Roma Tre University
Ph.D. in Computer Science and Engineering

A Methodology for Generating Grammars for Multimodal Languages

Arianna D'Ulizia

A Methodology for Generating Grammars for Multimodal Languages

A thesis presented by
Arianna D'Ulizia
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Informatics and Automation

February 2009

COMMITTEE:
Ing. Fernando Ferri

REVIEWERS:
Prof. Esteban Zimanyi
Prof. Irina Kondratova

To my parents and Domenico

"Tell me and I'll forget,
Show me and I may remember,
Involve me and I'll understand"

Abstract

Human communication is naturally multimodal. People normally interact through several communication channels, such as gesture, drawing, handwriting, facial expressions, gaze in combination with speech or speech only, which is the prevalent modality. This synergistic use of multiple interaction channels makes human communication flexible, natural and robust. In the last years several efforts have been made to endow computer interface with similar flexibility, naturalness and robustness. The research presented in this thesis represents one of this effort.

The main contributions of this thesis are twofold. First of all, it provides a methodology for multimodal language definition that is general enough to be applicable for whatever modalities and in whichever domains. Secondly, it provides an efficient incremental learning algorithm that, following an approach “by example”, allows to generate the production rules of the defined grammar starting from the acceptable multimodal sentences.

Acknowledgments

First, I would like to thank my supervisor Fernando Ferri at CNR who suggested many of the ideas realized in this work, encouraged me to tackle the problems during the whole period of studies, and gave many important comments on the text of the thesis.

I also want to express my gratitude to Patrizia Grifoni and all the members of the Multi Media & Modal Laboratory group at CNR who have created a great working atmosphere and provided a lot of useful feedback.

Finally, I would like to thank Domenico, my parents, and my relatives and friends for constant moral support and belief in my ability to do the work and write this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Human communication is naturally multimodal. People normally interact through several communication channels, such as gesture, drawing, handwriting, facial expressions, gaze in combination with speech or speech only, which is the prevalent modality. This synergistic use of multiple interaction channels makes human communication flexible, natural and robust. In the last years several efforts have been made to endow computer interface with similar flexibility, naturalness and robustness.

These efforts are producing an evolution of traditional Graphical User Interfaces (GUI) into multimodal interfaces incorporating human modalities, such as gesture, written or spoken language, as well as gaze and facial expressions into the computer system. Consequently, in the field of Human-Computer Interaction (HCI), that is a discipline "concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them [Hew92]", an increasing importance has been gained by the study of multimodal interaction, that refers to the "interaction with the virtual and physical environment through natural modes of communication" [Bou03].

Three of the main advantages of multimodal interfaces, compared to traditional GUI, consist in making human-computer communication more intuitive, natural and efficient, enabling a broader spectrum of users with different ages, skill levels and abilities to access technological devices, and increasing the level of freedom offered to users. These advantages are also demonstrated by many research studies [ODK97] [OC00] [OCW00] that emphasize the enhancement of multimodal interface in terms of usability, accessibility, flexibility and efficiency, compared to unimodal ones. In particular, a multimodal interaction is intrinsically natural because of the naturalness of human communication; it improves usability because it provides users with the means to choose among different available modalities, as s/he prefers. Moreover, multimodality improves accessibility to the device by encompassing a broader spectrum of users. Finally, it offers improved flexibility and interaction efficiency.

Several aspects characterize multimodal interaction compared to usual interaction through GUIs. Firstly, a GUI requires atomic and unambiguous inputs (such as the selection of an element by mouse or the input of a character by keyboard), whereas a multimodal interaction involves several simultaneous inputs that have to be recognized and opportunely combined by managing the uncertainty of inputs through probabilistic techniques. The process of integrating information coming from various input modalities and combining them into a complete command is called *multimodal fusion*. Secondly, temporal constraints of inputs have to be taken into account in a multimodal interaction process; consequently it requires to define a time-sensitive architecture and to record time intervals of each modality. Finally, in a GUI the output messages are conveyed only visually, whereas in a multimodal system a way of arranging outputs through the various channels has to be found in order to provide the user with consistent feedback. This process is called *multimodal fission*, in contrast with multimodal fusion.

Consequently, in the design and development of a multimodal system the two main challenges to face are the multimodal fusion and fission processes. My specific concern in this thesis is with the fusion of multiple input modalities.

In the literature, two different approaches to the fusion process have been proposed. The first one, which will be referred as grammar-based approach, combines the multimodal inputs at grammar level. This means that the different unimodal inputs are considered as a unique multimodal input by using the multimodal grammar specification. Subsequently, the dialogue parser applies the grammar rules to interpret the multimodal sentence. The second strategy, which is referred to as dialogue-based approach, combines the multimodal inputs at dialogue level. This means that the different unimodal inputs are distinctly interpreted and then they are combined by the dialogue management system.

A comparison of these two approaches [MPA06] showed that the grammar-based paradigm is the most natural one as it is more coherent with the human-human communication paradigm in which the dialogue is seen as a unique and multimodal communication act. Moreover, this approach allows an easier inter-modality disambiguation. However, the use of a grammar implies a higher computational complexity for generating the rule sets of the grammar as well as a highly expert user that is skilled in computational linguistics for writing the grammar.

As the benefits of the grammar-based paradigm meet the requirements of naturalness and flexibility for an efficient multimodal interaction, the problem to face is to overcome the deficiencies of this paradigm that preclude its use in multimodal language definition.

This thesis intends to provide a solution to this problem. Specifically, for dealing with the complexity of grammar definition it is proposed the adoption of a “by example” paradigm, which allows the end user to provide concrete examples of multimodal sentences that have to be recognized, and the system automatically generates the grammar rules to parse those examples. In such a way no skilled grammar writers are needed, but even non-expert users can define multimodal grammars. Moreover, to overcome the issue of the high computational complexity of the grammar-based paradigm, an efficient grammatical inference algorithm has been applied that allows to generate the grammar rules starting from the acceptable multimodal sentences (positive sample) in polynomial time.

Therefore, actually, the objective of this thesis concerns the development of an innovative multimodal languages editor that, unlike task-specific multimodal grammars, allows to define complex multimodal expressions, integrating whatever input modalities and maintaining at the same time a low computational complexity.

Specifically, the editor relies on a multimodal grammar, the Multimodal Attribute Grammar, which is an extension of attribute grammars for multimodal input processing [Knu68]. The choice of this kind of grammar has been led by the capability to manage whatever modalities and to represent temporal constraints into the grammar rules.

To generate the grammar, a computationally efficient algorithm for grammatical inference, which extends the inductive CYK algorithm proposed by Cocke-Younger-Kasami [Kas65] to multimodal sentences, has been developed. This algorithm has been a valuable starting point as enabled to learn the multimodal grammar from positive sample strings in polynomial time.

The activity to attain this result started from modeling multimodal inputs, as they will compose the alphabet of terminal elements of the grammar. After a careful comparative analysis of existing grammars for natural language, the attribute grammar has been chosen due to its capability to represent multiple modalities and temporal constraints and consequently an original evolution of this grammar adapted to define multimodal sentences has been proposed. The analysis of the grammatical inference methods existing in literature has resulted in the choice of the inductive CYK algorithm for its acceptable computational time, and its extension for the inductive inference of the defined multimodal grammar has been proposed. Finally, the multimodal language editor has been designed, implemented and validated in its applicability through several experiments.
At the end of these activities, the main contributions of this thesis are twofold:

- a grammatical framework for multimodal language definition that is general enough to be applicable for whatever modalities and in whichever domains,
- an efficient incremental learning algorithm that, following an approach “by example”, allows to generate the production rules of the defined grammar starting from the acceptable multimodal sentences (positive sample).

The remainder of this dissertation presents the results of my research according to the following structure.

Chapter 2 gives an introduction to multimodal human-computer interaction, giving some preliminary definitions that will be used during this dissertation and focusing on conceptual and architectural aspects of multimodal interaction systems. The chapter also discusses the main kinds of modalities that can convey information from a human user to a computational machine, and the advantages of multimodal interfaces in terms of accessibility, robustness, stability and expressive power.

Chapter 3 presents an overview of research related to multimodal fusion strategies, classifying them according to the data fusion level (e.g. the fusion process takes place at recognition, decision or in both levels). The chapter also provides a critical survey of the literature on multimodal grammars approaches, as the multimodal language processor described in this thesis uses this kind of approach.

Chapter 4 surveys current literature on methodologies for inferring context-free grammars from sample sentences. After introducing some preliminary definitions and notations concerning learning and inductive inference, the attention will be focused on the existing models of learning. The last section of the chapter will explore the state of the art concerning the algorithms for learning context-free languages and grammars.

Chapter 5 describes the theoretical foundations at the base of the proposed multimodal grammar editor. The aim is to allow an easy multimodal grammar specification, overcoming the difficulties arising from the textual description of the grammar production rules (that require the skill of computer programmers and linguistic experts together) and proposing a "by example" approach in order to define a multimodal grammar in a very intuitive way.

Chapter 6 details the design process that has been followed to develop the Multimodal Grammar Editor (MGE). This editor constitutes one of the many system components needed in the construction of the Multimodal Language Processing (M2LP) framework. Even in its general validity, the design description of the Multimodal Grammar Editor has been carried out using outputs of the unimodal recognizers for speech, gesture, handwriting and sketch, and involving concepts implied by multimodal inputs.

Chapter 7 presents the implementation process that has been followed to develop the Multimodal Grammar Editor. For explaining the software classes implemented in the prototype, the class diagrams of the main packages are presented following the standard Unified Modeling Language (UML) notation. The editor is implemented using the Java language due to its portability in order to maximize the system independence and to make possible to deploy it on the World Wide Web.

Chapter 8 offers some validation of the Multimodal Grammar Editor (MGE), whose theoretical foundations, design and implementation are described in previous chapters. The goals of the validation are mainly twofold. First of all, the workability and usability of the MGE has been assessed for understanding how well it works in practice. Secondly, the evaluation of the grammar inference algorithm has been performed for measuring the correctness of the induced grammar.

Chapter 9 summarizes the contributions of the research of this thesis and outlines some directions for future work.

Chapter 2

Multimodal Interaction

This chapter presents an introduction to multimodal human-computer interaction. The first four sections give an overview of the multimodal interaction process, giving some preliminary definitions that will be used during this dissertation and focusing on conceptual and architectural aspects of multimodal interaction systems. The next section discusses the main kinds of modalities that can convey information from a human user to a computational machine. The last section describes the advantages of multimodal interfaces in terms of accessibility, robustness, stability and expressive power.

2.1 Introduction

The use of the five senses of touch, hearing, sight, smell and taste allows human beings to perceive the external world. A combination of these senses is also used, in all situations of the everyday life, during natural human-human communication. Therefore, communication between human beings is multimodal in nature.

In the last few years this multimodal paradigm has been extensively applied in computer interfaces with the aim of making computer behaviour closer to human communication paradigm. Multimodal human-computer interaction refers to the "interaction with the virtual and physical environment through natural modes of communication" [Bou03]. Multimodal interaction provides the user with a way to interface with a system in both input and output, enabling users to communicate more freely and naturally with automated systems [StS05].

Specifically, in a multimodal system the user communicates with the computer through the simultaneous or alternative use of input/output channels at a time. Such a kind of systems offers a more flexible, efficient and usable environment allowing the user to interact through input modalities, such as speech, handwriting, hand gesture and gaze, and to receive information by the system through output modalities, such as speech synthesis and smart graphics and others modalities, opportunely combined.

Multimodal systems have been largely studied since the 1980s when the first original system "put-that-there" was developed by Bolt [Bol80]. This system used speech and a cursor to point on a touchpad display the location to allow a simple deictic reference, as for example "create a blue square here". Note that a deictic is a word (e.g., "this", "that", "here", "there", etc.) that specifies identity or spatial or temporal location from the perspective of a speaker in the context in which the communication occurs. Deictic expressions are commonly used in multimodal interaction.

As well as the "put-that-there" system, several attempts to overcome common graphical user interface have been made since the 1990s until now [NeS91] [NiC95] [CJM97] [Vo98] [WRB01]. CUBRICON [NeS91] used typed and spoken sentences and deictic mouse clicks as input in order to interact with a two-dimensional map. MATIS (Multimodal Airline Travel Information System) [NiC95] allows the user to ask for information about the departure/arrival time of air flights by using speech and pen-based gesture modalities, along with mouse clicks and keyboarding. QuickSet [CJM97] was developed with the aim of training Californian military troops and used speech and pen-based gestures to interact with a geo-referenced map. QuickTour [Vo98] is a multimodal system that enables a spoken and pen-based interaction to navigate geographical maps. The Smartkom [WRB01] is another multimodal dialogue system that merges gesture, speech and facial expressions for both input and output via an anthropomorphic and affective user interface.

In the next sections an overview of multimodal interaction is given, starting from illustrating some of the main characteristics of multimodal human-computer communication process. Then, a brief description of conceptual features of multimodal interaction and architectural aspects of a multimodal system is presented. Finally, the main interaction modalities used in multimodal systems are introduced and the main advantages of multimodal interaction are discussed.

2.2 The Multimodal Human-Computer Communication Process

The success of the human-computer communication depends on the possibility of sharing a common ground by exchanging information through the communication modalities. Such a communication modality refers to the medium or channel of communication that conveys information [CoC91]. Multimodality refers to the quality of a system to allow more than one communication modality to be used during human-computer interaction.

A general model of multimodal human-computer communication is shown in Figure 2.1. Four different kinds of input/output communication modalities can be identified, according to the study of Schomaker et al. [SNC95]:

- the *human output modalities*, that are devoted to control and manipulate computational systems by achieving a high level of interactivity and naturalness of the multimodal interface. The speech is the dominant modality that carries most of the informational content of a multimodal dialogue. However, gesture and gaze modalities are extensively studied in literature as efficient input modalities that are better suited to represent spatio-temporal information and are usually complementary (that is, their information need to be merged in order to be complete and meaningful) modalities of the speech input;
- the *human input channels*, that are devoted to perceive and acquire information coming from the feedback channels of computational systems. The most frequently used perception channels are eyes, ears and touch, among which the first is the dominant input modality that receives the most information flow, followed by the auditive and tactile channels;

- the *computer input modalities*, through which the computer gets information from the human output modalities. Some examples of devices for computer input modalities are microphone, camera, keyboard, mouse Once acquired, the inputs need to be brought together and interpreted in order to give a coherent meaning to the multimodal act of the user;
- the *computer output channels*, that are devoted to give a feedback to the user, as, for instance, display, loudspeakers, haptic feedback and so on.

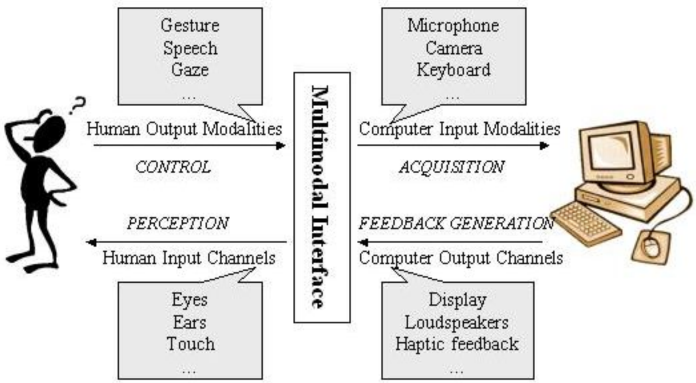


Figure 2.1: The multimodal human-computer communication process

In order to allow that the multimodal human-computer communication process takes place successfully, the actions that the user expresses through the human output modalities have to be acquired by the system through the computer input modalities, and the human input channels of the user have to be able to perceive and understand feedback from the computer output channels.

The informational flow that involves the human output and the computer input modalities is named *input flow*, whereas the flow that involves the human input and computer output channels is named *feedback flow*. The multimodal fusion, that refers to the process of integrating information from various input modalities and combining them into a complete command, takes place during the input flow, while the multimodal fission, which refers to the process of disaggregating information through the various computer output channels, takes place during the feedback flow.

Concerning the human output modalities, six different types of cooperation between these modalities can be identified, as described in the typology proposed by Martin et al. [MGA01]:

- *Equivalence*: several modalities cooperate by equivalence if the same information may be processed as an alternative by either of them;
- *Specialization*: modalities cooperate by specialization if a specific kind of information is always processed by the same modality;
- *Redundancy*: modalities cooperate by redundancy if the same information is processed by these modalities;
- *Complementarity*: several modalities cooperate by complementarity if different information are processed by each modality but have to be merged;
- *Transfer*: modalities cooperate by transfer if information produced by a modality is used by another modality;
- *Concurrency*: several modalities cooperate by concurrency if different information are processed by several modalities at the same time but must not be merged.

In multimodal systems, fusion techniques are mostly applied to complementary and redundant modalities in order to integrate the information provided by them. In particular, complementary modalities provide the system with non-redundant information that have to be merged in order to get a complete and meaningful message. In the same way, redundant modalities require a fusion process that avoids non-meaningful information, increasing, at the same time, the accuracy of the fused message by using one modality to disambiguate information in the other ones.

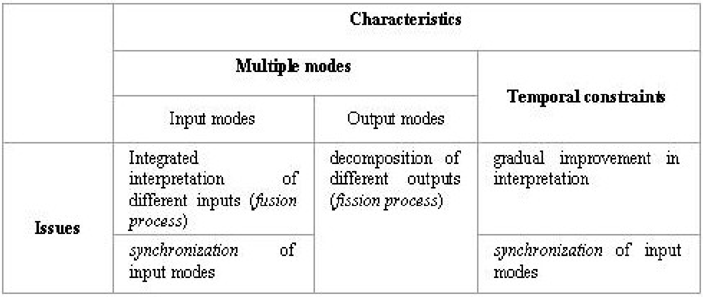
2.3 Conceptual Features of Multimodal Interaction

Characteristics of multimodal interaction relevant to computational modeling of user interfaces and interaction languages include:

- *Multiple modes*: the modalities through which user and system can exchange information are manifold, and include speech, gesture, eye tracking, keyboarding, etc. An interaction modality can be defined [BNB04] as a couple <d, L>, in which d is the physical device and L is the interaction language. Each modality provides a specific piece of information and taken together, they enable the command to be interpreted. Modalities can be classified as active or passive. The former is used when the user must explicitly perform an action with a device to specify a command. The latter is used when an explicit user action is not necessary to specify a command. The information specified by different modalities may be redundant.
- *Temporal constraints*: in a multimodal dialogue there is not a clear, definite instant in which the user finishes formulating the command.

In defining multimodal interaction languages, the input and output modes, temporal constraints and their related issues must be taken into account, as shown in Table 2.1.

Table 2.1: Characteristics of a multimodal interaction and relative issues



	Characteristics		
	Multiple modes		**Temporal constraints**
	Input modes	Output modes	
Issues	Integrated interpretation of different inputs (*fusion process*)	decomposition of different outputs (*fission process*)	gradual improvement in interpretation
	synchronization of input modes		*synchronization* of input modes

To better understand the difficulties in formalizing languages for a multimodal environment, an explanation of these issues is given below.

- Integrated interpretation of different inputs (fusion process). As a multimodal dialog involves the simultaneous use of multiple modalities, the user's input/commands must be interpreted through a fusion process. This integrates information from various input modalities by removing redundant or complementary information across the modalities and combining them into a complete command.

- Synchronization of input modes. Timing is essential in conveying information during a multimodal interaction, so a tight synchrony among the various communicative modalities is required. This means that user inputs must be synchronized to deliver the correct information at the right time.

- Decomposition of different outputs (fission process). The system has to find ways to integrate output through the various channels in order to provide the user with consistent feedback. This process is called fission, in contrast with multimodal fusion.

- Gradual improvement in interpretation. The system must interpret the input while the interaction is ongoing and refine the interpretation when a new multimodal action is performed by the user.

Many works have focused on the development of a multimodal dialogue system that considers all the interaction features and issues described above. Gupta [Gup03] outlines a method to collect input information supplied in different modalities, to determine when the user has finished providing input, to fuse the collected information to create a joint interpretation using an unification algorithm, and to send the joint interpretation to a dialogue manager that can perform reasoning. This method also considers temporal relationships between the modalities used during the interaction. Another comprehensive exploratory analysis of multimodal integration and synchronization patterns during pen-voice human-computer interaction is conducted by Oviatt et al. [ODK97].

2.4 Architectural Features of Multimodal Systems

Having looked at conceptual aspects of multimodal communication, some remarks about the architectural features of a multimodal system are given in this section.

