             lwm
      state           "processing"
   +visual>
      isa             visual-object
      screen-pos      =visual-location)

; Production rule FIND-NEXT applies for all non-initial words in a
; sentence. It requests the visual location of the next word.

(p find-next
"
IF the goal is lwn
   and VS is looking for the next word
THEN
   request the location of the
      object to the right,
      nearest to the currently attended object
   and attend to the new object
"
   =goal>
      isa             lwm
      state           "find-next"
==>
   +visual-location>
      isa             visual-location
      screen-x        greater-than-current
      nearest         current
   =goal>
      isa             lwm
      state           "attending")

; Production rule READ-WORD applies when the object is word, and it
; represents the transition from sting of letters to the meaning

(p read-word
"
IF the goal is lwm
   and VS is processing the screen
   and the object is idenified as the text
      with string X
THEN
   encode the word
   by retrieving the symbolic link
      that contains the string X
"
   =goal>
      isa             lwm
      state           "processing"
   =visual>
      isa             text
      value           =word
==>
   -visual-location>
   =goal>
      isa             lwm
      state           "encoding"
   +retrieval>
      isa             s-link
      orth            =word)
;------------------------------------------------------------------;
; When the word "test" is displayed, the test session begins.
; The purpose slot of the goal chunk in changed from "study" to
; "test".

(p respond-to-test
"
IF the goal is lwm
   and the word is being encoded
   and the retrieved symbolic link
      contains concept test*
THEN
   free the visual module
   and change the goal
      purpose to test
"
   =goal>
      isa             lwm
      state           "encoding"
   =retrieval>
      isa             s-link
      sem             test*
==>
   -visual-location>
   -visual>
   =goal>
      isa             lwm
      context         =ctx
      purpose         "test"
      state           "attending")
;------------------------------------------------------------------;
; Production rule ENCODE-WORD applies after the retrieval
; of the symbolic link, and it represents the access to the
; conceptual content of a lexical item

(p encode-word
"
IF the goal is lwm
   and VS is encoding a word
   and the retrieved symbolic link
      contains the concept X
THEN
   request the retrieval of the part-whole link
   between the concept X and its base
"
   =goal>
      isa             lwm
      state           "encoding"
   =retrieval>
      isa             s-link
      sem             =concept
==>
   =goal>
      isa             lwm
      state           "reading"
   +retrieval>
      isa             pw-link
      part            =concept)

;==================================================================;
; Constructional Schemas
;==================================================================;
;------------------------------------------------------------------;
; Nominals
;------------------------------------------------------------------;
; Production rules INTRODUCE-NOMINAL-NEW and CATEGORIZE-NOMINAL-
; NEW represent the constructional schema for a(n)-thing combination

(p introduce-nominal-new
"
IF the goal is lwm
   and the new grounded thing schema
      was retrieved
THEN
   add new entity to mental space
   ground it
   note that tschema is new grounded thing
      and read the next word
"
   =goal>
      isa             lwm
      context         =MS
      state           "reading"
   =retrieval>
      isa             pw-link
      part            g-thing-new*
      whole           =base
==>
   =nom>
      isa             entity
   =gr-link>
      isa             pw-link
      part            =nom
      whole           =MS
   =goal>
      isa             lwm
      tschema         g-thing-new*
      tentity         =nom
      ttype           thing*
      tbase           =base
      state           "find-next")
```

```
(p categorize-nominal-new
"
IF the goal is lwm
   and there is no schema
   and tschema is new grounded thing
   and the concept was retrieved
THEN
   categorize the new entity as an instance of
      the retrieved concept
   and change the schema to grounded thing
      and read the next word
"
   =goal>
      isa             lwm
      schema          nil
      tschema         g-thing-new*
      tentity         =nom
      ttype           thing*
      context         =ctx
      state           "reading"
   =retrieval>
      isa             pw-link
      part            =con
      whole           =base
==>
   =c-link>
      isa             c-link
      inst            =nom
      type            =con
      ctx             =ctx
   =goal>
      isa             lwm
      schema          g-thing*
      entity          =nom
      type            =con
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      state           "find-next")

; production rule CAETGORIZE-NOMINAL-NEW-LM applies as the second
; production rule of the a(n)-thing schema in cases when the nominal
; is not at the beginning of a sentence.

(p categorize-nominal-new-LM
"
IF the goal is lwm
   and there is a schema
   and tschema is new grounded thing
   and the concept was retrieved
THEN
   categorize the new entity as an instance of
      the retrieved concept
   and change tschema to grounded thing
      and read the next word
"
=goal>
      isa             lwm
      schema          nil
      tschema         g-thing-new*
      tentity         =nom
      ttype           thing*
      context         =ctx
      state           "reading"
   =retrieval>
      isa             pw-link
      part            =con
      whole           =base
==>
   =c-link>
      isa             c-link
      inst            =nom
      type            =con
      ctx             =ctx
   =goal>
      isa             lwm
      tschema         g-thing*
      tentity         =nom
      ttype           =con
      tbase           =base
      state           "find-next")

;-------------------------------------------------------------------;
; NOMINAL-OLD applies upon reading the word "the"

(p nominal-old
"
IF the goal is lwm
   and the old grounded thing schema
      was retrieved
THEN
   note that tschema is old grounded thing
      of type thing
      and read the next word
"
   =goal>
      isa             lwm
      state           "reading"
   =retrieval>
      isa             pw-link
      part            g-thing-old*
      whole           =base
==>
   =goal>
      isa             lwm
      tschema         g-thing-old*
      ttype           thing*
      tbase           =base
      state           "find-next")

;-------------------------------------------------------------------;
; Production rules CATEGORIZE-OLD-NOMINAL and CHECK-NOMINAL-YES-TR
; represent the constructional schema for the-thing combination
```

```
(p categorize-old-nominal
"
IF the goal is lwm
   and tschema is old grounded thing
   and a concept was retrieved
THEN
   request the retrieval of the
      referent of the concept
   and note that schema is grounded thing
"
   =goal>
      isa              lwm
      tschema          g-thing-old*
      tentity          nil
      ttype            thing*
      context          =ctx
      state            "reading"
   =retrieval>
      isa              pw-link
      part             =concept
      whole            =base
==>
   +retrieval>
      isa              c-link
      type             =concept
      ctx              =ctx
   =goal>
      isa              lwm
      tschema          g-thing*
      ttype            =concept
      tbase            =base
      state            "find-referent")


;--------------------------------------------------------------------;
; Production rules for finding the referents of the-thing schemas

; CHECK-NOMINAL-YES-TR applies when the referent of the first
; nominal is retrieved

(p check-nominal-yes-TR
"
IF the goal is lwm
   and there is no schema
   and the referent was retrieved
THEN
   change the schema to grounded thing
      and read the next word
"
   =goal>
      isa              lwm
      schema           nil
      tschema          g-thing*
      ttype            =concept
      tbase            =base
      state            "find-referent"
   =retrieval>
      isa              c-link
      inst             =nominal
      type            =concept
==>
   =goal>
      isa             lwm
      schema          g-thing*
      entity          =nominal
      type            =concept
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      state           "find-next")

; CHECK-NOMINAL-NO-TR applies when the referent of the first nominal
; cannot be retrieved. It adds an entity to mental space, grounds
; and categorizes it like the a(n)-thing schema

(p check-nominal-no-TR
"
IF the goal is lwm
   and there is no schema
   and the referent was not retrieved
THEN
   add new entity to the mental space,
   categorize it,
   and ground it,
   change the schema to grounded thing,
      and read the next word
"
   =goal>
      isa             lwm
      schema          nil
      tschema         g-thing*
      ttype           =concept
      tbase           =base
      context         =ctx
      purpose         "study"
      state           "find-referent"
   =retrieval>
      isa             error
==>
   =nom>
      isa             entity
   =c-link>
      isa             c-link
      inst            =nom
      type            =concept
      ctx             =ctx
   =pw-link>
      isa             pw-link
      part            =nom
      whole           =ctx
   =goal>
      isa             lwm
      schema          g-thing*
      entity          =nom
      type            =concept
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      state           "find-next")