A common architecture of a multimodal system [Ovi02], that involves speech, sketch and handwriting modalities, is depicted in Figure 2.2. During the acquisition phase, the input that the user expresses through these human output modalities is acquired through the appropriate computer input channels (touch-pad for sketch and handwriting, and microphone for speech) and processed by the related recognition modules (sketch and handwriting recognition and Natural Language Processing (NLP), respectively) in the subsequent recognition phase. Afterwards, the multimodal fusion system carries out the integration of the recognized inputs, by removing possible redundancy, merging complementary information from each modality and synchronizing the information in order to produce a meaningful and correct input. At this point, the dialogue management system aims at processing the integrated multimodal message/command by activating appropriate applications and service in order to retrieve the output to be returned to the user (decision phase).

The mapping between the input message expressed by the user and the corresponding output returned by the system is defined input interpretation. Thus the interpretation process involves, generally, four phases, corresponding to the main architectural levels of a multimodal system, from the top to the bottom (see Figure 2.2): the acquisition, recognition, integration and decision phases (levels). Although the acquisition, recognition and decision are consecutive phases, the same does not occur for the integration phase (where the fusion process takes place), because in some systems the integration phase is prior to the recognition or decision phases, whereas in other systems it is just the opposite.

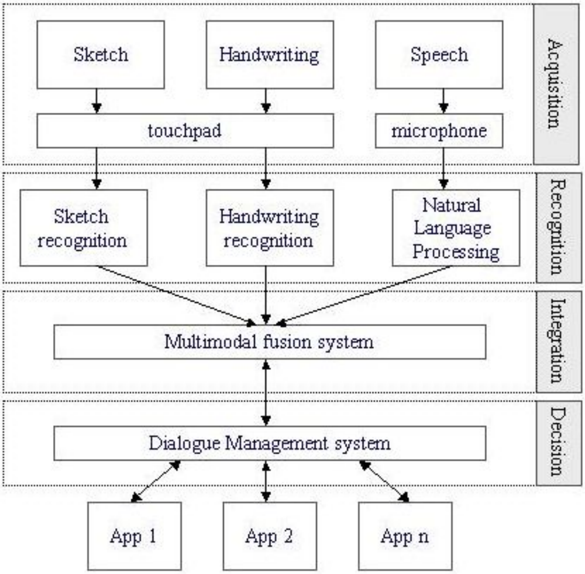


Figure 2.2: A common architecture of a multimodal system

2.5 Human-Computer Interaction Modalities

In this section the human output modalities will be illustrated starting from the speech, which is the prevalent modality that carries most of the informational content of a multimodal dialogue and concluding with gesture, drawing and other less conventional modalities.

2.5.1 Speech

There is no doubt that speech is the most spontaneous modality through which a user can communicate with computational machines. Therefore, speech recognition is of primary importance for the development of multimodal systems.

Speech recognition consists in translating from a data stream into a sequence of meaningful words that need to be interpreted by the recognizer. Although in the last few years several efforts have been made and impressive achievements have been carried out, the issue of automatic speech recognition is not at all solved. The difficulties to correctly recognize input data are mainly due to:

- the use of a spontaneous language: speech input may contain any noise, hesitation and other prosodic behaviors as well as meaningful information. These phenomena characterize a speaker that talks spontaneously. As a multimodal system relies on a flexible, natural and spontaneous interaction, it is necessary to consider these phenomena during speech recognition.
- the number of words in the vocabulary: more is the amount of words in the vocabulary used by the recognizer, more is the accuracy of the recognizer. At the same time, however, the probability of an incorrectly recognition for a word increases and the processing time gets longer.
- speaker-dependency: the automatic speech recognition has to be independent from the speaker as more as possible. However, a speaker-dependent recognizer is more efficient than a speaker-independent one.
- environmental factors: noise due to the environment around the speaker and the overlapping of simultaneous dialogues can make the performance of recognition worse.

Generally, speech recognition follows a grammar-based approach, which is able to define a set of acceptable sentences. Speech recognition systems that are based on grammar (both regular and context-free) provide the best performance in terms of simplicity and efficiency.

2.5.2 Handwriting and Gesture

Handwriting can be classified as a pen-based input modality as, generally, the user interacts through the use of a pen on a touch-sensitive screen. Similarly, a particular kind of two-dimensional gesture, named pointing gesture, makes use of the finger, instead of a pen, for indicating an object on the screen.

Handwriting and pointing gesture recognition has been extensively studied in literature.

Concerning handwriting [TSW90], two different approaches to input processing exist: off-line, in which the handwritten words are captured by a static picture, and on-line, in which the system dynamically acquires the strokes, where a stroke is a pen down, pen movement, pen up sequence on a touch-sensitive screen. The difficulties in handwriting recognition are similar to those in automatic speech recognition: the presence of noise, the size of vocabulary, the writer-dependency. Other difficulties may arise from:

- character ambiguity: some characters are similar and consequently they may be not correctly recognized, as for example the number zero and a capital ‘O’. In this case the context is very useful for resolving the ambiguity.
- the kind of alphabet: the Italian alphabet is composed of twenty-one characters, each one can be generally represented by one or two strokes. Chinese alphabet is composed of 50,000 characters; each one is represented by eight/ten strokes on the average.
- spacing of characters: in the Italian italic text there is no space among the characters. On the contrary, in a Chinese written text characters of the same word are spaced.

Pen-based gesture recognition is quite similar to handwriting, except for the arbitrariness of the alphabet. According to the taxonomy of gesture modalities define by Blattner et al. [BlM95], three different categories of gesture can be used in a multimodal interface:

- Arbitrary gesture: these gestures can be interpreted without a preliminary training of the system. Generally, they are called non-transparent gestures because are not immediately derived.
- Mimetic gesture: this kind of gestures can be interpreted by the system at a glance, as for example the iconic gesture. For this reason they are named also transparent gesture.
- Deictic (or pointing) gesture: this kind of gesture is used only with reference to the situation in which it is expressed. The following three kinds of deictic gestures exist:

- Specific, if they refer to a precise object;
- Generic, if they refer to the whole class of objects;
- Mimetic, when the reference is followed by an additional movement.

2.5.3 Other Modalities

In the literature several other less conventional modalities have been explored besides speech and pen-based gestures. A brief overview of to this kind of modalities is given below.

Lip-reading is a method used for increasing the efficiency of speech recognition algorithms. In fact, the analysis of lip movement allows to synchronize the visual information source with the audio stream and permits to easily distinguish acoustically confusable speech words.

Another modality used in multimodal systems is three-dimensional gesture, which is more complete than two-dimensional ones, even if more complex to process. In 2D gesture a digital pen is used, while 3D gesture are captured using a glove, a camera, or some kinds of sensors.

Eye movement is another useful source of information that allows to identify what is the referred object while a user is performing a task.

2.6 Advantages and Critical Elements of Multimodal Interaction

Although multimodal applications are considerably more complex than traditional unimodal ones, they are characterized by several advantages that will be shown in this section. The use of multimodality has benefits for the user in terms of naturalness of interaction, accessibility and expressive power. A critical element, even for a well-designed multimodal system is its robustness and stability.

2.6.1 Naturalness and Accessibility

Humans in the everyday life use their body and five senses to communicate each other; for this reason imitating face to face interaction between humans involving all senses makes the interaction process very natural and intuitive. For this reason multimodal interfaces, if correctly designed, appear to be intrinsically natural. Naturalness improves the easy use of interfaces and, consequently, the accessibility to information, services and more generally to resources.

The term accessibility refers to the ability of a device, service or resource of being easily accessible by a large number of different users in various contexts. The accessibility is one of the most relevant features of a multimodal system. Since each person can have her/his own preferences and communication ability, multimodal interfaces join user needs by allowing users of whichever age, intellectual ability, skill, physical or sensorial disability, language, etc., to access the computational systems.

Therefore, a multimodal system that has to be flexible must allow not only to combine data coming from different sources, but also to let the user to choose the preferred interaction way according to his/her task and context.

Research studies [CBB94] [CoO91] have proved that users prefer spoken language for describing objects or situations, while generally they use pen-based modalities for communicating numbers or graphic topics and for pointing and highlighting.

Moreover, multimodal systems that enable speech, pen-based gesture and handwriting ease the exploration of new hardware and software technologies, mainly in mobile field.

2.6.2 Robustness and Stability

Robustness is one of the features of a computational system, concerning the suitable management of unexpected situations, such as errors or wrong uses. It is a critical aspect in particular for complex systems such as a multimodal one. Indeed, due to the complexity of the interaction process the multimodal interface design could improve the criticism of the communication process or, on the contrary a good design could produce a major aptitude for handling errors than unimodal one. In fact, it could allow to:

- select the input modality that is less error prone in a given context, by avoiding, in such a way, to introduce possible mistakes;

- use several modalities by reducing complexity of natural language processing and consequently errors due to the recognition process;
- change from a modality to another one, making easier to correct possible errors due to the interpretation process.

Moreover, a multimodal interface that is correctly designed enables a mutual disambiguation of input modalities. Semantic information coming from each modality may be used as (partial) support for clarifying the semantic meaning of the other modalities. Therefore, this mutual disambiguation property makes a multimodal interface more robust.

2.6.3 Expressive Power and Efficiency

Computational systems that interpret inputs from different modalities aim at achieving a powerful interface able to acquire and manage these inputs. Interfaces that rely only on keyboard and mouse modalities, like traditional computational systems, are inadequate for interacting with mobile devices and last-generation technologies. Moreover, multimodal interaction allows users with temporary disabilities (i.e. people that are not able to express an input through a specific modality for a limited time) to use the system. For instance, when the user is driving he/she can interact with mobile devices through the keyboard, but he/she could use speech input.

As speech and pen-based modalities are the prevalent communication channels used in face-to-face interaction, they represent an easy and useful way to express objects' descriptions, constraints, and spatial relationships to a multimodal system. For instance, a user could mark an area on a map by drawing a circle and, at the same time, express the name of an element related to the area by voice. Whether the user had to express the same information by one modality only, for instance by voice, he/she would have more difficulties. Moreover, the user is free to distribute various parts of the message to different modalities to ease (complex) communication and to reduce cognitive loading [Ovi04]. For these reasons, the use of a multimodal interface increases the efficiency and offers expressive power to the language at user's disposal.

Chapter 3

Multimodal Fusion and Grammars

In this chapter an overview of research related to multimodal fusion strategies is presented. The first part analyses the existing approaches to multimodal fusion, classifying them according to the data fusion level (e.g. the fusion process takes place at recognition, decision or in both levels). The last section focuses on the grammar-based fusion approach providing a critical survey of the literature on multimodal grammars approaches, as the multimodal language processor described in this thesis uses this kind of approach.

3.1 Introduction

The integration of multiple interaction modalities in multimodal systems is a fundamental process that has been largely studied in the literature giving rise to a wide variety of multimodal fusion approaches.

Analyzing the human-human communication process, it becomes obvious that the interaction between multiple modalities can occur at different levels of their production. For instance, considering speech and pointing gestures, information conveyed by these two modalities are not only referred to the same mental concept of the speaker but they are also generated by the same lower level mental process of the speaker, as suggested by some studies on human communicative behavior [LeM92]. Analogously, considering the perspective of sensor data fusion, different levels of data integration can be identified. Consequently, the integration of multimodal features at different levels of analysis becomes obvious also in multimodal systems. In Section 3.2 a survey of existing multimodal fusion approaches, classified according to the level at which the fusion takes place, is presented.

This dissertation follows the approach that integrates multiple input modes with the use of a multimodal grammar. This choice is due to the ability of the grammar-based paradigm to meet the requirements of naturalness and flexibility needed for achieving an efficient multimodal interaction. In contrast to the multimodal grammars existing in the literature, of which a brief description is provided in Section 3.3, the grammatical approach proposed in this thesis relies on an attribute grammar that is able to handle an arbitrary number of modalities as well as temporal information into the grammar, and provides explicit constructions for modeling semantic aspects of the language. Moreover, a “by example” paradigm has been followed, which allows the end user to provide concrete examples of multimodal sentences that have to be recognized, and the system automatically generates the grammar rules to parse those examples. All these choices are justified in the discussion, provided in Section 3.3.6.

3.2 Data Fusion Levels in Multimodal Fusion

The input signals, expressed by the user through the human output modalities and acquired by the system through the computer input modalities, can be combined at several different levels [SPH98]. As introduced in the previous chapter (see Section 2.4), a multimodal system is composed of four main architectural levels (acquisition, recognition, integration and decision). The integration level, in which the fusion of the input signals is performed, may be placed: (i) immediately after the acquisition level and we refer to the fusion at acquisition, or signal, level; (ii) immediately after the recognition level and in this case we refer to the fusion at recognition, or feature, level; (iii) during the decision level and we refer to the fusion at decision, or conceptual, level.

The *fusion at acquisition level* (see Figure 3.1.a) generally consists in mixing two or more, electrical signals. As this kind of fusion may be performed if the signals are synchronized and of the same nature (two speech inputs, two sketch inputs, etc.) it cannot be applied to multimodal inputs, which usually are of different nature. Consequently, this level of fusion is not taken into account hereafter.

The *fusion at recognition level* (named also *early fusion* or *recognition/feature-based fusion*) consists in merging the outcomes of each recognizer by using integration mechanisms, such as, for example, statistical integration techniques, agent theory, hidden Markov models, artificial neural networks, etc. The integrated sentence is therefore processed by the decision manager that provides the most probable interpretation of the sentence (see Figure 3.1.b). Thus a unimodal recognition stage and an integrated decision stage characterize the interpretation process of the early fusion. This strategy is generally preferred for closely and synchronized inputs that convey the same information (redundant modalities), as for example speech and lip movements for speech recognition or voice and video features for emotion recognition. The main drawbacks of the early fusion are the necessity of a large amount of data for the training, and the high computational costs.

The *fusion at decision level* (named also *late fusion* or *decision/conceptual-based fusion*) means merging neither the signals nor the features of each recognized input, but directly the semantic information that are extracted from the specific decision managers (see Figure 3.1.c). In fact, in this kind of fusion the outcomes of each recognizer are separately interpreted by the decision managers and the extracted semantic meanings are integrated by using specific dialogue-driven fusion procedures to yield the complete interpretation. Late fusion is mostly suitable for modalities that differ both in their nature and in the time scale. This implies that a tight synchrony among the various communicative modalities is essential to deliver the correct information at the right time. As each input modality is separately recognized and interpreted, the main advantages of this kind of fusion rely on the use of standard and well-tested recognizers and interpreters for each modality, as well as the greater simplicity of the fusion algorithms.

In addition to these three levels of multimodal fusion, a fourth level, named *hybrid multi-level fusion*, can be identified (as described also in [Vo98]). In this kind of fusion the integration of input modalities is distributed among the acquisition, the recognition and decision levels. In particular, the interdependence among modalities, that allows predicting subsequent symbols knowing previous symbols in the input data flow, is exploited to improve accuracy of the interpretation process. This implies that a joint multimodal language model, which relies on the symbols acquired during the acquisition phase and which is governed by their semantic meanings extracted during the decision phase, is the basis of the hybrid multi-level fusion strategy.

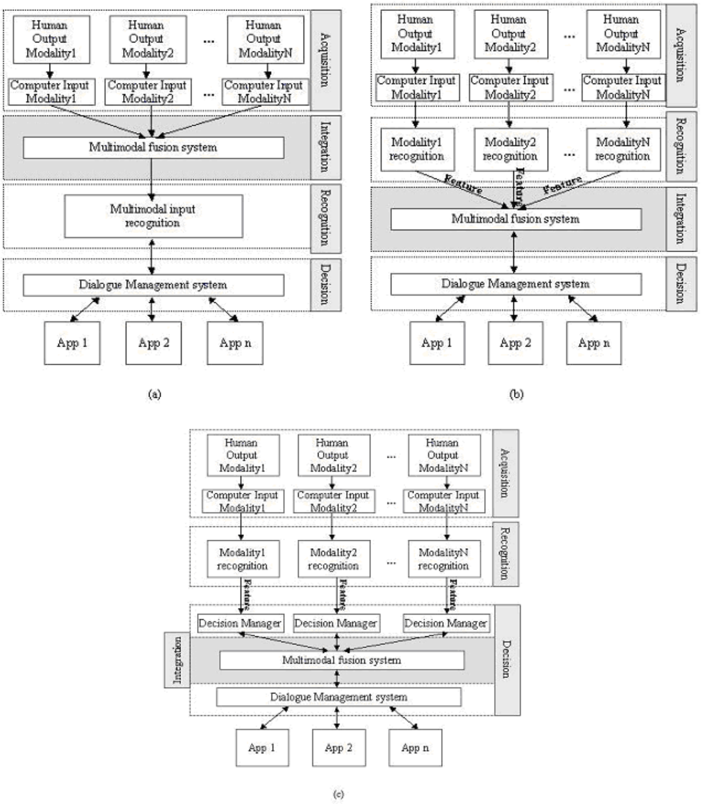


Figure 3.1: Possible levels of multimodal data fusion: a) fusion at signal level; b) fusion at recognition level; c) fusion at decision level

To sum up, depending on the data fusion level at which the different inputs are combined, multimodal fusion strategies can be broadly classified as: recognition-based, decision-based and hybrid multi-level strategies.

3.2.1 Recognition-based Fusion Strategies

To achieve the integration of input signals at recognition level, multimodal systems have to rely on appropriate structures to represent these signals. In particular, three main representations can be found in literature, namely: action frame [Vo98], input vectors [PBH97] and slots [APS98].

In the approach based on action frame, proposed by Vo [Vo98], the multimodal input is regarded as a set of parallel information streams. Each stream represents one unimodal input coming from a computer input modality (e.g. a sequence of words and phrases in spoken modality, shapes in gestures, etc.) and consists of elements associated to a set of parameters. The integration of unimodal inputs consists in producing a sequence of input segments, named parameter slot, which separately contribute to the multimodal input interpretation, called action frame. Such an action frame specifies the action that has to be performed by the system in response to the multimodal input. Each parameter slot specifies one action parameter and should contain enough information to determine the value of the corresponding parameter. The integration of the information streams is carried out through the training of a Multi-State Mutual Information Network (MS-MIN). More in detail, this network allows to find an input segmentation and a corresponding parameter slot assignment in order to extract the actual action parameters from the multimodal input. To achieve that the a posteriori probability of the parameter slot assignment conditional on the input segmentation is introduced. This probability is estimated by output activations in the MS-MIN network and can be interpreted as the score of a path that goes through the segmented parameter slots. An example of path over two multidimensional inputs (the spoken words “How far is it from here to there?” and the drawing of an arrow between two points) is shown in Figure 3.2.

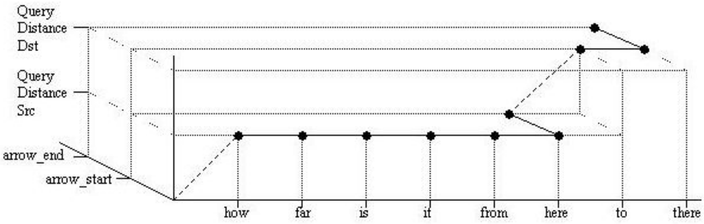


Figure 3.2: The output path of the MS-MIN of Vo [Vo98]

Therefore, a path score maximization algorithm is applied to find the input segmentation and the corresponding parameter slot assignment. This algorithm creates an extra layer on the top of the network. In particular, each output unit of the MS-MIN is an output state and the top layer of the network produces the best states sequence that fits the input, according to the path score maximization algorithm. The main advantage of this approach relies on the use of the MS-MIN network that allows the incremental and automatic learning of the mapping from input messages to output actions and the consequent improvement of the interpretation accuracy during the real use.

The input vectors proposed by Pavlovic et al. [PBH97] are used to store the outputs of the visual and auditory interpretation modules. More in detail, the visual module firstly tracks the features of the video data by using skin colour region segmentation and motion-based region tracking algorithms and the time series of the tracked features is stored into an input vector. Secondly, these features are dynamically classified by using Probabilistic Independence Networks (PINs) and Hidden Markov Models (HMMs). Therefore, the output of this module consists in a set of higher level features ranged from gestural movement elements, called visemes (e.g. “left movement”), to full gestural words (e.g. symbol for “rotate about x-axis). The auditory module has the same architecture and functions of the visual module applied to audio data. A HMM PIN allows to classify the auditory features into auditory elements, called phones, and full spoken words. The integration of the two interaction modalities is carried out through a set of HMM PIN structures (see Figure 3.3), each corresponding to a predefined audio/visual command. The state of each HMM is defined according to the input vectors containing the high level features coming from the auditory and visual modules. As the multimodal integration occurs after a two-stage recognition process (for audio and visual data, distinctly) and before the interpretation of the joint features has been performed, the fusion approach of Pavlovic et al., similarly to the action frame approach, can be classified as a recognition-based fusion strategy.