; CHECK-NOMINAL-YES-LM applies when the referent of the second
; nominal is retrieved

(p check-nominal-yes-LM
"
IF the goal is lwm
   and there is a schema
   and the referent was retrieved
THEN
   note that tentity is the referent
      and read the next word
"
   =goal>
      isa             lwm
      schema          =schema
      tschema         g-thing*
      ttype           =concept
      tbase           =base
      state           "find-referent"
   =retrieval>
      isa             c-link
      inst            =nominal
      type            =concept
==>
   =goal>
      isa             lwm
      tentity         =nominal
      state           "find-next")

; CHECK-NOMINAL-NO-LM applies when the referent of the second
; nominal cannot be retrieved. It adds an entity to mental
; space, grounds and categorizes it like the a(n)-thing schema

(p check-nominal-no-LM
"
IF the goal is lwm
   and there is a schema
   and the referent was not retrieved
THEN
   add new entity to the mental space,
   categorize it,
   and ground it,
   and read the next word
"
   =goal>
      isa             lwm
      schema          =schema
      tschema         g-thing*
      ttype           =concept
      context         =ctx
      purpose         "study"
      state           "find-referent"
   =retrieval>
      isa             error
==>
   =nom>
      isa             entity
   =c-link>
      isa             c-link
      inst            =nom
      type            =concept
      ctx             =ctx
   =pw-link>
      isa             pw-link
      part            =nom
      whole           =ctx
   =goal>
      isa             lwm
      tschema         g-thing*
      tentity         =nom
      ttype           =concept
      tbase           =base
      state           "find-next")

;------------------------------------------------------------------;
; Production rule NO-NEXT applies at the end of the sentence.
; Since previous schema is nominal, it will request the composition
; of this nominal with the previous schema, which is a process.

(p no-next
"
IF the goal is lwm
   and there is no next word
THEN
   free the visual module
   and integrate the nominal
"
   =goal>
      isa             lwm
      state           "attending"
   =visual-location>
      isa             error
==>
   -visual>
   -visual-location>
   =goal>
      isa             lwm
      state           "integrate-LM")

;------------------------------------------------------------------;
; Clauses
;------------------------------------------------------------------;
; Production rules for passive sentences

; GROUND-CLAUSE-BE adds new process to the mental space. Together
; with CATEGORIZE-CLAUSE-PASSIVE it forms the constructional
; schema for was-process combination.
```

```
(p ground-clause-BE
"
IF the goal is lwm
      and schema is grounded thing
   and the past grounded process (was) was retrieved
THEN
   add new process to the mental space
   and ground it
   and read next word
"
   =goal>
      isa             lwm
      schema          g-thing*
      context         =ctx
      purpose         "study"
      state           "reading"
   =retrieval>
      isa             pw-link
      part            g-p-past*
      whole           =base
      cfg             process*
==>
   =cl>
      isa             entity
   =gr-link>
      isa             pw-link
      part            =cl
      whole           =ctx
   =goal>
      isa             lwm
      tschema         g-process*
      tentity         =cl
      ttype           process*
      tbase           =base
      state           "find-next")

; CATEGORIZE-CLAUSE-PASSIVE applies when "was" was read, and the
; retrieved concept is a process. (Second production rule for the
; was-process constructional schema.)

(p categorize-clause-passive
"
IF the goal is lwm
      and previous schema is grounded thing
      and current schema is grounded process
   and the retrieved concept is a process
THEN
   categorize the current process as an
      instance of the retrieved process type
   and request the retrieval of the PATIENT
      participant of the process
"
   =goal>
      isa             lwm
      schema          g-thing*
      tschema         g-process*
      tentity         =cl
      context         =ctx
      purpose         "study"
      state           "reading"
   =retrieval>
      isa             pw-link
      part            =con
      whole           =w
      cfg             process*
==>
   =c-link>
      isa             c-link
      inst            =cl
      type            =con
      ctx             =ctx
   =goal>
      isa             lwm
      tschema         g-process*
      tentity         =cl
      ttype           =con
      tbase           =w
      state           "integrate-TR"
   +retrieval>
      isa             pw-link
      whole           =w
      cfg             PAT)
;-------------------------------------------------------------------;
; Production rules for active sentences

; CATEGORIZE-AND-GROUND-CLAUSE-ACTIVE applies for active sentences
; upon reading the verb. Together with the production rule INTEGRATE
; -TR, it represents the constructional schema for thing-process
; combination.