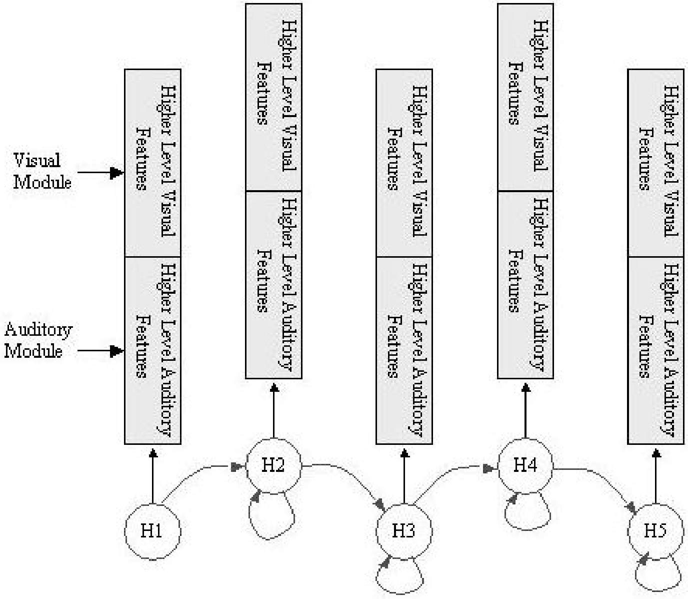


Figure 3.3: The multimodal integration approach of Pavlovic et al. [PBH97]

In the strategy based on slots [APS98], the information inputted by the user is stored into a slot buffer, which allows back referencing of past lexical units (e.g.: “it” can be used to reference the previously selected object). The command language of the application is encoded in semantic units called frames. The command frames are composed of slots, i.e. lexical units provided by the multimodal input. For instance, considering the “move frame” two slots can be identified: “object” (to specify the object) and “where” (to specify the final position). The frames are predefined (computed off line) and are application-dependent. The parser extracts the lexical units from different input modalities and fills the appropriate slots in the slot buffer. The slot buffer is continuously monitored checking for filled frames. Once a frame is filled (enough information to generate a command), the fusion agent sends it to be executed in the current application. The main advantage of this architecture is the uniform access of the input modes.

In all the three fusion strategies, described above, the input signals are merged after recognizers have transformed them into a more appropriate representation (action frames, input vectors, and slots) but before any interpretation has been assigned to the unimodal input. This has led us to classify them as recognition-based fusion strategies.

The main advantage of these strategies relies on the great coherence with the human-human communication paradigm in which the dialogue is considered as a unique and multimodal communication act. Analogously, the recognition-based fusion strategies merge the recognized inputs into a unique multimodal sentence that has to be opportunely interpreted. Moreover, they allow an easier inter-modality disambiguation. The main drawbacks of the recognition-based fusion strategies consist in the significant computational load and the high dependency on time measures. This dependency implies as well a large amount of real data to train the network (both the MS-MIN and the PIN HMM).

3.2.2 Decision-based Fusion Strategies

In the decision-based approach, the outcomes of each recognizer are separately interpreted by specific decision managers and then sent to the dialogue management system that performs their integration by using specific dialogue-driven fusion procedures to yield the complete interpretation. To represent the partial interpretations coming from the decision managers and achieve the integration of input signals at decision level, past and actual multimodal systems employ several kinds of structures, namely: typed feature structures [CJM97] [Joh98], melting pots [NiC95] [BNG04], semantic frames [VoW96] [RSH05], and time-stamped lattices [CMB03].

The typed feature structures, originally proposed by Carpenter [Car92], are used by Cohen et al. [CJM97] to represent the semantic contributions of the different input modalities. In particular, this data structure consists of two main elements: the type that specifies the class which the input to be represented belongs to, and a collection of feature-value pairs, in which the values can be atoms or another feature structure. An example of typed feature structure representing the syntactic features of a proper noun is shown in Figure 3.4.a. Feature structures and atoms are assigned to hierarchically ordered types. The authors achieve the integration of spoken and gestural inputs through a unification operation over these typed feature structures. Such operation requires pairs of feature structures or pairs of atoms that are compatible in type and the result of the unification is the most specific feature structure or atom in the type hierarchy. Figure 3.4.c shows the unification of the two feature structures represented in Figures 3.4.a and 3.4.b, which is the syntactic features of the word “dog”. To select the best-unified interpretation among the alternative solutions probabilities are associated with each unimodal input. This decision-based fusion strategy is implemented in QuickSet [CJM97], a multimodal system briefly described in Section 2.1.

$$\begin{bmatrix} & \text{cat : LING - SIGN} \\ \text{content :} & \begin{bmatrix} \text{sintax :} & \begin{bmatrix} \text{cat : N} \\ \text{N\_type : proper} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(a)

$$B = \begin{bmatrix} & \text{cat : LING - SIGN} \\ \text{content :} & \begin{bmatrix} \text{phon : /DOG/} \end{bmatrix} \end{bmatrix}$$

(b)

$$A \cup B = C = \begin{bmatrix} & \text{cat : LING - SIGN} \\ \text{content :} & \begin{bmatrix} & \text{phon : /DOG/} \\ \text{sintax :} & \begin{bmatrix} \text{cat : N} \\ \text{N\_type : proper} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

(c)

Figure 3.4: An example of typed feature structures unification

Johnston [Joh98] carries on the study of Cohen et al. [CJM97] introducing a grammar representation in which spoken sentences and pen gestures are the terminal elements of the grammar, referred to as lexical edges. Each lexical edge is assigned grammatical representations in the form of typed feature structures. For instance, to represent the spoken word ‘helicopter’ the feature structure in Figure 3.5 is created, where the cat feature indicates the basic category of the element, the content feature specifies the semantic content, and the remaining features represent the modality, the temporal interval and the probability associated with the edge. Multimodal grammar rules are encoded as feature structure rule schemata that can be hierarchically ordered allowing the inheritance of basic constraints from general rule schemata. The application of these rules enables the unification of two candidate edges and the consequent fusion of the corresponding multimodal elements.

Although these two approaches, based on typed feature structures, provide a generally applicable representation for the different modalities and the exploitation of well-known grammar-based techniques extensively explored in natural language processing, they show significant limitations on the expressivity and complexity.

$$\begin{bmatrix} cat : create\_unit \\ content : \begin{bmatrix} fsTYPE : create\_unit \\ object : \begin{bmatrix} fsTYPE : unit \\ type : car \\ group : vehicle \end{bmatrix} \\ location : [fsTYPE : point] \end{bmatrix} \\ modality : speech \\ time : interval(\ldots\ ,\ \ldots) \\ prob : 0.85 \end{bmatrix}$$

Figure 3.5: An example of representation of a spoken word by typed feature structure

The fusion strategy based on melting pots, proposed by Nigay and Coutaz [NiC95], was originally implemented within the

MATIS multimodal system. As shown in Figure 3.6, a melting pot is a 2-D structure, in which the vertical axis contains the “structural parts”, i.e. the task objects generated by the input actions of the user, and the horizontal axis is the time. The fusion is performed within the dialogue manager by using a technique based on agents (PAC-Amodeus agents). Three criteria are used to trigger the fusion of melting pots. The first criterion, referred to as microtemporal fusion, is used to combine information produced either in parallel or over overlapping time intervals. The second criterion, called macrotemporal fusion, takes care of either sequential inputs or time intervals that do not overlap but belong to the same temporal window. A further criterion, referred to as contextual fusion, serves to combine input according to contextual constraints without attention to temporal constraints.

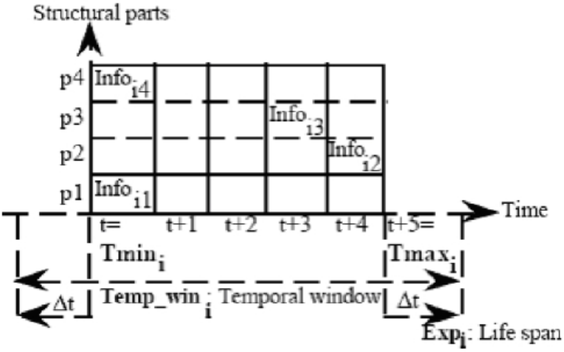


Figure 3.6: The structure of a melting pot [NiC95]

A refinement of the approach of Nigay and Coutaz [NiC95] has been carried out by Bouchet et al. [BNG04] and implemented in the ICARE (Interaction CARE - Complementarity Assignment, Redundancy and Equivalence) framework. Such framework considers both pure modalities, described through elementary components, and combined modalities, specified through composition components. Two kinds of elementary components are defined: the device components that abstract the captured input signals into recognized inputs, and the interaction language components that abstract the recognized inputs coming from the device components into commands. Finally, the composition components describe the fusion criteria of data provided by the elementary components, in line with the criteria defined in [NiC95]. The main advantage of the ICARE approach relies on the component-based structure that allows to reduce production costs ensuring a high reusability and maintainability.

In the approach based on semantic frames, proposed by Vo and Wood [VoW96], input from each modality is parsed and transformed into a semantic frame containing slots that specify command parameters. The information in these partial frames may be incomplete or ambiguous if not all elements of the command were expressed in a single modality. A domain independent frame-merging algorithm combines the partial frames into a complete frame by selecting slot values from the partial frames to maximize a combined score. This approach is quite similar to the melting-pot strategy described above.

The use of semantic frames with slots is followed also by Russ et al. [RSH05]. As opposed to the previous fusion mechanism, in the approach of Russ et al. each slot (called main slot) contains also the connections to a semantic network, as well as the attributes associated to each recognized input (contained into the attribute slots), as shown in Figure 3.7. A node in the network consists of a term and an activation value. If a connected node of the semantic network is activated, the slots of the frames are filled with the attributes as well as the activation values of the nodes. Therefore, the overall activation of a frame corresponds to the probability that the user input correlates with the frame. As each input can have multiple interpretations, this probability is taken into account to evaluate the best candidate interpretation. The main advantage of this approach is the uselessness of knowing a predetermined language or specific commands.

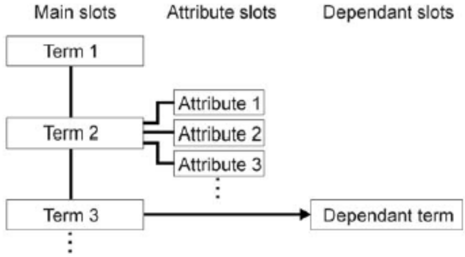


Figure 3.7: The structure of the semantic frame of Russ et al. [RSH05]

In the approach based on time-stamped lattices, proposed by Corradini et al. [CMB03], each recognizer produces a set of candidate interpretations where each one stands for an independent and diverse interpretation of the input signal. They are encoded by means of word lattices where several paths through the word lattice reflect the individual interpretations or n-best lists. The fusion engine combines the time-stamped lattices received from the recognizers, selects its multimodal interpretation, and passes it on to the dialogue manager. The selection of the most probable interpretation is carried out by the dialogue manager that rules out inconsistent information by both binding the semantic attributes of different modalities and using environment content to disambiguate information from the single modalities.

All the approaches introduced above occur at decision level, since individual input coming from the specific recognizers are partially interpreted before their integration.

The main advantage of these strategies is the multi-tasking, as different multimodal channels, recognizers and interpreters are arranged for carrying out independent unimodal input processing at the same time. This implies also the possibility to use standard and well-tested recognizers and interpreters for each modality. On the other hand, decision-based fusion strategies are characterized by a high complexity of the inter-modality disambiguation, particularly when dealing with more complex modalities that need not only pairs item-time but full lattices from each channel to disambiguate the multimodal input.

3.2.3 Hybrid Multi-level Fusion Strategies

In the hybrid multi-level approach, the integration of input signals is distributed among the acquisition, the recognition and decision levels. To parse multiple input streams and to combine their content into a single semantic representation three main methodologies have been applied in literature: finite-state transducers [JoB00], multimodal grammars [SCS06] and dialogue moves [PAM05].

The approach based on finite-state transducers was proposed by Johnston et al. [JoB00]. The authors perform multimodal parsing and understanding by using weighted finite-state transducers (FSTs) running on three tapes, in which the first tape represents the speech stream (words), the second the gesture stream (gesture symbols), and the third their combined meaning (meaning symbols). The transitions of the FST, which consist of an input and output symbol, are traversed if the input symbol matches the current recognized symbol and consequently it generates the corresponding output symbol. Figure 3.8 shows an example of transducer relating the spoken input "Email this person and that organization" and the gesture with the pen on the appropriate person and organization on the screen. Modalities integration is carried out by merging and encoding into a FST both semantic and syntactic content from multiple streams. In this way, the structure and the interpretation of multimodal utterances by using FST is roughly equivalent to a context-free multimodal grammar that parses the inputs and yields the output tape providing semantic information.

The FST approach is very versatile and provides a high degree of flexibility, allowing a huge spectrum of multimodal commands to be implemented. On the other hand, this approach does not support mutual disambiguation, i.e., using information from a recognized input to enable the processing of any other modality. Moreover, a huge amount of data is required to train the FST limiting portability.

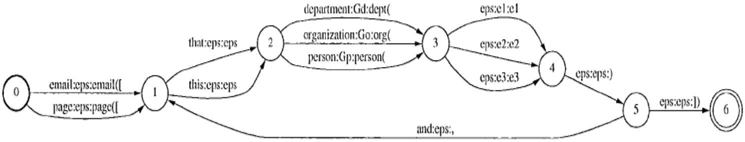


Figure 3.8: A finite-state transducer in the approach of Johnston et al. [JoB00]

In the approach based on multimodal grammars, the outcomes of each recognizer are considered as terminal symbols of a formal grammar and consequently they are recognized by the parser as a unique multimodal sentence. Therefore, in the interpretation phase the parser uses the grammar specification (production rules) to interpret the sentence. This fusion strategy has been implemented in the MUMIF system [SCS06]. The fusion module of MUMIF applies a multimodal grammar to unify the recognized unimodal inputs into a unique multimodal input that is represented by using the TFS (Typed Feature Structures) structure proposed by Carpenter [Car92]. The MUMIF multimodal grammar is composed of two kinds of rules: lexical rules that are used to specify the TFS representation and grammar rules that constrain the unification of inputs.

The dialogue moves are used by Perez et al. [PAM05] to represent multimodal user inputs coming from the lexical-syntactical analyzer. This structure, originally proposed by Quesada et al. [QTG00], consists of a feature-value structure with four main features, which are DMOVE, TYPE, ARG and CONT (DTAC). An example of DTAC for the command “Turn on the kitchen light” is shown in Figure 3.9. The DTAC is quite similar to the typed feature structure of Carpenter [Car92]. This approach is implemented in the Delfos system, consisting of Multimodal input pool, a Natural Language Understanding (NLU) module and a collaborative dialogue manager. The multimodal input pool receives and stores all inputs (each one considered as an independent dialogue move) including information such as time and modality. The NLU module parses the input and adds further features in the DTAC structure, such as the modality of the event, the time at which the event started and ended. The dialogue manager checks the input pool regularly to retrieve the corresponding input. It operates by means of update unification rules, which define the constraints on the integration of DTAC structures. If more than one input is received during a certain time frame, further analysis is performed in order to determine whether those independent multimodal inputs are truly related or not.

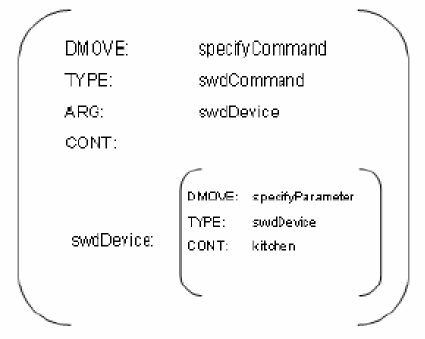


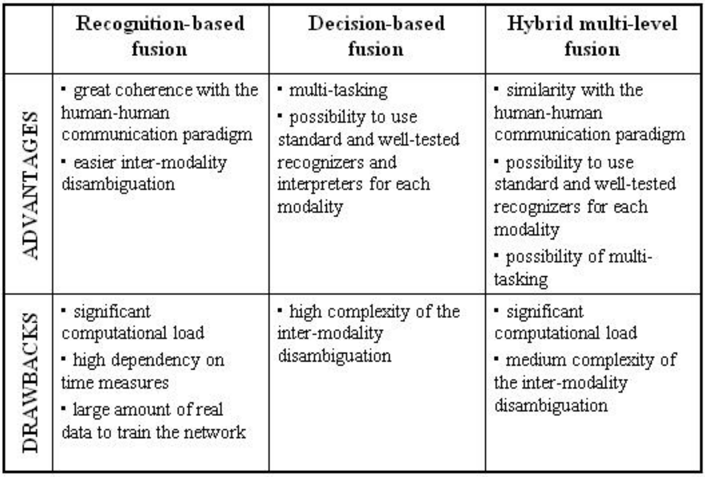
Figure 3.9: An example of dialogue move in the approach of Perez et al. [PAM05]

The main advantage of the hybrid multi-level fusion strategies relies on the similarity with the paradigm used in the human-human communication, in which the dialogue is considered as a unique linguistic phenomenon. On the other hand, these strategies are characterized by a high complexity of the inter-modality disambiguation.

3.2.4 Final Discussion on Multimodal Fusion Approaches

As discussed in the previous sub-paragraphs, each multimodal fusion strategy has some advantages and drawbacks, which can be summarized as shown in Table 3.1.

Table 3.1: Advantages and drawbacks of multimodal fusion strategies



	Recognition-based fusion	Decision-based fusion	Hybrid multi-level fusion
ADVANTAGES	▪ great coherence with the human-human communication paradigm ▪ easier inter-modality disambiguation	▪ multi-tasking ▪ possibility to use standard and well-tested recognizers and interpreters for each modality	▪ similarity with the human-human communication paradigm ▪ possibility to use standard and well-tested recognizers for each modality ▪ possibility of multi-tasking
DRAWBACKS	▪ significant computational load ▪ high dependency on time measures ▪ large amount of real data to train the network	▪ high complexity of the inter-modality disambiguation	▪ significant computational load ▪ medium complexity of the inter-modality disambiguation

The multimodal integration approach proposed in this thesis follows the hybrid multi-level fusion paradigm, as it joins together the advantages of recognition-based and decision-based fusion strategies and, at the same time, it is able to overcome most of their drawbacks, with the exception of the high computational load.

In particular, a grammar-based approach has been adopted as it is the most coherent with the human-human communication paradigm and it meets the requirements of naturalness and flexibility for an efficient multimodal interaction.

The next section surveys the literature on grammars used in multimodal fusion, providing a critical comparison of them.

3.3 Grammars for Multimodal Fusion

The first works on grammars for multi-dimensional languages (i.e. languages whose expressions are assembled in more than one dimension) were addressed to define two-dimensional graphical expressions. Some examples of grammatical framework for visual languages were given by Constraint Multiset Grammars [HMO91] and Relation Grammars [CGN90]. The main difficulty of these grammars, that precludes their reuse in multimodal language definition, consists in the significant computational complexity. In fact, a study of Wittenburg et al. [WWT91] showed that an exponential number of combinations of visual input elements needs to be considered, at worst, giving rise to a high complexity of the parsing too.

In multimodal expressions the number of elements to be parsed is generally far smaller than in complex graphical expressions. Some studies of Oviatt et al. [Ovi96] [ODK97] showed that a multimodal utterance generally does not contain more than three elements. Consequently, the number of potential combinations of these elements remains sufficiently small to enable a fast processing, which is not achievable through the aforementioned visual grammars.

A more promising approach, put forward by several researchers, consists in starting from techniques used in Natural Language Processing (NLP) (see [JCM97]) and extending them to Multimodal Language Processing (MLP). As traditional grammars for natural language (NL) (that is, the kind of language used by human beings) are not powerful enough to cope with the syntactic structure of multimodal languages, an evolution of NL grammars towards multimodal grammar is necessary.

This section only presents a review of those grammatical theories that have been developed in natural language processing and subsequently extended and adapted for multimodal input processing. I firstly provide a short overview of the most traditional grammar formulation: the context-free grammar. Furthermore, a brief description and a critical analysis of formal grammatical approaches extensively used in NLP are provided in Sections 3.3.2, 3.3.3, 3.3.4, and 3.3.5, in conjunction with an explanation of how these grammars have been adapted to multimodal inputs.