(p categorize-and-ground-clause-active
"
IF the goal is lwm
      and schema is grounded thing
   and the process (not was) was retrieved
THEN
   add new process to the mental space,
   ground it
   and categorize it s an instance of the
      retrieved process type
   and request the retrieval of the AGENT
      participant of the process
"
   =goal>
      isa             lwm
      schema          g-thing*
      context         =ctx
      purpose         "study"
      state           "reading"
   =retrieval>
      isa             pw-link
      part            =con
      whole           =w
      cfg             process*
==>
   =cl>
      isa             entity
   =gr-link>
      isa             pw-link
      part            =cl
      whole           =ctx
   =c-link>
      isa             c-link
      inst            =cl
      type            =con
      ctx             =ctx
   =goal>
      isa             lwm
      tschema         g-process*
      tentity         =cl
      ttype           =con
      tbase           =w
      state           "integrate-TR"
   +retrieval>
      isa             pw-link
      whole           =w
      cfg             AG)
;-------------------------------------------------------------------;
; INTEGRATE-TR is the second production rule for constructional
; schemas thing-process and thing-[was process].

(p integrate-TR
"
IF the goal is lwm
      previous schema is grounded thing
      current schema is grounded process
   and the elaboration site (part) of the process was retrieved
THEN
   categorize nominal as a participant
      of the process
   with role X
   and note that current schema is grounded process
      and read next word
"
   =goal>
      isa             lwm
      schema          g-thing*
      entity          =nominal
      type            =t2
      tschema         g-process*
      tentity         =clause
      ttype           =t1
      tbase           =base
      context         =ctx
      state           "integrate-TR"
   =retrieval>
      isa             pw-link
      part            =esite
      cfg             =val
==>
   =c-link>
      isa             c-link
      inst            =nominal
      type            =esite
      ctx             =ctx
   =pw-link>
      isa            pw-link
      part           =nominal
      whole          =clause
      cfg            =val
   =goal>
      isa            lwm
      schema         g-process*
      entity         =clause
      type           =t1
      base           =base
      tschema        nil
      tentity        nil
      ttype          nil
      tbase          nil
      context        =ctx
      state          "find-next")

;------------------------------------------------------------------;
; PROCESS-BY applies for passive sentences upon retrieval of by.

(p process-BY
"
IF the goal is lwm
   and the current schema is grounded process
   and the concept by* was retrieved
THEN
"
   =goal>
      isa            lwm
      schema         g-process*
      context        =ctx
      state          "reading"
   =retrieval>
      isa            pw-link
      part           by*
==>
   =goal>
      isa            lwm
      schema         by*
      state          "find-next")
;------------------------------------------------------------------;


;==================================================================;
; Test Session
;==================================================================;
; Nominals are processed by the same production rules with the
; encoding stage. The difference is when no referrent can be
; retrieved during the study, the referent is added, but during the
; test no referent is added and an error is noted. For the failure
; to retrieve the referent of the first nominal, production rule
; CHECK-NOMNAL-NO-TR-TEST is applied; and for the failure to
; retrieve the referent of the second nominal, production rule
; CHECK-NOMNAL-NO-TR-TEST is applied. They apply for both active
; and passive sentences.
```