Grammar descriptions are brief as they are intended only to provide a background for the discussion in Section 3.3.6, which aims at justifying the theoretical choice of the approach proposed in this thesis.

3.3.1 Context-Free Grammars

The most popular kind of grammar that has been firstly used to define the syntax of natural language is the context-free grammar

(CFGs), defined by Chomsky in the mid-1950s [Cho57]. A grammar is termed context-free when the expansion of a symbol does not depend on its context (i.e., the position of the symbol in a sequence or the relationship with surrounding symbols). A context-free grammar consists of four components:

T, is a finite set of terminal symbols;

N, is a finite set of non-terminal symbols;

P, is a finite set of production rules;

X, is a start symbol in N.

Terminal symbols are the words that constitute the alphabet of the language (represented in italics in the subsequent examples).

Non-terminal symbols represent the grammatical categories, that may be sentence (in short S), noun phrase (in short NP), verb phrase (in short VP), prepositional phrase (in short PP), determiner (in short DET), noun (in short N), verb (in short V), preposition (in short PREP), etc.

A production rule consists of a single non-terminal symbol, followed by an arrow → that is followed by a finite sequence of terminal and/or non-terminal symbols. They express how different grammatical categories can be built up.

Any sequence of terminal symbols that can be derived from the start symbol is called sentence. Therefore, the set of sentences that can be derived from the start symbol applying the set of production rules constitute the language generated by the grammar.

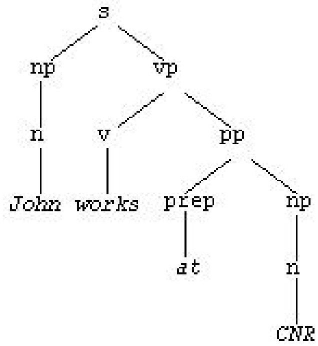
The following is an example of a CFG for a small fragment of English:

```
S -> NP VP       % A sentence (s) is a noun phrase (np)
                 plus a verb phrase (vb)
NP -> N          % A noun phrase is a noun
VP -> V PP       % A verb phrase is a verb plus a
                 prepositional phrase (pp)
PP -> PREP NP    % A prepositional phrase (pp) is a
                 preposition plus a noun phrase
N -> John        % 'John' 'CNR' are nouns
N -> CNR
PREP -> at       % 'at' is a preposition
```

`V -> works` % 'works' is a verb

Consider the string of words "*John works at CNR*". This is a sentence in the language defined by the previous grammar, since the sequence of terminal symbols "*John*" "*works*" "*at*" "*CNR*" can be derived from the start symbol *S* by repeatedly applying the production rules.

In addition to the information about the grammaticality of a sentence, CFGs can be used also to parse (i.e. syntactically analyze) sequences of terminal symbols, assigning them a structure in the form of a parse tree. For instance, the following is a parse tree for the sentence "*John works at CNR*":



To know the sentence structure is very important in order to understand what the sentence actually means (that is, if we wanted to do *semantics*).

The main advantages of CFGs are the simplicity and the possibility to use methodologies and tools that have been widely studied for more than half a century.

However, two limitations of the CFG formulation that need to be overcome in order to use this formulation in MLP are the following:

- there is no possibility to represent symbols (neither terminal nor non-terminal) from different modalities;
- there are no explicitly defined constructions for modeling semantic aspects of input symbols.

3.3.2 Multi-Modal Definite Clause Grammar

The Multi-Modal Definite Clause Grammar (MM-DCG) [ShT95] is the first reported grammatical framework for multimodal languages. This grammar is an extension of Definite Clause Grammar (DCG) [PeW80]. The following is a brief introduction to some of the main features of DCG. Afterwards, a description of the additional characteristics of MM-DCG over DCG can be found, along with a brief discussion on advantages and limitations of MM-DCG for MLP.

DCGs are an evolution of context free grammars that have proven their usefulness for describing natural languages, and that may be conveniently expressed and executed in Prolog.

Unlike the context-free case, in which only simple non-terminal symbols can be expressed, in DCG non-terminals are allowed to be compound terms. Moreover, in the head of a rule, in addition to non-terminals, lists of terminals and sequences of Prolog procedure call (written within the curly brackets) can occur.

DCGs allow to build representations of the meaning of sentences by adding extra arguments to the non-terminal symbols. For instance, if the meaning of a proper noun “John” is “john” an argument is added to the rule “proper_noun” in this way:

```
     proper_noun(john) --> [john]
```

The fragment of English, which has been written using CFG formulation in the previous paragraph, can be expressed in the following DCG notation:

```
   sentence(s(NP,VP))   -->   noun_phrase(NP),
verb_phrase(VP).
   noun_phrase(np(ProperNoun))-->
propernoun(Propernoun).
   noun_phrase(np(Noun)) --> noun(Noun).
   verb_phrase(vp(IV,PP))-->
intrans_verb(IV), prep_phrase(PP).
   prep_phrase(prep(P,NP))     -->     prep(P),
noun_phrase(NP).
   prep(p(at))--> [at], {is_preposition(at)}.
   propernoun(propernoun(john))-->[john],
{is_propernoun(john)}.
   noun(noun(CNR)) --> [CNR], {is_noun(CNR)}.
   intrans_verb(IV(works))     -->     [works],
{is_intrans(works)}.
```

The Multimodal Definite Clause Grammar (MM-DCG) extends DCG in the following ways:

- Any input from every stream contains the start and end times. Therefore, each item of an input sequence is represented in the following way:

  ```
  Input(start-time,end-time,<actual
  input>)
  ```

  This means that the actual input was begun at `start-time` and completed at `end-time`. This extension permits to define chronological constraints among categories.

- A non-terminal symbol in the head of a rule may be accompanied by the consuming stream name. As an example, consider the following rule:

  ```
  noun_phrase --> speech:pronoun
  ```

  This means that whereas a pronoun category is generated from the speech stream a noun-phrase is found.
  If the non-terminal symbol is not accompanied by any consuming stream name, it is regarded as coming from whichever modes.

- A terminal symbol is always accompanied by a specific stream name. For example, the following rule:

  ```
  noun(noun(CNR)) --> keyboard:[CNR]
  ```

  means that if a string “CNR” is inputted via the keyboard stream, the noun category is instantiated and an argument with the meaning of that symbol is added to the noun category.

The major advantages of MM-DCG include the capability to handle an arbitrary number of modalities as well as temporal information in grammar rules. Temporal information is tightly integrated into the grammar formulation in the form of time variables and time-out specifications.

However, this grammar lacks of representing semantic aspects, mainly, for the combined input. Moreover, it is not independent from the programming language for executing the grammar rules, since specific constructions for calling Prolog procedures into the grammar rules are provided.

3.3.3 Finite-State Multimodal Grammar

The Finite-state Multimodal Grammar (FMG) has been proposed by Johnston and Bangalore [JoB05] to support parsing and interpretation of multimodal utterances.

The FMG relies on a finite-state device that operates over n+1 tapes, where n tapes represent the inputs from n possible modalities and the n+1th tape represents their combined semantic meaning.

The syntax of FMG is composed of:

- a set of non-terminal symbols that, similarly to CFG formalism, represent the grammatical categories (S, NP, VP, V, etc);
- a set of terminal symbols, in which the multimodal aspects of the grammar are noticeable. In fact, each terminal contains n+1 components corresponding to the n+1 tapes of the finite-state device. For instance, for a three tape finite-state device which reads speech and gesture input, the terminal symbols are expressed in the following way:

  W:G:M

  where *W* is the spoken language stream, *G* is the gesture stream and *M* is the combined meaning;
- a set of production rules that are quite similar to the CFG rules. In fact, the body of the rule may contain only non-terminal symbols, while the head of the rule both terminals and non-terminals.

An example of a FMG for a small fragment of English is:

```
S --> NP,VP  ε:ε:])
NP --> N
```

```
VP --> V ADV
N --> john:ε:john([
V --> works:ε:works
ADV --> here:G:place( ENTRY
ENTRY --> ε:e1:e1  ε:ε:)
```

The ε is the empty symbol, and symbol e_1 is used as reference to the entity referred to by the gesture *G*. This fragment of grammar allows to parse a multimodal sentence composed of the spoken words "*John works here*" and the gesture at the appropriate organization icon on the screen.

The FMG has the benefit to enable a higher level of compensation for recognition errors in individual modes, since it directly influences the recognition phase of unimodal input. Moreover, FMG has the capability to represent the combined semantic meaning of multiple inputs.

However, this formalism is strongly centered on speech and gesture input. More importantly, it does not support mutual disambiguation, i.e., using the speech recognition information to inform the gestural recognition processing, or the processing of any other modality.

3.3.4 Multimodal Functional Unification Grammar

The Multimodal Functional Unification Grammar (MUG) is a non-deterministic grammar formalism defined by Reitter et al. [RPC04] for specifying adaptable user interfaces. The authors started from the Functional Unification Grammars (FUGs) [Kay79] [ElR92], a well-known technique for NLP from which MUG is derived.

FUGs follow a unification-based approach that, as suggested by the name, is a formalism where unification is the only information-combining operation [Shi86]. More specifically, entities are represented by feature structures (attribute value matrices, previously introduced in Section 3.2.2), and information carried by such entities is combined only through unification.

MUG extends FUG by introducing the possibility to support several coordinated modes and to unify one grammar rule for each mode. To achieve that, the MUG is composed of a set of components, named functional descriptions (FDs). Each FD is an attribute-value matrix and specifies a realization variant for a given partial semantic and syntactic representation (specific to a mode or generic), similar to a production rule in a CFG grammar. Variables can be named and always start with an upper-case letter.

MUG is based on the unification of such attribute-value matrices. In particular, a FD is unified with each *m*-constituent substructure, i.e. a FD that has an attribute path *m*|cat which is designed as a constituent for the mode *m*.

An example of MUG functional description for the confirmation of tasks by voice or screen is shown in Figure 3.10. The FD is obtained by the unification of four m-constituents. The symbols $\boxed{1}$ $\boxed{3}$ $\boxed{4}$ $\boxed{5}$ denote the elements that are the same, i.e. shared.

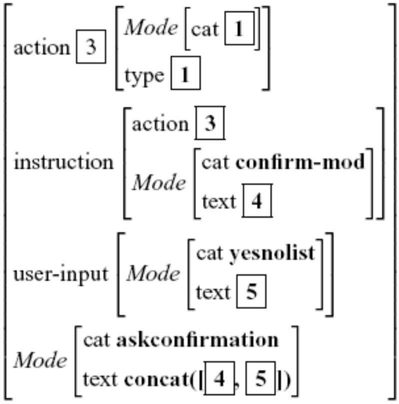


Figure 3.10: An example of MUG functional description

The main advantages of MUG consist in:

- the possibility to support several coordinated modalities;
- the uniform representation of syntax and semantics into an overall structure, which makes the grammar easier to maintain;
- the improvement of parsing disambiguation since it allows to use semantics to prune possible alternatives.

At the same time, there are certain drawbacks to using the MUG approach. First of all, it is not sufficiently amenable to capturing detailed lexical semantic properties. In fact, as observed by Fodor and Lepore [FoL98] the real meaning of a sentence is not easily captured by a fixed set of FDs. Moreover, since the FDs of a language are numerous and each FD is defined by using prose, the definition of complex categories requires a high computational effort that makes this formalism difficult to use.

3.3.5 Multimodal Combinatory Categorial Grammar

Multimodal Combinatory Categorial Grammar (MCCG) is one of the most recent approaches to MLP developed by Sun et al. [SSC07]. MCCG is an adaptation of Combinatory Categorial Grammar (CCG) [Ste00] [StB03] for multimodal utterances.

Similarly to CCG, the MCCG is a form of lexicalised grammar based on a set of syntactic rules, whose application is conditioned on the syntactic type (or category) of their inputs.

Categories identify constituents of the grammar and may be atomic categories or functions. The atomic categories (e.g. NP, PP, S, N) may have some features, such as number, case, etc. Functions are a combination of atomic categories, the forward application operator (/), the backward application operator (\) and appropriate bracketing to define the order in which the categories must be combined.

The set of MCCG rules allows to combine categories and to type-raise a category to another one. In particular, in addition to the two basic rules of forward and backward application, i.e. X/Y Y => X and Y X\Y => X respectively, the MCCG rules include:

- forward composition rule:	X/Y Y/Z => X/Z
- backward composition rule:	Y\Z X\Y => X\Z
- forward type-raising rule:	X => T/ (T\X)
- backward type-raising rule:	X => T\ (T/X)
- forward crossing composition rule:	X/Y Y\Z => X\Z
- backward crossing composition rule:	Y/Z X\Y => X/Z

The forward composition rule means that a function (X/Y) takes another function (Y/Z) to its right and returns function (X/Z). By applying these rules the complete parse tree can be built from the bottom to up.

For example, consider the sentence composed of the spoken words "*John works here*". *John* will be assigned the atomic category N. The intransitive verb *works* is an entity that will expect one argument (N) to its left and a prepositional phrase (PP) to its right. Once this argument is applied the result will be a sentence (S). Therefore the verb will be tagged (N \ S) / PP. The application of MCCG rules to these categories allows to parse the sentence, as shown in the following:

```
John        works       here
 N         (N\S)/PP      PP
           -----------------------------=>right
 N                N\S
--------------------------------=>left
        S
```

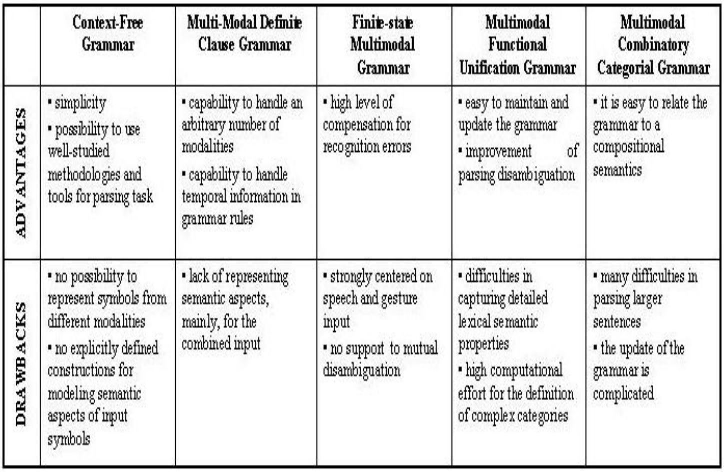
Analogously to CCG, MCCG has the advantage that it is easy to relate the grammar to a compositional semantics by assigning semantic values to the lexical entries and semantic functions to the combinatory rules such that no intermediate representation is required [StB03].

However, there are several drawbacks in the use of MCCG for multimodal language processing. First of all, the parse tree is defined by categories that may become increasingly complex in larger sentences, leading to many difficulties in parsing them. Secondly, the update of the grammar is much more complicated that with simple CFG. In fact, since the parse tree is completely dependent on the word categories and in complex sentences there are interdependencies between words, if a change of a word category occurs other word categories have to be changed causing a ripple effect throughout the parse tree.

3.3.6 Final Discussion on Multimodal Grammars

As discussed in the previous sub-paragraphs, each multimodal grammar formalism has a set of advantages and drawbacks, which can be summarized as shown in Table 3.2

Table 3.2: Advantages and shortcomings of multimodal grammar formalisms



	Context-Free Grammar	Multi-Modal Definite Clause Grammar	Finite-state Multimodal Grammar	Multimodal Functional Unification Grammar	Multimodal Combinatory Categorial Grammar
ADVANTAGES	▪ simplicity ▪ possibility to use well-studied methodologies and tools for parsing task	▪ capability to handle an arbitrary number of modalities ▪ capability to handle temporal information in grammar rules	▪ high level of compensation for recognition errors	▪ easy to maintain and update the grammar ▪ improvement of parsing disambiguation	▪ it is easy to relate the grammar to a compositional semantics
DRAWBACKS	▪ no possibility to represent symbols from different modalities ▪ no explicitly defined constructions for modeling semantic aspects of input symbols	▪ lack of representing semantic aspects, mainly, for the combined input	▪ strongly centered on speech and gesture input ▪ no support to mutual disambiguation	▪ difficulties in capturing detailed lexical semantic properties ▪ high computational effort for the definition of complex categories	▪ many difficulties in parsing larger sentences ▪ the update of the grammar is complicated

The multimodal grammar proposed in this thesis follows the context-free paradigm. This choice was partly motivated by the ultimate aim of the multimodal language processor, presented in this dissertation, to define and update multimodal language,learning it by example.

Since a multimodal language is characterised by a large variety of linguistic syntactic phenomena, the ideal candidate grammar, which is sufficiently expressive to represent any of these phenomena, is the class of context-sensitive grammars. This kind of grammar is similar to CFG,except that the body of the production rules may contain both terminal and non-terminal symbols. The name context-sensitive comes from the fact that the expansion of a symbol depends on its context (i.e., the position of the symbol in a sequence or the relationship with surrounding symbols).

Context-sensitive grammars, however, have two shortcomings with respect to multimodal language processing:

- parsing complexity: all known algorithms for parsing these grammars have exponential time dependency.

- too needless expressiveness: only a few linguistic phenomena, such as cross-serial dependency (that can occurs in Dutch and Swiss-German languages), require the expressiveness of context-sensitive grammars. In the majority of cases, the sublanguages generated by these grammars do not occur in multimodal language.

Consequently, although CFGs have less expressive power than context-sensitive ones, they are able to model all frequent linguistic phenomena of multimodal language assuring, at the same time, a lower parsing complexity. For these reasons this thesis has been developed opting for context-free grammars instead of context-sensitive ones.

However, as said earlier, in order to use CFG for multimodal language processing it is necessary to overcome the two main deficiencies of this grammatical formalism, i.e. the lack of constructions both for representing input symbols from different modalities (and how they are combined together into the input sentence), and for modeling semantic aspects of input symbols.

The four grammatical frameworks presented in the previous subparagraphs constitute an attempt to resolve these issues. In particular, MM-DCGs, following the context-free paradigm of DCGs, introduce a notation that allows to specify the input modality associated with input symbols, their temporal information and their semantic meanings. However, this formalism does not overcome the limitation due to the lack of representing semantic aspects, mainly, for the combined input. This last issue is evident also in MUGs, which, in addition, have the disadvantage that do not follow a context-free paradigm. On the contrary, FMGs provide a solution to the limitation of capturing semantic properties of multimodal input, but they do not allow to handle whichever modality as they are strictly related to speech and gesture input only. MCCGs, too, provide a solution for assigning semantic values to lexical input but, similarly to MUGs, they do not rely on a context-free paradigm.

The grammatical framework proposed in this dissertation tries to join together the efforts made by the aforementioned multimodal grammar formalisms in overcoming the CFG limitations. In particular, the proposed grammar, named Multimodal Attribute

Grammar (MAG), is based on the context-free paradigm and provides constructions for representing multiple input streams, the meaning of these inputs as well as temporal relationships among inputs. A detailed description of the MAG notation is given in Section 5.4.

Chapter 4

Learning of Grammars

In this chapter a survey of existing methodologies for inferring context-free grammars from sample sentences is presented. After introducing some preliminary definitions and notations concerning learning and inductive inference, the attention will be focused on three existing models of learning. The last section of the chapter will explore the state of the art concerning the algorithms for learning context-free languages and grammars.

4.1 Introduction

The mathematical theory of language learning (also known as learnability theory, grammar induction, or grammatical inference) deals with idealized learning procedures for acquiring grammars on the basis of exposure to evidence about languages [Pul03]. A more accurate definition of language learning and some notations used in the chapter will be introduced in Section 4.2.

The main research studies in grammatical inference, particularly for CFGs, have been made in several application domains, such as speech recognition [Bak79], computational linguistics [Adr92], computational biology [SBH94][SaB02], and machine learning [Sak97][HiO03].

All these studies agree with the fact that the learnability of various language classes, either in the Chomsky hierarchy (i.e. regular languages, context-free languages, context-sensitive languages, and unrestricted languages) or not, is a hard problem. From a mathematical point of view, three different reference models of learning have been studied in the literature. The first more classical paradigm, namely identification in the limit model, was presented by Gold [Gol67] in the middle 1960s. Then, the learning with queries model was proposed by Angluin [Ang81] in the early 1980s. The most recent model, named Probably Approximately Correct (PAC) model, was proposed by Valiant [Val84] in the middle 1980s. Some details of these models of learning will be given in the Section 4.3.

The majority of these learning models takes as input an initial set of positive training examples and output the language description, i.e. the specific grammar that is able to recognize only these examples. To achieve that, a set of negative examples (i.e. sentences that should not be recognized by the grammar) is also needed for limiting the extent of generalisation, as an overly general grammar will never be refuted considering a new positive example.

Therefore, the two main issues that grammar inference methodologies have to face are the overspecialisation (or over-fitting) and the over-generalisation. The former occurs when the inference process produces a grammar whose language is smaller than the unknown target language (which is always the case when algorithms are not trained ad infinitum). This issue can be prevented by some extent setting aside some data (which takes part of the so-called “validation set”) and measuring performance on this data after each training example has been processed. Analogously, the latter occurs when the inference process produces a grammar whose language is larger than the unknown target language. Over-generalisation can be controlled by using a set of negative examples.

In Multimodal Language Processing (MLP), similarly to NLP, large sets of positive examples may be available but it is rarely possible to obtain a set of negative examples for training. To overcome this lack of negative evidence, two solutions have been proposed in the literature:

- to restrict the language to one of the classes of formal languages, which have been proven to be learnable from positive examples only, such as reversible languages [Ang82], k-testable languages [GaV90], code regular and code linear languages [EST96], pure context-free languages [KMT97] and strictly deterministic automata [Yok95].

- to introduce various heuristics aiming to avoid over-generalisation without the use of negative examples, such as simplicity [LaS00].

The first solution does not fit with the choice of context-free language as basic paradigm for the multimodal grammar proposed in this dissertation. Consequently, the use of heuristics is the solution that can be applied for avoiding over-generalisation in multimodal grammar inference. The issue of learning CFGs from positive examples only and the proposed heuristics for solving the overgeneralization problem will be discussed in Section 4.4.

4.2 Notations

Following Gold [Gol67], in order to specify a learning environment it is necessary to specify:

- the class of languages $\mathcal{L}$ to be inferred;
- the language description (or hypothesis) class $\mathcal{H}$ used to describe the languages in $\mathcal{L}$, which corresponds to the grammar in our case. Let $h \in \mathcal{H}$, $L(h)$ denotes the language described by h;
- the way the learning process obtains information.

This statement has been followed by Lee [Lee96] for defining what the problem of grammatical inference means, whereby it is, in its broadest sense, the problem of learning a description of a language (i.e. a grammar) from data drawn from the language.

Formally, a learning algorithm L_A can be modeled as a function that takes as input a finite sequence of examples and gives as output a language description. A presentation is an infinite sequence of examples. Two types of presentations are usually allowed:

- A *text* for a language L is an infinite sequence of strings $x_1, x_2, \ldots$ from L such that every string of L occurs at least once in the text. The inference algorithms that use this type of information are said to learn from *positive examples*. Note that the class of all the possible text presentations for a language L is denoted by $\mathcal{P}_L$.

- An *informant* for a language L is an infinite sequence of pairs $(x_1, d_1), (x_2, d_2), \ldots$ in $L \times \mathbb{B}$, (where $\mathbb{B}$ is the set of Booleans) such that every string of L occurs at least once in the sequence and $d_i = \text{true} \Leftrightarrow x_i \in L$. The inference algorithms that use this type of information are said to learn from positive and negative examples. Note that the class of all the possible informant presentations for a language L is denoted by $\mathcal{PN}_{\mathrm{L}}$.

Therefore, a standard classification of learning algorithms can be done according to the presentation they adopt. In this chapter this classification is used for describing some examples of algorithms (see Section 4.4) that learn from text as well as informant.

4.3 Models of Learning

Language learning can be studied in a mathematical way by considering three different models of learning: identification in the limit [Gol67], Queries [Ang88], and PAC learning [Val84]. In the following sub-paragraphs further notions about these models are given.

4.3.1 Identification in the Limit

Identification in the limit is the most classic paradigm of learning, presented in a seminal article by Mark Gold [Gol67], which views inductive inference as an infinite process. In fact, in this paradigm a learning procedure is an algorithm infinitely running on a never-ending stream of inputs. The inputs are grammatical strings chosen from a target language in a known class of languages. That language has to be identified by choosing a grammar for it from a known set of grammars. At each point in the process, any string in the language might be the next string that turns up (strings can turn up repeatedly). After each input, the algorithm updates the grammar, so that it conforms to the new training string. Success in identifying a language "in the limit" consists in achieving a grammar that does not change when an additional string is inputted and which is correct for the target language.

Identifiability in the limit is a fragile property, however. Gold proved that none of the standard classes of formal languages (e.g., the regular languages, the context-free languages, the context-sensitive languages, or the unrestricted languages) are identifiable in the limit from text. On the contrary, regular, context-free and context-sensitive languages are identifiable in the limit from an informant.

However, in NLP and MLP, learners typically get evidence about what is grammatical (positive samples), but no details about what is not grammatical (negative samples). Therefore, natural and multimodal languages require to be identifiable in the limit from text. Such a requirement is supported by the statement of Gold whereby, choosing a number *k* and considering the class of all context-free languages generable by a context-free grammar with not more than *k* rules, it is demonstrable that every choice of *k* defines a class that is identifiable in the limit from text. And the same is true for context-sensitive grammars.

Consequently, if positive results for learning from positive examples are expected, it is necessary to restrict the language to non super-finite sub-classes of the context-free languages.

4.3.2 Queries

Learning with queries is another popular learning model that has been introduced by Angluin [Ang81] [Ang88]. In this model, the learner makes use of an oracle that is able to answer some questions about the target language. Generally, two types of queries, known as membership queries and equivalence queries, may be used. The former returns “true” if the given string belongs to the language, “false” if not. The latter is made by presenting to the oracle a grammar for a hypothesis language.

Similarly to identifiability in the limit model, learning with queries is a not useful model for dealing with context-free languages. Angluin [Ang90] showed that context-free grammars are not learnable from equivalence queries alone and that membership queries alone are insufficient. However, restricting the language to a simple deterministic language, i.e. recognizable by a deterministic push-down automaton by empty store, the learning with queries model gives positive results.

The problem is that these languages do not allow to express all linguistic phenomena occurring in natural and multimodal languages. Consequently, the learning with queries model do not guarantee exact inference of these languages.

4.3.3 PAC Learning

The "Probably Approximately Correct" (PAC) learning was introduced by Valiant [Val84] in an attempt to model distribution independent learning. The basic idea of PAC learning is that it is possible to minimize the chance of learning something that is wrong without being completely sure that this is right. Valiant applied his theory to Boolean concept learning.

Unfortunately, PAC learning, in its pure distribution free form, does not help Grammar Induction much. Even simple classes of languages are known to be not PAC learnable [Den01] [Den98]. In particular, although the PAC learning model takes many features of natural learning into account, in most cases it fails to describe such kind of learning.

4.4 Algorithms for Learning of Context-Free Grammars

The majority of grammar inference algorithms presented in the literature is based on an initial set of positive training examples and a specific grammar that is able to recognize only these examples.

According to the choice of following the context-free paradigm for defining the proposed multimodal attribute grammar, in the following subparagraphs three existing grammatical inference algorithms for CFGs are explored. The first is the inductive CYK algorithm, which belongs to the class of algorithms that learn from an informant. A further algorithm of this class is the learning by version space algorithm, which is presented in Section 4.4.2. Finally, an example of algorithm, named e-GRIDS, that learns from text is analyzed in Section 4.4.3.

4.4.1 Inductive CYK Algorithm

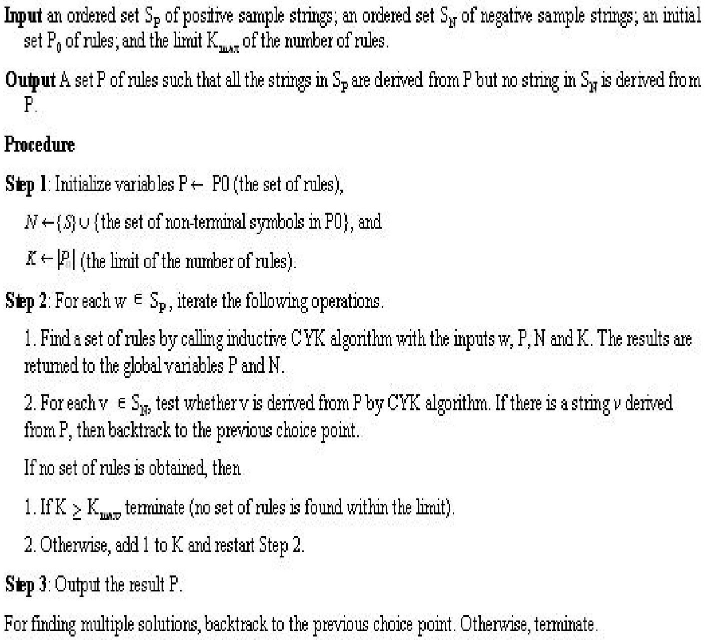
In this section the grammatical inference approach, namely inductive CYK (Cocke, Younger, Kasami) algorithm, is explored.

This approach, proposed by Nakamura et al. [NaI00, NaM02, Nak03], is implemented in an inductive grammar inference system called Synapse (Synthesis by Analyzing Positive String Examples).

Roughly, the algorithm synthesizes CFGs from positive and negative sample strings generating the minimum production rules, which derive positive strings, but do not derive any given negative strings. All the production rules generated by the algorithm follow the extended Chomsky Normal Form (extended CNF), that is have the forms A → β and A → βγ, where A is a non-terminal symbol, β and γ are terminal symbols. An important feature of the extended CNF is that it is simpler than the standard CFG production rules reducing the computation time of the inference process.

The grammatical inference approach of Synapse employs two main procedures, the top-level procedure and the procedure that implements the extended inductive CYK algorithm.

The top-level procedure is shown in Figure 4.1. It takes as inputs two ordered sets S_P and S_N of positive and negative sample strings, respectively, and an initial set P_0 of rules. The procedure searches for the set P of rules, that contains the set P_0 ($P \subseteq P_0$), and the set N of non-terminals such that all the string in S_P can be derived from P but no string in S_N is derived from P. This search is carried out by calling inductive CYK algorithm. A control on the search is performed by iterative deepening on the number of rules to be generated. Starting from an initial limit k of the number of rules, this limit is increased by one when the system fails to generate enough rules to parse the sample within this limit and repeats the search.



Input an ordered set S_P of positive sample strings; an ordered set S_N of negative sample strings; an initial set P_0 of rules; and the limit K_{max} of the number of rules.

Output A set P of rules such that all the strings in S_P are derived from P but no string in S_N is derived from P.

Procedure

Step 1: Initialize variables P ← P0 (the set of rules),

$N \leftarrow \{S\} \cup$ {the set of non-terminal symbols in P0}, and

$K \leftarrow |P|$ (the limit of the number of rules).

Step 2: For each $w \in S_P$, iterate the following operations.

1. Find a set of rules by calling inductive CYK algorithm with the inputs w, P, N and K. The results are returned to the global variables P and N.

2. For each $v \in S_N$, test whether v is derived from P by CYK algorithm. If there is a string v derived from P, then backtrack to the previous choice point.

If no set of rules is obtained, then

1. If $K \geq K_{max}$, terminate (no set of rules is found within the limit).

2. Otherwise, add 1 to K and restart Step 2.

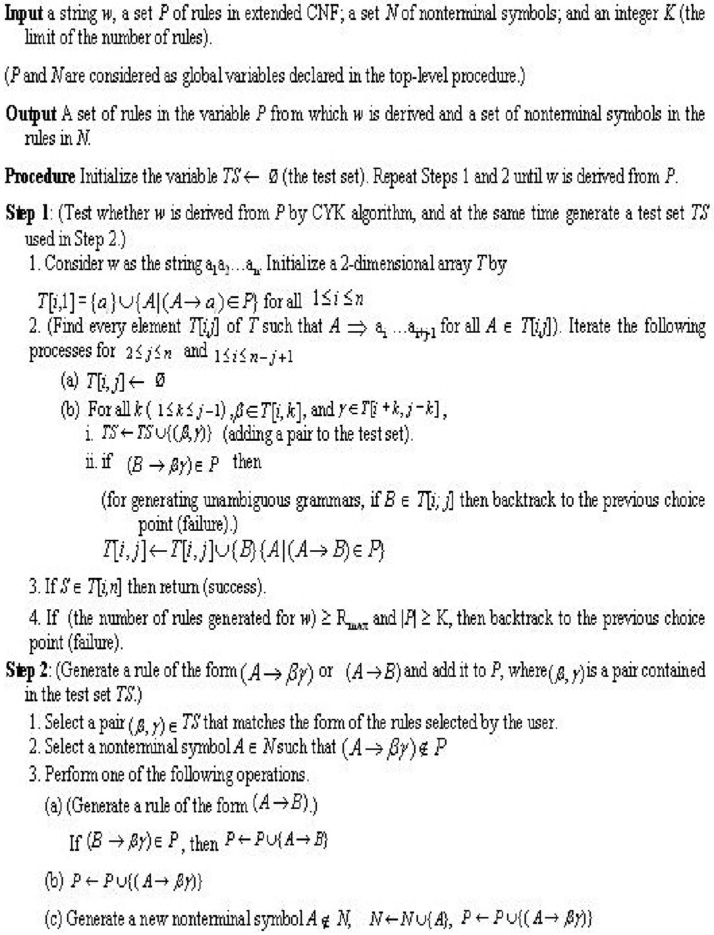
Step 3: Output the result P.

For finding multiple solutions, backtrack to the previous choice point. Otherwise, terminate.

Figure 4.1: The top-level procedure of Synapse

The procedure of the extended inductive CYK algorithm is shown in Figure 4.2. It takes as inputs a string *w* and a set P_0 of rules and outputs a set P_1 of rules such that *w* is derived from $P_0 \cup P_1$.

The extended inductive CYK algorithm is composed of two steps that have to be repeated until *w* is derived from the production rules. The first step includes CYK algorithm for testing whether the string *w* can be derived from P_0. This algorithm makes use of a variable TS that keeps the test set of symbol pairs (β, γ), to which a rule $A \rightarrow \beta\gamma$ is applied during the running of the algorithm. These pairs are candidates of the body of newly generated rules. The second step provides a function for adding production rules when the set P_0 does not derive the string *w*. The rules that are produced (included in the set P_1) are in the form $A \rightarrow \beta\gamma$ or $A \rightarrow B$, where (β, γ) is a pair contained in the test set TS.



```
Input a string $w$, a set $P$ of rules in extended CNF; a set $N$ of nonterminal symbols; and an integer $K$ (the
    limit of the number of rules).
($P$ and $N$ are considered as global variables declared in the top-level procedure.)
Output A set of rules in the variable $P$ from which $w$ is derived and a set of nonterminal symbols in the
    rules in $N$.
Procedure Initialize the variable $TS \leftarrow \emptyset$ (the test set). Repeat Steps 1 and 2 until $w$ is derived from $P$.
Step 1: (Test whether $w$ is derived from $P$ by CYK algorithm, and at the same time generate a test set $TS$
    used in Step 2.)
    1. Consider $w$ as the string $a_1 a_2 \ldots a_n$. Initialize a 2-dimensional array $T$ by
       $T[i,1] = \{a\} \cup \{A \mid (A \rightarrow a) \in P\}$ for all $1 \leq i \leq n$
    2. (Find every element $T[i,j]$ of $T$ such that $A \Rightarrow a_i \ldots a_{i+j-1}$ for all $A \in T[i,j]$). Iterate the following
       processes for $2 \leq j \leq n$ and $1 \leq i \leq n-j+1$
        (a) $T[i,j] \leftarrow \emptyset$
        (b) For all $k$ ( $1 \leq k \leq j-1$ ), $\beta \in T[i,k]$, and $\gamma \in T[i+k, j-k]$,
            i. $TS \leftarrow TS \cup \{(\beta,\gamma)\}$ (adding a pair to the test set).
            ii. if $(B \rightarrow \beta\gamma) \in P$ then
                (for generating unambiguous grammars, if $B \in T[i;j]$ then backtrack to the previous choice
                    point (failure).)
                $T[i,j] \leftarrow T[i,j] \cup \{B\} \{A \mid (A \rightarrow B) \in P\}$
    3. If $S \in T[i,n]$ then return (success).
    4. If (the number of rules generated for $w$) $\geq R_{max}$ and $|P| \geq K$, then backtrack to the previous choice
       point (failure).
Step 2: (Generate a rule of the form $(A \rightarrow \beta\gamma)$ or $(A \rightarrow B)$ and add it to $P$, where $(\beta, \gamma)$ is a pair contained
    in the test set $TS$.)
    1. Select a pair $(\beta, \gamma) \in TS$ that matches the form of the rules selected by the user.
    2. Select a nonterminal symbol $A \in N$ such that $(A \rightarrow \beta\gamma) \notin P$
    3. Perform one of the following operations.
       (a) (Generate a rule of the form $(A \rightarrow B)$.)
           If $(B \rightarrow \beta\gamma) \in P$, then $P \leftarrow P \cup \{A \rightarrow B\}$
       (b) $P \leftarrow P \cup \{(A \rightarrow \beta\gamma)\}$
       (c) Generate a new nonterminal symbol $A \notin N$, $N \leftarrow N \cup \{A\}$, $P \leftarrow P \cup \{(A \rightarrow \beta\gamma)\}$
```

Figure 4.2: The procedure of the extended inductive CYK algorithm

The extended inductive CYK algorithm has non-deterministic branches, or choice points, to which the control backtracks when the process fails. If the process terminates with success, the algorithm returns the sets of rules $P_0 \cup P_1$ and non-terminal symbols as a result. The algorithm can have multiple results for a single set of inputs, since the backtracking processes may generate different results.

The original CYK algorithm has worst-case computational complexity O(n^3), where n is the number of words in the sentence in input. The extended inductive CYK algorithm is similar to the usual CYK algorithm, except that when the rule set does not derive the sentence, it adds production rules so that the parsing always succeeds. Therefore, its computational complexity is still polynomial.

For increasing the synthesis speed several heuristics can be applied, as described in [Nak03]. First of all, when the process of inductive CYK algorithm generates a rule $A \rightarrow \beta\gamma$, a constraint can be applied that restricts the subsequent rule generation to not terminating until a rule containing A in the body is also generated. Secondly, a test on the effectiveness of the newly generated rules allows to perform an intelligent backtracking. In fact, whether any negative sample is derived from the set of newly generated rules, another rule may be generated in the redoing process. Finally, the use of a hash memory for checking whether each rule set has been processed, each time the system generates the set, allows to avoid repeated search.

The main advantages of the extended inductive CYK algorithm rely on the generation of simpler sets of rules and shorter computation times in the inference of CFG grammars for some simple languages.

A hard limitation of the grammar inference method of Synapse is that it cannot synthesize grammars with more than about 14 rules from their samples because of the computation cost.

4.4.2 Learning CFG by Version Space

The grammar inference algorithm based on version space has been proposed by Vanlehn and Ball [VaB87] and belongs to the class of algorithms that learn from an informant, i.e. it needs of positive and negative examples for inferring the grammar.

A version space is a set of all generalizations of a grammar, consistent with a given set of instances. The algorithm applies a particular induction technique, called version space strategy, which is based on a compact way of representing the version space. In particular, the central idea of this strategy is that the space of generalizations defined by the representation language can be partially ordered by generality.

For applying the version space strategy to grammar inference the main issue to face is that the version space is potentially infinite. A well-known theorem [VaB87] states that for any class of grammars the version space is infinite for any finite set of training examples.

To make the version space finite several restrictions are made. First of all, a restriction on the form of grammar rules is introduced by considering only simple CFGs, i.e. grammar in which rules have the following features: (i) no rule has an empty body, (ii) if a rule has just one symbol on its body, then the symbol is a terminal, (iii) and every non-terminal appears in a derivation of some string. Secondly, the grammar has to be reduced, i.e. all the rules in an inferred grammar are necessary for the derivation of some positive training examples. According to these restrictions, given a finite set of training examples (positive and negative), there are finitely many reduced simple CFGs consistent with those examples. Consequently, a finite version space is produced.

The version space strategy allows to calculate a reduced version space, but it cannot be directly applied due to the undecidability of the problem to testing whether the language generated by a grammar *A* includes the language generated by a grammar *B*. To solve this problem the version space algorithm makes use of three strategies:

- A set, called the *derivational version space*, that is a superset of the reduced version space and a subset of the version space.
- A computable predicate, called *FastCovers*, that is a partial order over grammars in the derivational version space.
- An *Update algorithm* for the maximal and minimal elements in FastCovers of the derivational version space.