```
(p check-nominal-no-TR-test
"
IF the goal is lwm
   and tschema is grounded thing
   and no referent was retrieved
      for the thing
THEN
   make the schema grounded thing,
      note the error
      and read the next word
"
   =goal>
      isa            lwm
      tschema        g-thing*
      ttype          =concept
      tbase          =base
      purpose        "test"
      state          "find-referent"
   =retrieval>
      isa            error
==>
   =goal>
      isa            lwm
      schema         g-thing*
      type           =concept
      base           =base
      tschema        nil
      tentity        nil
      ttype          nil
      tbase          nil
      error          "error"
      state          "find-next")

(p check-nominal-no-LM-test
"
IF the goal is lwm
   and there is a shcema
      and tschema is grounded thing
   and no referent was retrieved
      for the thing
THEN
      note the error
      and read the next word
"
   =goal>
      isa            lwm
      schema         =schema
      tschema        g-thing*
      purpose        "test"
      state          "find-referent"
   =retrieval>
      isa            error
==>
   =goal>
      isa            lwm
      schema         =schema
      tschema        g-thing*
      error          "error"
      state          "find-next")
```

```
;---------------------------------------------------------------------;
; Production rules CHECK-CLAUSE-BE, CHECK-GROUNDED-CLAUSE-PASSIVE,
; and CLAUSE-YES-CHECK-TR-PASSIVE are steps 2 & 3 of the proposi-
; tional strategy for passive sentences.

; CHECK-CLAUSE-BE applies upon reading "was", and instead of intro-
; ducing a new process to the mental space, waits for the past-
; participle.

(p check-clause-BE
"
IF the goal is lwm
      and schema is grounded thing
    and the concept grouded past process was retrieved
THEN
   note that tschema is grounded process
   and read the next word
"
   =goal>
      isa             lwm
      schema          g-thing*
      purpose         "test"
      state           "reading"
   =retrieval>
      isa             pw-link
      part            g-p-past*
      whole           =base
      cfg             process*
==>
   =goal>
      isa             lwm
      tschema         g-process*
      ttype           process*
      tbase           =base
      state           "find-next")

; CHECK-GROUNDED-CLAUSE-PASSIVE applies upon reading the past-
; participle, and checks whether there is a process of the same type
; in the mental space

(p check-grounded-clause-passive
"
IF the goal is lwm
      schema is grounded thing
      and tschema is grounded process
   and the process was retrieved
THEN
   check whether there is a process of the
      same type in the mental space
"
   =goal>
      isa             lwm
      schema          g-thing*
      tschema         g-process*
      context         =ctx
      purpose         "test"
      state           "reading"
   =retrieval>
      isa             pw-link
      part            =concept
      whole           =base
      cfg             process*
==>
   +retrieval>
      isa             c-link
      type            =concept
      ctx             =ctx
   =goal>
      isa             lwm
      tschema         g-process*
      ttype           =concept
      tbase           =base
      state           "check-clause-pas")

; CLAUSE-YES-CHECK-TR-PASSIVE checks whether the patient of
; the probe sentence was the patient of the process in the
; mental space

(p clause-yes-check-TR-passive
"
IF the goal is lwm
      schema is grounded thing,
      tschema is grounded process
   and there is a process of the same
      type in the mental space
THEN
   check whether the thing participated
      in the process with the same role
"
   =goal>
      isa             lwm
      schema          g-thing*
      entity          =nominal
      tschema         g-process*
      purpose         "test"
      state           "check-clause-pas"
   =retrieval>
      isa             c-link
      inst            =clause
==>
   +retrieval>
      isa             pw-link
      part            =nominal
      whole           =clause
      cfg             PAT
   =goal>
      isa             lwm
      tentity         =clause
      state           "check-TR")

;-------------------------------------------------------------------;
; Production rules CHECK-GROUNDED-CLAUSE and CLAUSE-YES-CHECK-TR
; represent the steps 2 & 3 of the propositional strategy for
; active sentences.
; CHECK-GROUNDED-CLAUSE applies upon reading the verb and checks
; whether there is a process of the same type in the mental space.