Given a set of positive strings, the derivational version space is the set of grammars corresponding to all possible labeling of each tree sequence in the simple tree product for those strings, where a simple tree is a derivational (or parse) tree for a simple grammar, and a simple tree product is given by the Cartesian product over the sets of unlabelled simple trees for each strings of the given set.

Given a set of positive and negative strings, the derivational version space is the derivational version space for the positive strings minus those grammars that generate any of the negative strings.

For instance, consider the two positive strings 'b' and 'ab'. As there is one unlabelled tree for 'b' and four unlabelled trees for 'ab', so there are four tree sequences in the Cartesian product of the trees for 'a' and the trees for 'ab'. These four tree sequences constitute the simple tree product, which is shown in Figure 4.3.

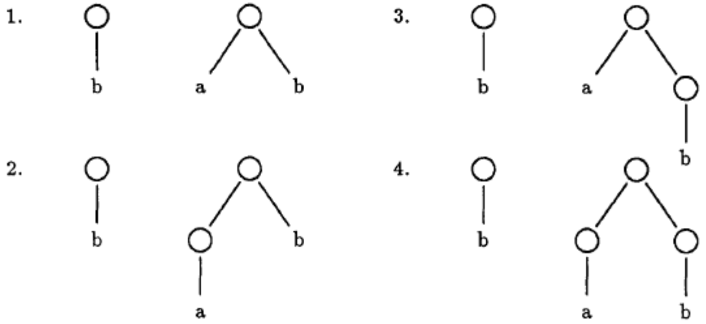


Figure 4.3: The simple tree product for the positive strings 'b' and 'ab'

For each of the four tree sequences, the construction of the derivational version space consists in partitioning the nodes in the trees and assigning labels. Figure 4.4 illustrates how the derivational version space is constructed for the fourth unlabelled tree sequence in Figure 4.3. Trees 1 through 5 show all possible partitions of the four nodes and the labeling of the trees that result. Each of the resulting labeled tree sequences is converted to a grammar, as shown in the third column of the figure. The derivational version space is the union of these grammars, which derive from the fourth tree sequence, with the grammars from the other tree sequences.

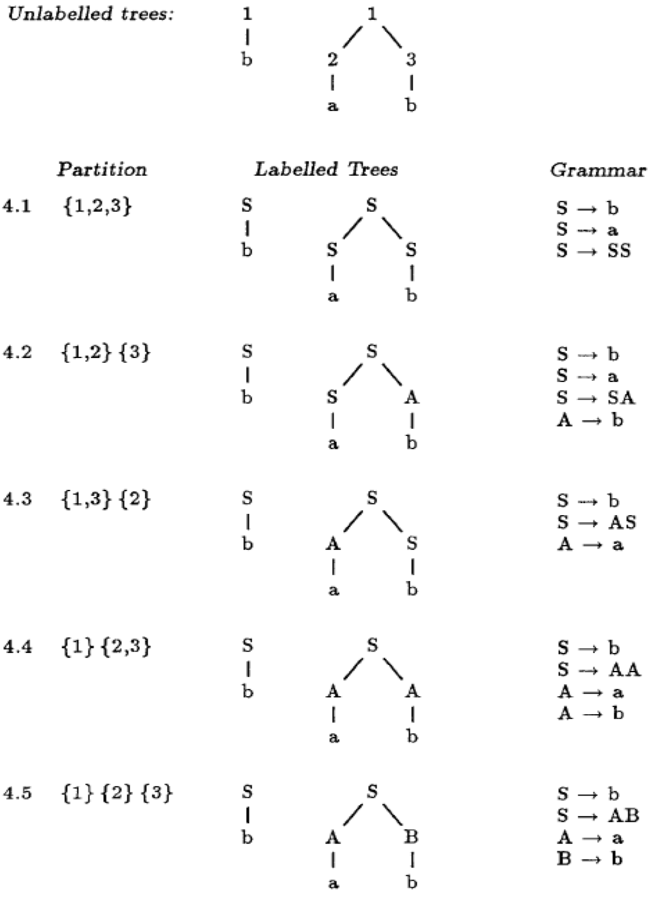


Figure 4.4: Construction of the derivational version space for the fourth tree sequence

At this point, the version space algorithm calculates a partial order for the set of grammars of the derivational version space by applying the *FastCovers* predicate. A formal definition of this predicate can be found in [VaB87]. This dissertation only highlights the usefulness of this operator within the version space algorithm for ordering the derivational version space.

Finally, the version space algorithm applies the *Update algorithm* that takes as input an instance and the current pair [*P+, G*], where *P+* is the set of positive examples and *G* is the current derivational version space, and returns a revision of the pair that is consistent with the given instance. Briefly, the *Update algorithm* proceeds in the following way:

1. If the string is positive and a member of *P+*, then do nothing and return the current version space. If the string is not a member of *P+*, then add it to *P+* and call the UpdateG+ procedure.
2. If the string is negative and a member of *P+*, then return NIL. If the string is not a member of *P+*, then call the UpdateG- procedure.

The task of the UpdateG- procedure is to modify *G*, so that none of the grammars will parse the negative string. All the grammars in *G* are organized in a queue and the procedure picks a grammar off the queue and verifies if it parses the negative string. If it does not, then the grammar is placed in *NewG*, the revised version of *G*. If it does parse the string, then the algorithm refines the node partition once, in all possible ways. As each of these partitions corresponds to a new grammar, the algorithm verifies if they parse the negative string and eventually places them in the *NewG* set. When the queue is exhausted, i.e. all grammars are verified, the *NewG* set contains the maximal set of the grammars that fail to parse the negative string.

The UpdateG+ procedure proceeds in the following three steps:

1. Given a positive string, form the set of all unlabelled simple derivation trees for that string.
2. For each grammar in the old *G* and for each tree for the new positive string,
   a. append the tree onto the end of the tree sequence of the grammar's triple, and
   b. allocate the new tree's nodes to the partition elements in all possible ways.

3. Place all the candidate grammars generated in the preceding step on the queue for the UpdateG- algorithm, which tests that the grammar is consistent with all the negative strings in the presentation that have been received so far.

To clarify the algorithm's behaviour a simple example is presented. Suppose that the set of training examples is composed of four positive and three negative strings of command words, that are *P* = {+"*delete all-of-them*", -"*all-of-them delete*", -"*delete delete*", +"*delete it*", -"*it it*", +"*print it*", +"*print all-of-them*"}. Considering the first string "*delete all-of-them*", there are four possible unlabelled simple trees that lead directly to four grammars for the derivational version space *G*, shown in Figure 4.5.

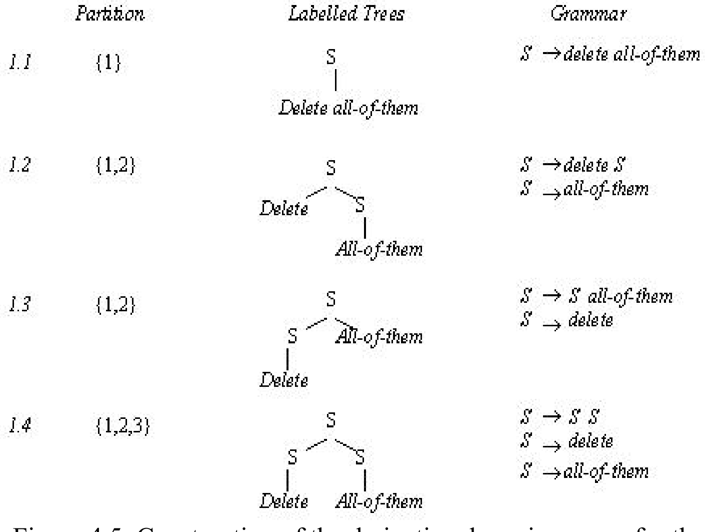
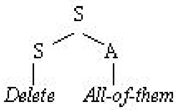


Figure 4.5: Construction of the derivational version space for the example

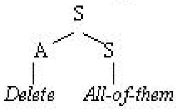
Suppose the next string is a negative string "*all-of-them delete*". This string cannot be parsed by grammars 1, 2 or 3, so they remain unchanged in the *G* set. The fourth grammar is overly general, so it is split in three legal partitions {1 2}{3}, {1 3}{2}, and {1}{2 3}. The first two survive becoming the grammar 1.5 and 1.6, shown below.

1.5 {1,2} {3}



$S \to S\ A$
$S \to$ *delete*
$A \to$ *all-of-them*

1.6 {1,3} {2}



$S \to A\ S$
$A \to$ *delete*
$S \to$ *all-of-them*

The next string is the negative instance "*delete delete*". None of the grammars in *G* parse this string, so the *G* set remains unchanged. The next string is positive, "*delete it*". There are four possible unlabelled simple derivation trees for this string. Each is paired with each of the five grammars in the current *G*, yielding 20 combinations. The resulting 20 grammars are queued and verified against *P-*. At the end of the version space algorithm the *NewG* set contains 25 grammars.

As demonstrated also by the previous example, the algorithm is inapplicable for a large set of training instances due to the combinatorial explosion inherent in the UpdateG+ algorithm when more instances are present. Moreover, the version space algorithm is not immediately applicable to grammar induction, because it produces a set of grammars and some other process have to choose among them. Therefore, the algorithm is good for being used as a general framework for the development of practical, task-specific learning machines.

4.4.3 e-GRIDS Algorithm

The e-GRIDS algorithm [PPK04] is a grammar inference method that extends the GRIDS algorithm [LaS00] by improving the search performed by the learning operators in the space of possible grammars. Like its predecessor, the e-GRIDS algorithm utilises a simplicity bias for inferring CFGs from positive examples only.

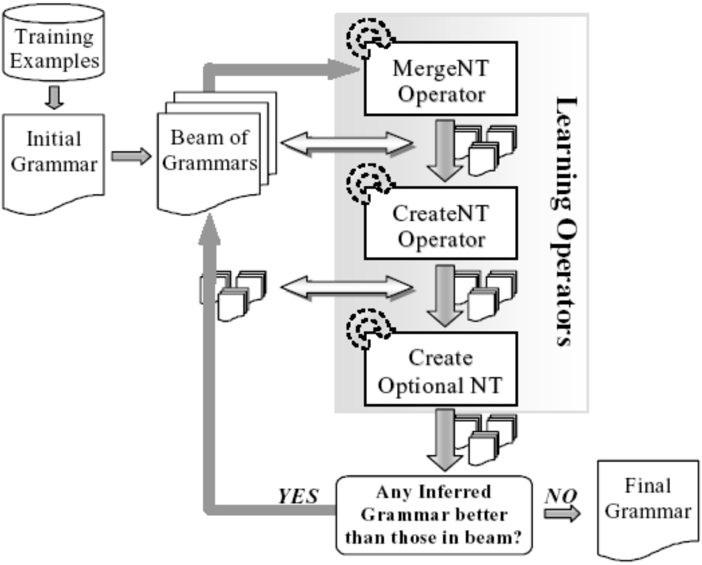


Figure 4.6: The e-GRIDS algorithm

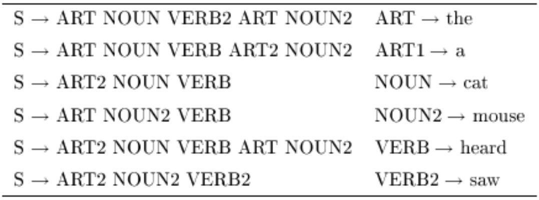
A general workflow of the e-GRIDS algorithm is shown in Figure 4.6. e-GRIDS uses the training sentences in order to construct an initial grammar by converting each one of the training examples into a grammatical rule. Subsequently, the learning process takes place, which is organised as a beam search. Having an initial hypothesis (the initial grammar) in the beam, e-GRIDS uses three learning operators in order to explore the space of CFGs:

- *MergeNT* operator, which merges two non-terminal symbols into a single symbol *X*, thereby replacing all their appearances in the head and the body of rules by *X*;
- *CreateNT* operator, which creates a new non-terminal symbol *X* from two existing non-terminal symbols that are its constituent symbols.
- *Create Optional NT*, which duplicates a rule created by the CreateNT operator and appends a non-terminal symbol to the rule, thus making this symbol optional.

The learning process occurs in three steps, according to the operator that is applied. In the first step, called "merge" step, the MergeNT operator is repeatedly applied for merging non-terminal symbols in each grammar in the beam. The resulting grammars are then evaluated for deciding if replacing the grammar in the beam that has the lowest score with the newly generated grammar that has a better score. The second step is the "create" step that considers all ways of creating new terms from pairs of symbols that occur in sequence within the grammar, by repeatedly applying the CreateNT operator. Finally, in the "create optional" step all ways of duplicating a rule by the addition of an optional extra symbol at the end of the rule body are examined by repeatedly applying the CreateOptionalNT operator. The learning process terminates when it is unable to produce a successor grammar that scores better than the ones in the beam.

As mentioned above, the e-GRIDS algorithm uses a simplicity bias for directing the search through the space of CFGs and avoiding overly general grammars. This criterion measures the simplicity of a grammar through its *description length* that is defined as the sum of the number of symbols required to encode the grammar and the number of symbols required to encode the training examples. Therefore, the algorithm directs the learning process described above towards grammars that are compact, i.e. ones that have minimum description length.

To clarify the algorithm's behaviour a simple example is described. Suppose that the set of training samples is composed of the following six sentences: "*The cat saw the mouse*", "*The cat heard a mouse*", "*The mouse heard*", "*A cat saw*", "*A cat heard the mouse*", "*A mouse saw*". Therefore, the initial grammar is shown in Figure 4.7.

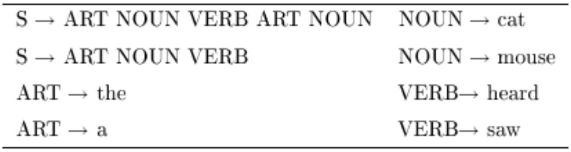


S → ART NOUN VERB2 ART NOUN2	ART → the
S → ART NOUN VERB ART2 NOUN2	ART1 → a
S → ART2 NOUN VERB	NOUN → cat
S → ART NOUN2 VERB	NOUN2 → mouse
S → ART2 NOUN VERB ART NOUN2	VERB → heard
S → ART2 NOUN2 VERB2	VERB2 → saw

Figure 4.7: The initial grammar for the e-GRIDS algorithm

This grammar has description length equal to 54, as the rules contain 30 words and there are six training sentences containing 24 words.

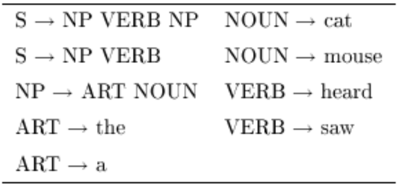
The e-GRIDS algorithm begins by considering all possible merges of symbols that occur in the heads of rules. Merging ART with ART2, VERB with VERB2, and NOUN with NOUN2 allows to eliminate four redundant sentences producing the grammar in Figure 4.8, which has a score of 22.



S → ART NOUN VERB ART NOUN	NOUN → cat
S → ART NOUN VERB	NOUN → mouse
ART → the	VERB→ heard
ART → a	VERB→ saw

Figure 4.8 The grammar after the “merge” step of the e-GRIDS algorithm

The algorithm continues applying the CreateNT operator that produces the term NP as an ART followed by a NOUN, and then replaces all occurrences of this sequence with the new symbol. This operation introduces another rule into the grammar, but it simplifies the two rules, giving 18 as evaluation score. At this point, the algorithm does not produce simpler grammar and therefore it terminates returning the grammar in Figure 4.9.



S → NP VERB NP	NOUN → cat
S → NP VERB	NOUN → mouse
NP → ART NOUN	VERB → heard
ART → the	VERB → saw
ART → a	

Figure 4.9: The final grammar produced by the e-GRIDS algorithm

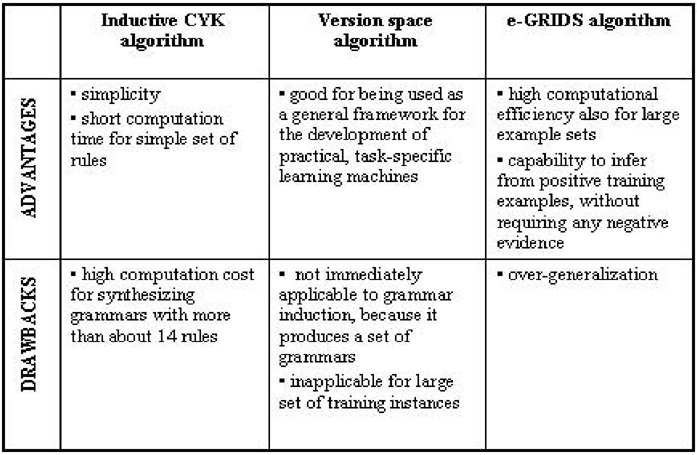
One the main advantages of the e-GRIDS algorithm is its computational efficiency which facilitates its scalability to large example sets. Although this algorithm is able to infer grammars that perform well, based on relatively small sets of training examples, it is also able to handle large example sets in significantly reduced amounts of time.

The main property that leads this algorithm to be a good candidate for use in domains like natural and multimodal language processing is the capability to infer from positive training examples, without requiring any negative evidence.

4.5 Final Discussion on Learning Methods

As discussed in the previous sections, each CFG grammar inference algorithm has a set of advantages and drawbacks, which can be summarized as shown in Table 4.1.

Table 4.1: Advantages and shortcomings of CFG grammar inference algorithms



	Inductive CYK algorithm	**Version space algorithm**	**e-GRIDS algorithm**
ADVANTAGES	▪ simplicity ▪ short computation time for simple set of rules	▪ good for being used as a general framework for the development of practical, task-specific learning machines	▪ high computational efficiency also for large example sets ▪ capability to infer from positive training examples, without requiring any negative evidence
DRAWBACKS	▪ high computation cost for synthesizing grammars with more than about 14 rules	▪ not immediately applicable to grammar induction, because it produces a set of grammars ▪ inapplicable for large set of training instances	▪ over-generalization

The grammar inference method proposed in this thesis follows the learning from text paradigm. This choice has been motivated by the fact that it is difficult to obtain a set of negative examples for training for NLP and MLP. In the literature, the majority of the learning algorithms applied to NLP, in fact, infers grammars solely from positive examples.

Among the aforementioned algorithms, the e-GRIDS is the only one that learns CFGs from text, without requiring any negative examples. As described in Section 4.4.3, it assumes to have an initial grammar that is obtained by converting each one of the training examples into a grammatical rule. Subsequently, this algorithm applies the learning operators for improving the grammar description in order to make it more accurate and it resolves the over-generalization problem. The core of the e-GRIDS algorithm is then represented by the application of the learning operators, while the production of the initial grammar is not a primary task for the algorithm. An improvement of this algorithm could be achieved by optimizing the production of the initial grammar.

The inductive CYK algorithm, unlike the e-GRIDS one, focuses on the production of a CFG, but it starts from positive and negative sample strings. As this algorithm has the property of being simple and efficient, mainly for simple set of training examples, it has been used in this thesis for producing the initial grammar. As it learns from an informant, however, an adaptation of the algorithm is necessary in order to make it able to learn from positive example only.

Therefore, the grammar inference method, proposed in this dissertation, tries to join together the strengths of the inductive CYK and e-GRIDS algorithms. In particular, a revised version of the inductive CYK algorithm is provided for generating the initial grammar from positive sample sentences, while the e-GRIDS learning operators are taken as starting point for improving the initial grammar description. A detailed description of the proposed grammar inference method is given in Section 5.5.

Chapter 5

The Multimodal Grammar Editor: Theoretical Foundations

This chapter describes the theoretical foundations underlying the proposed multimodal grammar editor. The aim is to allow an easy multimodal grammar specification, overcoming the difficulties arising from the textual description of the grammar production rules (that require the skill of computer programmers and linguistic experts together) and proposing a “by example” approach in order to define a multimodal grammar in a very intuitive way.

5.1 Introduction

Multimodal interfaces can be used in many real-world applications, such as, for example, command and control systems, web and mobile search engines, and information retrieval systems. An overview of four different application scenarios, used as examples throughout the course of this dissertation, is shown in Section 5.2.

The analysis of these scenarios proves not only the advantages of multimodal interfaces in term of usability, naturalness and robustness, but also the difficulties of building a multimodal language.