(p check-grounded-clause
"
IF the goal is lwm
      and schema is grounded thing
   and the process was retrieved
THEN
   check whether there is a process in mental space
      that was categorized as an instance of this
      process type
   and note that tschema is grounded process
"
   =goal>
      isa              lwm
      schema           g-thing*
      context          =ctx
      purpose          "test"
      state            "reading"
   =retrieval>
      isa              pw-link
      part             =concept
      whole            =base
      cfg              process*
==>
   +retrieval>
      isa              c-link
      type             =concept
      ctx              =ctx
   =goal>
      isa              lwm
      tschema          g-process*
      ttype            =concept
      tbase            =base
      state            "check-clause")

; CLAUSE-YES-CHECK-TR checks whether the agent of the probe
; sentence was the agent of the process in the mental space.

(p clause-yes-check-TR
"
IF the goal is lwm
      and schema is grounded thing
      and tschema is grounded process
   and there is a process of the same type
      in the mental space
THEN
   request the retrieval of the part-whole link
      to check whether the thing participated
      in the process with the same role (AGENT)
"
   =goal>
      isa              lwm
      schema           g-thing*
      entity           =nominal
      tschema          g-process*
      purpose          "test"
      state            "check-clause"
   =retrieval>
      isa             c-link
      inst            =clause
==>
   +retrieval>
      isa             pw-link
      part            =nominal
      whole           =clause
      cfg             AG
   =goal>
      isa             lwm
      tentity         =clause
      state           "check-TR")

;-----------------------------------------------------------------;
; Production rules CLAUSE-NO-ACTIVE and CLAUSE-NO-PASSIVE apply for
; the cases when no referent for the process can be retrieved.
; However, since the referent can be always retrieved, they are
; never used.

(p clause-no-active
"
IF the goal is lwm
      schema is grounded thing
      tschema is grounded process
   and there is no process in the mental space
      of the same type
THEN
   change schema to grounded process
      and note the error
"
   =goal>
      isa             lwm
      schema          g-thing*
      tschema         g-process*
      ttype           =concept
      tbase           =base
      purpose         "test"
      state           "check-clause"
   =retrieval>
      isa             error
==>
   =goal>
      isa             lwm
      schema          g-process*
      type            =process
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      error           "error"
      state           "find-next")
```

```
(p clause-no-passive
"
IF the goal is lwm
      schema is grounded thing
      tschema is grounded process
   and there is no process in the mental space
      of the same type
THEN
   change schema to grounded process
      and note the error
"
   =goal>
      isa             lwm
      schema          g-thing*
      tschema         g-process*
      ttype           =concept
      tbase           =base
      purpose         "test"
      state           "check-clause-pas"
   =retrieval>
      isa             error
==>
   =goal>
      isa             lwm
      schema          g-process*
      type            =process
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      error           "error"
      state           "find-next")

;-----------------------------------------------------------------;
; Production rules CHECK-TR-YES and CHECK-TR-NO apply for both
; passive and active sentences. In cases the first nominal in the
; probe sentence participated in the process profiled by the verb
; with the same role (i.e. VS successfully retrieves pw-link between
; the process and the nominal), CHECK-TR-YES is selected. Otherwise
; CHECK-TR-NO is seleected, and it changes the error index to
; "error".

(p check-TR-yes
"
IF the goal is lwm
      and schema is grounded thing
      and the constructed schema is grounded process
      and the purpose is to test
      whether the thing participated
      in the process with the same role
   and the chunk was retrieved
      containing such information
THEN
      encode that schema
      is the grounded process
      and read the next word
"
```

```
   =goal>
      isa             lwm
      schema          g-thing*
      entity          =nominal
      tschema         g-process*
      tentity         =clause
      ttype           =process
      tbase           =base
      purpose         "test"
      state           "check-TR"
   =retrieval>
      isa             pw-link
      part            =nominal
      whole           =clause
==>
   =goal>
      isa             lwm
      schema          g-process*
      entity          =clause
      type            =process
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      state           "find-next")

(p check-TR-no
"
IF the goal is lwm
      and schema is grounded thing
      and the constructed schema is grounded process
      and the purpose is to test
      whether the thing participated
      in the process with the same role
   and no chunk was retrieved
      containing such information
THEN
      note the error and
      encode that schema
      is the grounded process
      and read the next word
"
   =goal>
      isa             lwm
      schema          g-thing*
      tschema         g-process*
      tentity         =clause
      ttype           =process
      tbase           =base
      purpose         "test"
      state           "check-TR"
   =retrieval>
      isa             error
==>
   =goal>
      isa             lwm
      schema          g-process*
      entity          =clause
      type            =process
      base            =base
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      error           "error"
      state           "find-next")
;------------------------------------------------------------------;

(p check-LM-active
"
IF the goal is lwm
   and schema is process
   and tschema is grounded thing
   with the referent
THEN
   request the retrieval of part-whole link
     to check whether the thing participated
     in the process with the same role (PATIENT)
"
   =goal>
      isa             lwm
      schema          g-process*
      entity          =clause
      tschema         g-thing*
      tentity         =nominal
      purpose         "test"
      state           "integrate-LM"
==>
   +retrieval>
       isa            pw-link
       part           =nominal
       whole          =clause
       cfg            PAT
   =goal>
       isa            lwm
       state          "checking-LM")

(p check-LM-passive
"
IF the goal is lwm
   and schema is by
   and tschema is grounded thing
   with the referent
THEN
   request the retrieval of part-whole link
     to check whether the thing participated
     in the process with the same role (AGENT)
"
   =goal>
      isa             lwm
      schema          by*
      entity          =clause
      tschema         g-thing*
      tentity         =nominal
      error           nil
      purpose         "test"
      state           "integrate-LM"
```

```
==>
   +retrieval>
      isa               pw-link
      part              =nominal
      whole             =clause
      cfg               AG
   =goal>
      isa               lwm
      schema            g-process*
      state             "checking-LM")

(p checking-LM-yes
"
IF the goal is lwm
      and schema is grounded process
      and tschema is grounded thing
   and the part of process was retrieved
THEN
   categorize the thing as instance of
      the part of the process
   note that the thing participated in
      the process with role =role
   and prepare for the next sentence
"
   =goal>
      isa               lwm
      schema            g-process*
      entity            =clause
      type              =type1
      base              =base1
      tschema           g-thing*
      tentity           =nominal
      context           =ctx
      purpose           "test"
      state             "checking-LM"
   =retrieval>
      isa               pw-link
      part              =nominal
      whole             =clause
==>
   =goal>
      isa               lwm
      schema            g-process*
      entity            =clause
      type              =type1
      base              =base1
      tschema           nil
      tentity           nil
      ttype             nil
      tbase             nil
      context           =ctx
      state             "answer")
```

```
(p checking-LM-no
"
IF the goal is lwm
      and schema is grounded process
      and tschema is grounded thing
   and the part of process was retrieved
THEN
   categorize the thing as instance of
      the part of the process
   note that the thing participated in
      the process with role =role
   and prepare for the next sentence
"
   =goal>
      isa             lwm
      schema          g-process*
      entity          =clause
      type            =type1
      base            =base1
      tschema         g-thing*
      tentity         =nominal
      context         =ctx
      purpose         "test"
      state           "checking-LM"
   =retrieval>
      isa             error
==>
   =goal>
      isa             lwm
      schema          g-process*
      entity          =clause
      type            =type1
      base            =base1
      tschema         nil
      tentity         nil
      ttype           nil
      tbase           nil
      context         =ctx
      error           "error"
      state           "answer")
; Production rule INTEGRATE-LM-ACTIVE applies for the last nominal
; of the active sentence. Together with INTEGRATING-LM it forms the
; constructional schema for process-thing combination.