Consider for example a user saying, “*Show me the phone number of this person*” while pointing at the picture of that person on the display. In the attempt of representing such a multimodal sentence, the first difficulty derives from the multidimensionality of input. In fact, in unimodal languages each input data, which may be words in spoken or written modalities, shapes in gestures, or eye fixations in gaze, etc., can be represented as a stream (i.e. a sequence of tokens), reducing the dimensions from two or three (e.g., drawings or 3D gestures) to a single one. Though each unimodal input is one-dimensional, the combination of multiple streams gives rise to a multidimensional input. Therefore, in multimodal interfaces a technique able to model the multidimensionality of multimodal input is necessary. For instance, in the aforementioned example of sentence, the pointing gesture may be issued before, in-between or after speech. As the sentence's meaning may be different in each of these three cases, the problem of representing the multidimensionality of inputs is very important and requires a preliminary phase of input modeling. This issue is discussed in Section 5.3.

The second difficulty, strictly related to the first one, is about the syntactic structure and semantic meaning representations of multimodal sentences. Although several grammars for natural language have been defined since the 1950s, these grammars can not be used for multimodal input as they are not appropriate to model input symbols from different modalities due to the incompleteness connected with the modality semantics. For instance, considering again the example of the sentence previously given, speech input has an incomplete semantic meaning without considering the pointing gesture too. A well-formed syntactic structure and a complete semantic meaning can only be achieved by integrating inputs from both modalities using an appropriate grammar. Therefore, an extension of the concepts of grammar widely used in Natural Language Processing (NLP) to multimodal grammars is necessary and represents a challenge in order to develop an efficient multimodal language editor. This challenge is addressed in Section 5.4.

Finally, the third difficulty is strictly related to the grammar definition process. In fact, usually grammars are defined by writing a text file containing the grammar syntax rules. This file serves as input for the grammar parser, that acquires the new multimodal sentences and decides whether it belongs to the language generated by the grammar and also defines its structure by building the associated parse tree. Creating or editing a large grammar in textual form is not a simple task, which requires a high skill in computational linguistics. In contrast, designing a grammar "by example", i.e. by inserting the positive sample of multimodal sentences, which the system has to recognize, is much more intuitive and requires less training. This issue and the proposed solution will be discussed in Section 5.5.

5.2 General Discussion on Application Scenarios

Multimodal interfaces have been applied to a broad range of different real-world applications, including, for example, command and control systems, web and mobile search engines, information retrieval systems, and so on.

As the thesis goal is to develop a multimodal grammar editor that is not addressed to a specific task-driven application, but rather able to recognize whichever multimodal expression in whichever task domain, the generality and applicability of the editor to more than one domain is shown (Chapter 7) and some examples of multimodal sentences from these application domains are used throughout the course of all this dissertation.

Therefore, before starting with explaining the theoretical foundations, some real application scenarios, which all the subsequent examples of sentences are referred to, are presented. Note that unimodal inputs can have an incomplete meaning. Only the fusion with the other complementary modalities allows to give a complete sense to the sentence. For instance, the speech input "*Show this in Rome*" has an incomplete meaning if considered alone, while it acquires a complete meaning if joined to the sketching of a river/road.

A driver assistant system

One of the aims of a driver assistance system is to provide navigational assistance to car drivers. Multimodal dialogue interfaces are a support for enhancing interaction between humans and vehicles. In this scenario, for instance, the user can multimodally interact through speech and gesture for knowing information about the car state or the traffic condition, for setting some driving options, for switching lights on or off, for calling emergency or breakdown services, and so on. For instance, a set of multimodal sentences for this scenario include:

S_1: speech: "*Call this emergency service*"
gesture: to indicate the breakdown service on a touch-screen display

S_2: speech: "*Switch on/off this*"
gesture: to point the headlight icon

S_3: speech: "*Set to 22 degrees*"
gesture: to point the temperature icon

S4: speech: "*Search for the traffic condition*"
gesture: to indicate the map area for the search

A multimodal phone book

A multimodal phone book allows the user to communicate with the telephone to perform voice dialing and other phonebook control functions, such as to save or update telephone numbers. In this scenario the user can interact by the synchronized use of speech and handwriting modalities for setting and searching information about the telephone number, address, e-mail, and working company of people and organizations. For example, the user might say "*call this person*" while writing the name of the person on a touch-screen display. This is a scenario that might occur if the user wants to preserve his/her privacy. Other examples of multimodal sentences are the following:

S_1: speech: "*This person works at CNR*"
handwriting: the name of the person on a touch-screen display

S_2: speech: "*The new number of this person is*"
handwriting: the name of the person and the telephone number on a touch-screen display

S_3: speech: "*Give the e-mail of this organization*"
handwriting: the name of the organization on a touch-screen display

A flight timetable system

A flight timetable system provides timetable information of airline companies connections. In this scenario, the user can ask information about flights and companies using speech and pointing gesture. The system interprets a speech input combined with a pointing gesture on a digital map. Examples of acceptable multimodal sentence include:

S_1: speech: *"What company flies here?"*
gesture: to point the location on a map

S_2: speech: *"Does this company fly here?"*
gesture: to point the icon of the company on a touch-screen display and the location on a map

S_3: speech: "*Zoom*" or "*Zoom this*"
gesture: to point the location on a map

S_4: speech: "*I want to take this*"
gesture: to point the icon of the company on a touch-screen display

A map-based information retrieval system

A map-based information retrieval system allows the user to retrieve information using maps and concepts connected with them. The multimodal sentences, which allow to retrieve information are specified using a speech input in combination with a sketch or handwriting input that complete the meaning of the speech sentence. Some examples of mutlimodal sentences in this application scenario are the following:

S_1: speech: *"Show this house near school with garden"*
sketch: a drawing of a house on a touch-screen display

S_2: speech: *"Show this in Rome"*
sketch: a drawing of a river/road on a touch-screen display

S_3: speech: "*Show Italian river*"
handwriting: the word "name" on a touch-screen display

5.3 Multimodal Input Modeling

As mentioned in Chapter 2, most of the input modalities used in human communication can be interpreted as information streams, referring to sequences of data packets or tokens, which may be words, phrases and sentences in spoken or written modalities, shapes in gestures, or eye fixations in gaze, etc. Representing input data as a stream reduces multi-dimensional inputs (e.g., pen-based or 3D gestures) to a single temporal dimension. However, the combination of multiple streams gives rise to multidimensionality.

This multidimensionality is strictly related to the type of cooperation between modalities (see Section 2.2). In particular, following the typology of Martin et al. [MGA01], redundant modalities convey full information alone, without support from the other. As an example, consider when the address of a hotel is described, in parallel, by speech and by a pointing on a map. In this case the problem of representing the multimodal input is quite easy, because each input has a complete meaning and, therefore, can be specified by using the specific unimodal syntax. The problem is less obvious when the modalities are complementary, i.e. they are not just alternative ways to convey the same information but each modality processes different information that contributes to the overall meaning of the sentence. For instance, consider when a user says "*Call this person*" while handwriting his name on a touch-screen display. In this case, a method for combining modalities into a unique multimodal utterance is necessary. This necessity arises from the fact that traditional parsing is sequential and, therefore, the recognition of a sentence occurs only if a linear sequence of tokens is given as input to the parser.

The building of this multimodal sentence requires a solution to the following issues:

- how to represent each unimodal input stream;
- how to linearize different input streams into a unique structure, corresponding to the multimodal sentence that will be used as positive sample in the grammar inference process.

These two issues are described in detail in the following sections.

5.3.1 Representing Unimodal Input

In order to build a meaningful multimodal sentence from each unimodal input, first of all it is necessary to represent each input stream as a sequence of *input elements* (see Figure 5.1). Generally, input elements are separated by periods of inactivity, in which no input signals are detected. The choice of what is represented by an input element depends on the specific application and, in particular, on the level of granularity that is required by the application. For instance, some speech recognizers divide the input signal into utterances according to the period of silence and the prosody information, and assign to each utterance the role of an input element. On the contrary, other speech recognizers regard more consecutive utterances as a meaningful speech unit, giving them the role of input element.

In the proposed unimodal input representation, an input element refers to the basic unit of input that is meaningful to the application and that can be generated from each unimodal input recognizer. For instance, an input element may be a word in spoken or written modalities, a shape in gestures, an eye fixation in gaze, etc.

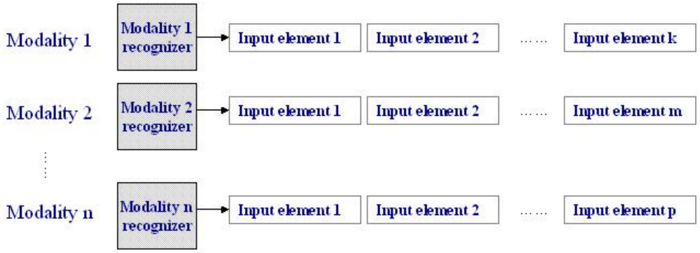


Figure 5.1: The input element representation

In order to determine whether input elements from different modalities have to be considered redundant or complementary and consequently decide how to group them according to the cooperation classes between modalities for building a unique multimodal sentence, the knowledge of the following information is necessary, for each input element, as depicted in Figure 5.2:

- the value of the element, which has to belong to the vocabularies of specific input recognition systems,
- the modality used to express the input element,
- the syntactic role that the element has inside the unimodal sentence. The standard nomenclature of the Penn treebank [MSM94] is used to represent the syntactic categories. This is composed by 45 syntactic categories, some of which are summarized in Figure 5.3.
- the modalities cooperation class, which represent the inter-modality relationships.

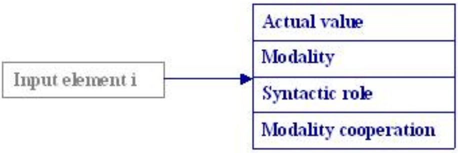
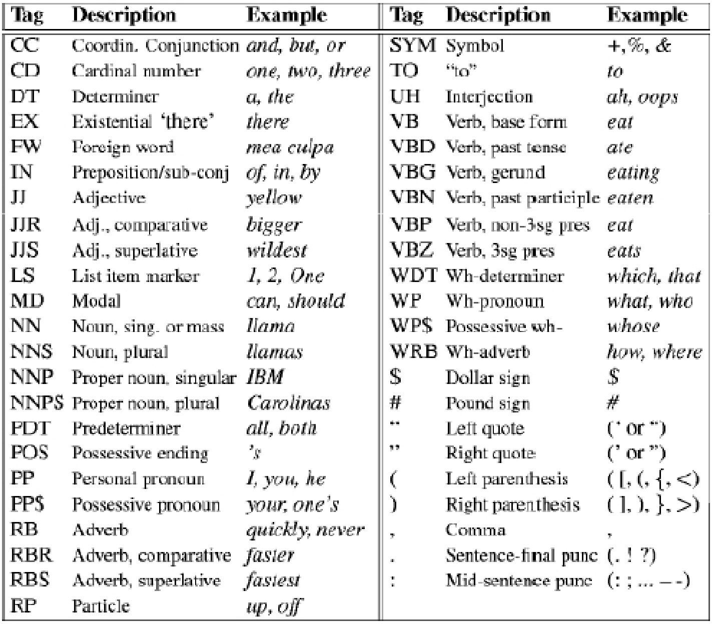


Figure 5.2: The set of attributes of input elements

Moreover, in order to decide what kind of interaction/dialogue the user wants to have with the system, the information about the category of multimodal sentence is necessary.

In natural language, sentences can be classified in: simple, compound, complex and compound-complex. Simple sentence consists of a single independent clause with one subject and one verb. Compound sentence has two or more clauses related by coordinating conjunctions. Complex sentence consists of at least one independent clause and one or more subordinating clauses related by subordinating conjunctions. Finally, compound-complex sentence join together compound and complex sentences. Considering the complexity of managing complex and compound sentences, in this thesis only simple sentences will be considered. The management of more complex sentences represents a future work.



Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	*#*
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*'s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Figure 5.3: Penn treebank syntactic categories

Another sentence classification in natural language is based on the sentence meaning. Following this criterion, sentences can be broadly classified as: declarative (statement), imperative (command), interrogative (question), and exclamatory. From the analysis of the human behavior in the aforementioned application scenarios, the following categories can be envisaged in a multimodal human-computer communication:

- Multimodal Question (MQ)

▪ *"Wh question"*	Ex: Who is this?
▪ *Interrogative form*	Ex: Does this person work in this company?

- Multimodal Command (MC)

▪ *Imperative*	Ex: Call
▪ *Imperative adverb command*	Ex: Show people that work

here

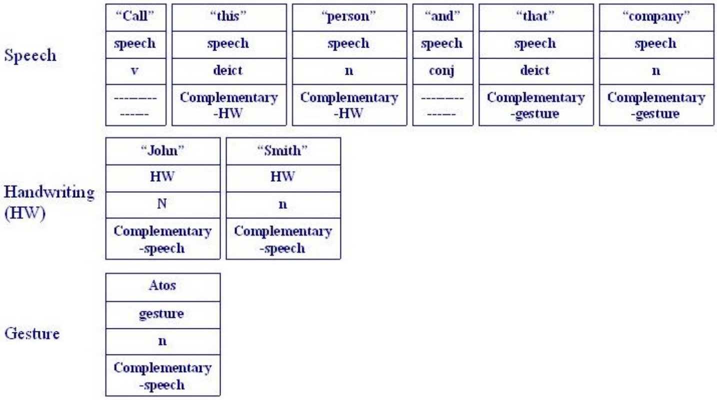
- *Imperative pronoun command* Ex: Call this
- *Imperative noun command* Ex: Modify the number of this person

- Multimodal Statement (MS)

- *Declarative sentence* Ex: This is the company of John Smith
- *Demonstrative noun sentence* Ex: This person works at CNR
- *Demonstrative pronoun sentence* Ex: The number of John Smith is this
- *Demonstrative adverb sentence* Ex: John Smith works here
- *Negative sentence* Ex: John Smith does not work here

Identifying the category of a multimodal sentence allows to determine its structure and the expected action. For instance, if the sentence belongs to the "imperative noun command" category, it should contain an imperative verb, which corresponds to the action expected by the user, and a demonstrative noun phrase, that represents the deictic expression. Moreover, a sentence belonging to the "wh question" category should contains an interrogative pronoun that begins with wh- (e.g., who, when, what,etc.).

As an example, consider the multimodal sentence composed of the speech "call this person and that company", by the handwriting of the person's name and by the pointing gesture on the company icon on a touch-screen display. The sequence of input elements of each one of the three modalities and their associated sets of attributes is depicted in Figure 5.4. Moreover, the category of this multimodal sentence is "imperative noun command".



Speech	“Call”	“this”	“person”	“and”	“that”	“company”
	speech	speech	speech	speech	speech	speech
	v	deict	n	conj	deict	n
	-------- ------	Complementary -HW	Complementary -HW	-------- ------	Complementary -gesture	Complementary -gesture
Handwriting (HW)	“John”	“Smith”				
	HW	HW				
	N	n				
	Complementary -speech	Complementary -speech				
Gesture	Atos					
	gesture					
	n					
	Complementary -speech					

Figure 5.4: The input element representation for the example

All these information, about input elements and the sentence category, will be used during the linearization process, as shown in the following section. Moreover, the input elements, as defined above, will constitute the set of terminal symbols of the multimodal grammar, which will be described in Section 5.4.

5.3.2 The Linearization Process

Starting from the input elements, described as shown in the previous section, the linearization process aims at combining these elements, grouping them opportunely, in order to generate a linear sequence of elements, which represents the multimodal sentence that will be used as positive sample during the grammar inference process. The linearization takes place according to modality cooperation and syntactic roles defined in the previous stage.

In particular, modality cooperation allows to determine whether input elements convey information that have some relations with the information conveyed by the other elements. Generally, cooperative input modalities are close together in time. Although temporal proximity is a quite easy and application-independent criterion, it does not take into account semantic aspects of input sentences, producing sometimes a linear sequence of elements that does not make sense. Moreover, forcing the user to express a multimodal sentence according to pre-defined constraints imposed by temporal links compromises the flexibility that has to be one of the main features of a multimodal language. Therefore, in a multimodal dialogue the knowledge of how two or more modalities cooperate each another represents a more meaningful information than pre-defined temporal links among input elements, as this information do not force the user to synchronize the input elements of a multimodal sentence, capturing likewise the inter-modality relations.

Information about modality cooperation is used during the linearization process in conjunction with other criteria that take into account the syntactic role of each input element. In particular, if two input elements, coming from different modalities, have the same syntactic role, they can be considered close together in syntax. This syntactic proximity is an easy criterion that reinforces the information about modality cooperation, making the linearization output more accurate.

Finally, the linearization process makes use of the sentence category, defined during the previous stage, as final criterion for determining which is the linear sequence of input elements that will be the positive sample for the grammar inference process. In particular, the sentence category allows to understand if a demonstrative (or deictic) expression should be detected into the sentence and whether this expression is expressed through a noun, pronoun or adverb phrase. These information allow to add further constraints on syntactic proximity.

Let us continue the illustrative example introduced in the previous section (see Figure 5.4). According to the classes of modality cooperation, the combination of input elements into a unique multimodal utterance can be given in several different ways. Figure 5.5 shows the cooperative relations for the input elements of the example (the highlighted rectangles include the elements involved in the cooperation). Any combination of input elements, which respects the unimodal input order, is acceptable inside the highlighted rectangles.

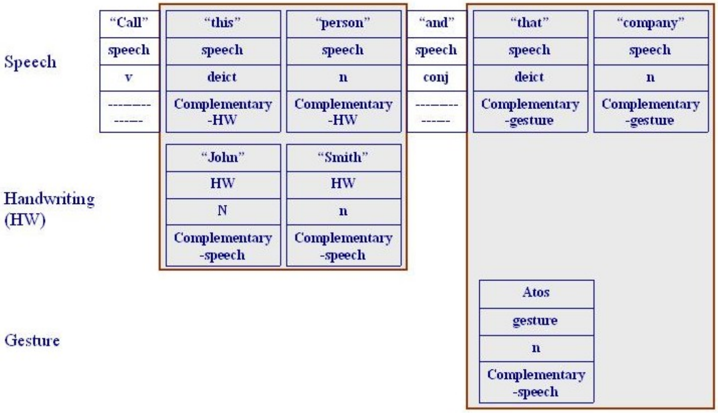
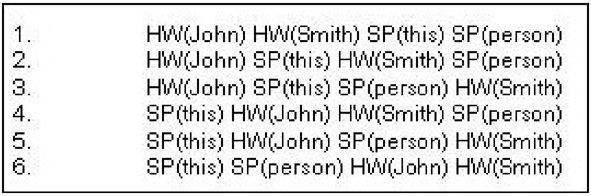


Figure 5.5: Cooperative relations of input elements in the example

As in the first rectangle (on the left of Figure 5.5) there are four input elements, six ways of combining them exist, as shown in Table 5.1. For each element of the sentence the modality (HW=handwriting, SP=speech, G=gesture) and the actual value of the element is expressed. For instance, the first sentence is the linear sequence composed by the handwriting elements "*John*" and "*Smith*" followed by the speech elements "*this*" and "*person*". The second sentence is composed by the handwriting element "*John*", followed by the speech element "*this*", followed by the handwriting element "*Smith*", and ended by the speech element "*person*", and so on for all the sequences.