(p integrate-LM-active
"
IF the goal is lwm
   and schema is process
   and tschema is grounded thing
   with a nominal
THEN
   request the retrieval of the
      PATIENT participant of the process
"
   =goal>
      isa             lwm
      schema          g-process*
      base            =base1
      tschema         g-thing*
      purpose          "study"
      state            "integrate-LM"
==>
   +retrieval>
       isa             pw-link
       whole           =base1
       cfg             PAT
   =goal>
       isa             lwm
       state           "integrating-LM")

(p integrating-LM
"
IF the goal is lwm
      and schema is grounded process
      and tschema is grounded thing
   and the part of process was retrieved
THEN
   categorize the thing as instance of
      the part of the process
   note that the thing participated in
      the process with role =role
   and prepare for the next sentence
"
   =goal>
      isa              lwm
      schema           g-process*
      entity           =clause
      type             =type1
      base             =base1
      tschema          g-thing*
      tentity          =nominal
      context          =ctx
      purpose          "study"
      state            "integrating-LM"
   =retrieval>
      isa              pw-link
      part             =esite
      whole            =base1
      cfg              =role
==>
   =c-link>
      isa              c-link
      inst             =nominal
      type             =esite
      ctx              =ctx
   =pw-link>
      isa              pw-link
      part             =nominal
      whole            =clause
      cfg              =role
   =goal>
      isa              lwm
      schema           g-process*
      entity           =clause
      type             =type1
      base             =base1
      tschema          nil
      tentity          nil
      ttype           nil
      tbase           nil
      context         =ctx
      purpose         "study"
      state           "done"
   +goal>
      isa             lwm
      purpose         "study"
      state           "attending"
      context         =ctx)

;-------------------------------------------------------------------;
; Production rule INTEGRATE-LM-PASSIVE applies for the last nominal
; of the passive sentence. Together with INTEGRATING-LM it forms the
; constructional schema for process-by-thing combination.

(p integrate-LM-passive
"
IF the goal is lwm
   and schema is by
   and tschema is grounded thing
THEN
   request the retrieval of the
      AGENT participant of the process
   and change the schema to process
"
   =goal>
      isa             lwm
      schema          by*
      base            =base1
      tschema         g-thing*
      purpose         "study"
      state           "integrate-LM"
==>
   +retrieval>
      isa             pw-link
      whole           =base1
      cfg             AG
   =goal>
      isa             lwm
      schema          g-process*
      state           "integrating-LM")

;-------------------------------------------------------------------;
(p check-LM-passive-error
"
IF the goal is lwm
   and schema is by
   and tschema is grounded thing
   and there was an error
THEN
   answer
"
   =goal>
      isa             lwm
      schema          by*
      entity          =clause
      tschema         g-thing*
      tentity         =nominal
      error           "error"
      purpose         "test"
      state           "integrate-LM"
==>
   =goal>
      isa             lwm
      schema          g-process*
      state           "answer")

;==================================================================;
; Responses
;==================================================================;
; Production rule RESPOND-YES applies when no error was detected
; during verification. The response key is pressed, and the subject
; prepares to read the next sentence.

(p respond-yes
"
IF the goal is lwm
   and no error was detected,
   and the manual module is free
THEN
   press the key k
   and set the goal
       to read a new sentence
"
   =goal>
      isa             lwm
      context         =ctx
      error           nil
      state           "answer"
   =manual-state>
      isa             module-state
      modality        free
==>
   -visual>
   +manual>
      isa             press-key
      key             "k"
   +goal>
      isa             lwm
      context         =ctx
      purpose         "test"
      state           "attending")

; Production rule RESPOND-NO is similar to the production rule
; RESPOND-YES, but it applies when an error was detected
; during verification. VS presses the response key and prepares
; to read the next sentence.
(p respond-no
"
IF the goal is lwm
   and an error was detected,
   and the manual module is free
THEN
   press the key d
   and set the goal
       to read a new sentence
"
```

```
   =goal>
      isa             lwm
      context         =ctx
      error           "error"
      state           "answer"
   =manual-state>
      isa             module-state
      modality        free
==>
   -visual>
   +manual>
      isa             press-key
      key             "d"
   +goal>
      isa             lwm
      context         =ctx
      purpose         "test"
      state           "attending")

;===================================================================;
; puts the chunk goal (specified above) into the goal buffer

(goal-focus goal)

; sets the initial point of attention

(pm-set-visloc-default :attended new :screen-x lowest)
;===========================END OF FILE============================;
```