Table 5.1: Linear sentences for the example


1. HW(John) HW(Smith) SP(this) SP(person)
2. HW(John) SP(this) HW(Smith) SP(person)
3. HW(John) SP(this) SP(person) HW(Smith)
4. SP(this) HW(John) HW(Smith) SP(person)
5. SP(this) HW(John) SP(person) HW(Smith)
6. SP(this) SP(person) HW(John) HW(Smith)

In the second highlighted rectangle (on the right of Figure 5.5) three input elements occur, giving rise to three different linearized sentences:

1. SP(that) SP(company) G(Atos)
2. SP(that) G(Atos) SP(company)
3. G(Atos) SP(that) SP(company)

Therefore, the overall sentence can be combined in eighteen different ways, such as for example:

1. SP(Call) HW(John) HW(Smith) SP(this) SP(person) SP(and) SP(that) SP(company) G(Atos)
2. SP(Call) HW(John) SP(this) HW(Smith) SP(person) SP(and) SP(that) SP(company) G(Atos)
3. SP(Call) HW(John) SP(this) SP(person) HW(Smith) SP(and) SP(that) SP(company) G(Atos)

   … … … … … … … … … … … …

18. SP(Call) SP(this) SP(person) HW(John) HW(Smith) SP(and) G(Atos) SP(that) SP(company)

By applying the syntactic proximity criterion, a reduction of the number of acceptable linearized sentences occurs. In fact, only sentences, in which input elements with the same syntactic role are close together, are acceptable. Figure 5.6 shows syntactic proximity of input elements in the example (the rectangles highlight the elements involved in the syntactic proximity). In the first interval the following three sentences will be considered:

1. SP(this) HW(John) HW(Smith) SP(person)
2. SP(this) HW(John) SP(person) HW(Smith)
3. SP(this) SP(person) HW(John) HW(Smith)

while in the second interval these one:

1. SP(that) SP(company) G(Atos)
2. SP(that) G(Atos) SP(company)

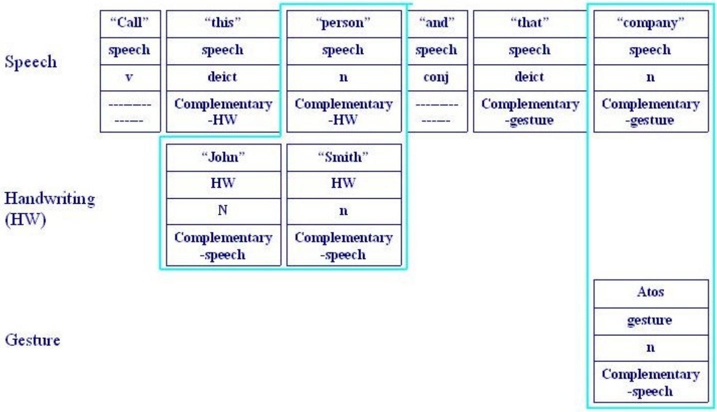


Figure 5.6: Syntactic proximity of input elements in the example

Therefore, the number of acceptable linearized sentences is decreased from eighteen to six, that are:

1. SP(Call) SP(this) HW(John) HW(Smith) SP(person) SP(and) SP(that) SP(company) G(Atos)
2. SP(Call) SP(this) HW(John) SP(person) HW(Smith) SP(and) SP(that) SP(company) G(Atos)
3. SP(Call) SP(this) SP(person) HW(John) HW(Smith) SP(and) SP(that) SP(company) G(Atos)
4. SP(Call) SP(this) HW(John) HW(Smith) SP(person) SP(and) SP(that) G(Atos) SP(company)
5. SP(Call) SP(this) HW(John) SP(person) HW(Smith) SP(and) SP(that) G(Atos) SP(company)
6. SP(Call) SP(this) SP(person) HW(John) HW(Smith) SP(and) SP(that) G(Atos) SP(company)

Finally, taking into account the information about sentence category, it is possible to detect whereas a demonstrative expression is defined into the sentence and which is its syntactic category (noun, pronoun or adverb sentence). In the example, the sentence category is an imperative noun command. This means that it contains a demonstrative expression that is used as noun phrase. Therefore, the deictic word "*this*" (or "*that*") has to be followed by a noun, that implies the selection of the third sentence (among the above six sentences), that is:

SP(Call) SP(this) SP(person) HW(John) HW(Smith) SP(and) SP(that) SP(company) G(Atos)

In summary, the multimodal input model presented here allows to represent each input stream as a set of input elements, and to linearize different input streams into a unique multimodal sequence, that will be used as positive sample for the inference of the multimodal grammar.

5.4 The Multimodal Attribute Grammar

Designing and developing a grammar editor for a multimodal language processor that is not addressed to a specific task-driven application, but rather able to recognize whichever multimodal expression, is a focal question. The editor is intended to be used by expert and/or non-expert users who want to specify the multimodal language that has to be recognized by the system.

As a language can be formally described through a grammar, the specification of a multimodal language requires the definition of a multimodal grammar. The most popular kind of grammar, extensively used in natural language processing and frequently adapted to represent multimodal languages, is the context-free grammar (CFG), previously introduced in Section 3.3.1.

For defining the multimodal grammar proposed in this dissertation, the context-free paradigm has been followed due to its ability to model all frequent linguistic sentences of multimodal language by assuring, at the same time, a lower parsing complexity. However, in order to use CFG for multimodal language processing it is necessary to overcome the two main deficiencies of this grammatical formalism, i.e. the lack of constructions both for representing input symbols from different modalities and for modeling semantic and temporal aspects of input symbols.

In this attempt, attribute grammars provide a good compromise between the context-free paradigm and the necessity to represent semantic and temporal aspects of multimodal input.

Attribute grammars [Knu68] were firstly developed by Donald Knuth as a means of formalizing the semantics of a context-free language. They may be informally defined as a context-free grammar that has been extended to provide context sensitivity using a set of attributes (associated with each distinct symbol in the grammar), assignment of attribute values, evaluation rules, and conditions.

Starting from the attribute grammar formalism, an extension of this notation for multimodal input processing is necessary. Therefore, the Multimodal Attribute Grammar (MAG) has been introduced, whose formal definition is given below.

Definition 5.1. *A Multimodal Attribute Grammar is a triple*

$$G = (\mathcal{G}, \mathcal{A}, \mathcal{R})$$

where:

(1) $\mathcal{G}$ is a context-free grammar (T,N,P,S) with T as set of terminal symbols, N as set of non-terminal symbols, P as set of production rules of the form:

$$X_0 \rightarrow X_1 X_2 \ldots X_n \qquad \text{where } n \geq 1,\ X_0 \in N \text{ and } X_k \in N \cup T \text{ for } 1 \leq k \leq n$$

and $S \in N$ as start symbol (or axiom)

(2) $\mathcal{A}$ is a collection $(A(X))_{X \in N \cup T}$ of attributes of the non-terminal and terminal symbols, such that for each $X \in N \cup T$, A(X) is split in two finite disjoint subsets I(X), the set of inherited attributes of X, and S(X), the set of synthesized attributes. The set S(X) with $X \in T$ includes a set of attributes MS(X), called set of multimodal synthesized attributes, composed of the following four attributes:

$$MS(X)=\{val, mod, synrole, coop\}$$

(3) $\mathcal{R}$ is a collection $(R_p)_{p \in P}$ of semantic functions (or rules). ■

A derivation tree for a sentence in a context-free language has the property that each of its leaf nodes is labeled with a symbol from *T* and each interior node *t* corresponds to a production $p \in P$ such that *t* is labeled with X_0 and *t* has *n* children labeled with $X_1, X_2, \ldots, X_n$ in left-to-right order, as shown in Figure 5.7.

The set of production rules *P* can be partitioned in two disjoint subsets, the set of background rules P_B and the set of target rules

P_T. The former is composed of rules that are directly derived from the background knowledge, i.e. they contain terminal symbols only in the body. The latter is composed of rules that are derived from the background rules, i.e. they contain at least one non-terminal symbol in the body. Therefore, background rules are of the form p_b: $X_0 \rightarrow A$, where $X_0 \in N$ and $A \in T$, while target rules are of the form p_t: $X_0 \rightarrow X_1...X_m$, where $X_0 \in N$ and $X_i \in N \cup T$.

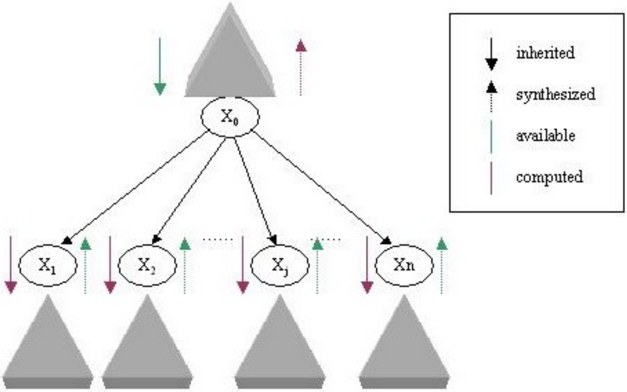


Figure 5.7: Information flow in the attribute grammar notation

The attributes of the set *MS(X)* are domain independent and are necessary for managing the multimodal properties of a sentence. As discussed in Section 5.3.1, each input element of a multimodal sentence, corresponding to a terminal symbol of the grammar, contains information about the actual value of the element, the modality, the syntactic role, and the modality cooperation. These information are contained into the four attributes of *MS(X)*. Precisely, the *val* attribute expresses the current value (concept) of the terminal symbol, the *mod* attribute represents the modality, the *synrole* attribute conveys information about the syntactic role, and finally the *coop* attribute expresses the modality cooperation with other terminal symbols. In particular, the domains of these attributes are the following:

D_{val} = T

D_{mod} = {speech, handwriting, gesture, sketch}

$D_{synrole}$ = {noun phrase, verb phrase, determiner, verb, noun, adjective, preposition, deictic}

D_{coop} = {complementary, redundant}

The set R_p of semantic functions associated with each production rule in *P* allows to compute the values of the inherited and synthesized attributes. In order to explain how these functions are defined the following definition has to be provided.

***Definition* 5.2.** *Let p be a production rule in P of the form*

$$p: X_0 \rightarrow X_1\, X_2 \ldots X_n$$

For each i=0,…,n and each a in $A(X_i)$, the notation $X_i.a$ denotes the occurrence of the attribute a of the symbol X_i in p. The set of all such occurrences is denoted Occ(p) and is called the set of occurrences of p. ■

Given a production rule *p*, the set *Occ(p)* is composed of two finite and disjoint sets, named *Input(p)* and *Output(p)*. The former contains the occurrences of the attributes in *p* that are available from the context, i.e. the inherited attributes for the head of the rule *p* and the synthesized attributes for the symbols in the body of *p*. The latter contains the occurrences of the attributes in *p* that have to be computed using the semantic functions, i.e. the synthesized attributes for the head of the rule *p* and the inherited attributes for the symbols in the body of *p*. This is expressed in the following lemma.

***Lemma* 5.1.** *The set of occurrences of a production rule $p: X_0 \rightarrow X_1\, X_2 \ldots X_n$ is given by*

$$Occ(p) = Input(p) \cup Output(p)$$

where

$$Input(p) = \{X_i.a \mid X_i \in I(X_0) \text{ or } X_i \in S(X_i) \text{ for } i>0\} \quad \text{and}$$

$$Output(p) = \{X_i.a \mid X_i \in S(X_0) \text{ or } X_i \in I(X_i) \text{ for } i>0\}$$ ■

Whether the production rule *p* contains any terminal symbol X_i in the body, then the set *Input(p)* contains also the occurrences of the attributes of X_i in $MS(X_i)$ as they are included in the synthesized attributes $S(X_i)$.

The attribute occurrences $X_i.a \in Input(p)$ take a value from some semantic domain (such as integers, strings of characters, or structures of some type) that is given by example. The attribute occurrences $X_i.b \in Output(p)$ take a value that has to be evaluated by semantic functions. Therefore, a finite set R_p of semantic functions is associated with the production p, with exactly one function for each attribute occurrence $X_i.b \in Output(p)$. Each semantic function in R_p is composed of an assignment statement. The left side of each assignment statement is an occurrence $X_i.b \in Output(p)$ while the right side contains an expression (in some predefined logical language) with variables in ($Occ(p) - X_i.b$). Formally, the definition of a semantic function can be given as follows.

Definition 5.3. *Let p: $X_0 \rightarrow X_1 X_2 \ldots X_n$ be a production rule in P. The set R_p is composed of semantic functions that have the form*

$$X_i.b \leftarrow f(y_1, \ldots, y_k) \qquad \text{with } k \geq 1$$

where

1. $X_i.b \in Output(p)$;

2. y_j, with $1 \leq j \leq k$, is an occurrence in $(Occ(p) - X_i.b)$;

3. f is a function that maps the values of $y_1, \ldots, y_k$ to the value of $X_i.b$. ■

Analogously to the set of production rules *P,* the collection $\mathcal{R}$ of semantic functions can be partitioned in two disjoint subsets, the set of background functions R_B associated with the production rules belonging to P_B and the set of target functions R_T associated with the production rules belonging to P_T.

Example 5.1. Consider again the multimodal sentence composed of the input elements shown in Figure 5.4. The consequent linearized sentence, that has been evaluated in Section 5.3.2, has the form:

SP(Call) SP(this) SP(person) HW(John) HW(Smith) SP(and) SP(that) SP(company) G(Atos).

The multimodal attribute grammar that is able to generate this sentence is written as follows.

- Terminal symbols: T={*Call, This, Person, John, Smith, And, That, Company, Atos*}

- Non-terminal symbols: N={*Sentence, VP, VB, NN, DT, NP, NNP, CC, NNS*}

- Start symbol: S

- Attributes: A(X) = I(X) ∪ S(X) where

I(X) = ∅

S(X) = {*val, mod, synrole, coop*} so that

- *S(NN)=S(DT)=S(NNP)=S(CC)=S(NNS)=S(VB)=*{*val, mod, synrole, coop*} and
- *S(Sentence)=S(VP)=S(NP)= {val, mod}*

- Production rules and semantic functions:

P1) S → VP Sentence
R1.1) S.val ← VP.val + Sentence.val
R1.2) S.mod ← VP.mod + Sentence.mod
P2) VP → VB
R2.1) VP.val ← VB.val
R2.2) VP.mod ← VB.mod
P3) Sentence → NP CC NP
R3.1) Sentence.val ← NP.val + CC.val + NP.val
R3.2) Sentence.mod ← NP.mod + NP.mod
P4) NP → DT NN NNP1 NNP2
R4.1) NP.val ← NNP1.val + NNP2.val
R4.2) NP.mod ← DT.mod + NNP1.mod
P5) NP → DT NN NNS
R5.1) NP.val ← NNS.val
R5.2) NP.mod ← DT.mod + NNS.mod
P6) NN → Person
R6.1) NN.val ← person
R6.2) NN.mod ← speech
R6.3) NN.synrole ← noun

R6.4) NN.coop ← complementary

P7) NN → Company

R7.1) NN.val ← company

R7.2) NN.mod ← speech

R7.3) NN.synrole ← noun

R7.4) NN.coop ← complementary

P8) DT → This

R8.1) DT.val ← this

R8.2) DT.mod ← speech

R8.3) DT.synrole ← deictic

R8.4) DT.coop ← complementary

P9) DT → That

R9.1) DT.val ← that

R9.2) DT.mod ← speech

R9.3) DT.synrole ← deictic

R9.4) DT.coop ← complementary

P10) NNP1 → John

R10.1) NNP1.val ← John

R10.2) NNP1.mod ← handwriting

R10.3) NNP1.synrole ← noun

R10.4) NNP1.coop ← complementary

P11) NNP2 → Smith

R11.1) NNP2.val ← Smith

R11.2) NNP2.mod ← handwriting

R11.3) NNP2.synrole ← noun

R11.4) NNP2.coop ← complementary

P12) NNS → Atos

R12.1) NNS.val ← Atos

R12.2) NNS.mod ← gesture

R12.3) NNS.synrole ← noun

R12.4) NNS.coop ← complementary

P13) VB → Call

R13.1) VB.val ← call

R13.2) VB.mod ← speech

R13.3) VB.synrole ← verb

P14) CC → And

R14.1) CC.val ← and

R14.2) CC.mod ← speech
R14.3) CC.synrole ← conjunction

The set of background rules is P_B = {P6, …, P14} while the set of target rules is P_T = {P1, P2, P3, P4, P5}.

The background functions R_B, associated with rules in P_B, assign the values of the attributes of terminal symbols to the attributes of the corresponding syntactic categories. For instance, Function R14.1 assigns the value "*and*" to the attribute *val* of the syntactic category CC, while Function R14.2 assigns the value *speech* to the attribute *mod* of the syntactic category CC, and so on for all the background functions.

The target functions R_T, associated with rules in P_T, map the opportunely combined values of the attributes of non-terminal symbols in the body of the rules into the attributes of the non-terminal symbols in the head. For example, Function R4.1 assign to the attribute *val* of non-terminal symbol NP the value obtained by combining the values of the attribute *val* of the non-terminal NPP1 and NPP2 (the + operator produces the sequence of terminal symbols involved in the operation), while Function R4.2 assign to the attribute *mod* of non-terminal symbol NP the value obtained by combining the values of the attribute *mod* of the non-terminal DT and NPP1 (+ represent the operation of union between two modalities), and so on for all the target functions.

The derivational tree of the multimodal sentence in the Example 5.1 is shown in Figure 5.8.

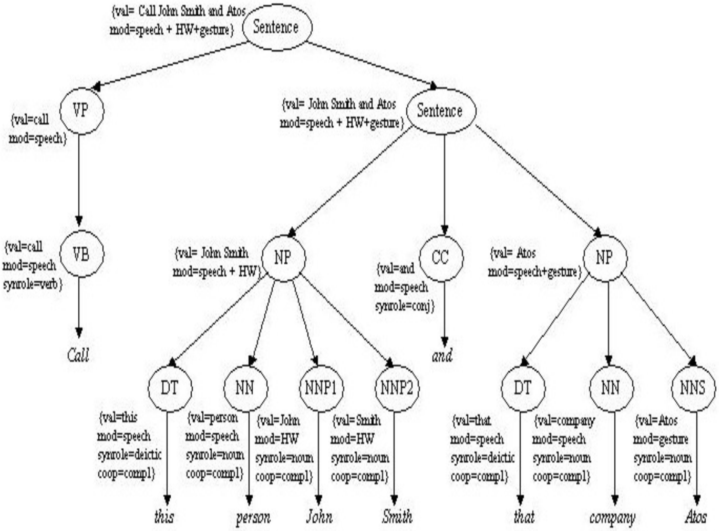


Figure 5.8: The derivational tree of the sentence in Example 4.1

5.5 The Grammar Inference Algorithm

Given the set of multimodal linearized sentences and the multimodal attribute grammar formalism, both described in the previous paragraphs, the next step in developing a multimodal language editor is to define a computationally efficient algorithm for grammatical inference that takes as input the set of linearized sentences (containing positive examples only) and generates the MAG production rules and the associated semantic functions to parse those examples.

This section describes such an algorithm that relies on the inductive CYK (Cocke-Younger-Kasami [Kas65]) algorithm and the e-GRIDS [PPK04] learning operators. In particular, a revised version of the inductive CYK algorithm is provided for generating the initial grammar from positive sample sentences, while the e-GRIDS learning operators are taken as starting point for improving the initial grammar description. The choice of the CYK algorithm has been led by its simplicity and efficiency, mainly for simple set of training examples, while the e-GRIDS learning operators are able to improve the grammar description making it more accurate and it resolves the over-generalization problem. Therefore, the proposed grammar inference method tries to join together the strengths of the inductive CYK and e-GRIDS algorithms, adapting them to multimodal input. In particular, this method consists of two main steps (see Figure 5.9): the first step includes a revised version of the inductive CYK algorithm for generating the multimodal attribute grammar that is able to parse the input sentence; the second step makes use of the e-GRIDS operators for improving the grammar description coming from the first step and avoiding the over-generalization problem.

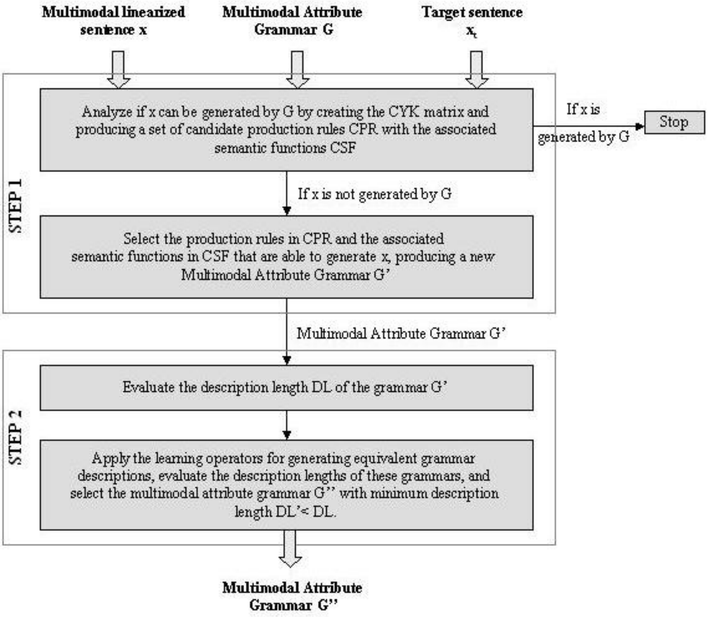

Multimodal linearized sentence x
Multimodal Attribute Grammar G
Target sentence x_t
STEP 1
Analyze if x can be generated by G by creating the CYK matrix and producing a set of candidate production rules CPR with the associated semantic functions CSF
If x is generated by G
Stop
If x is not generated by G
Select the production rules in CPR and the associated semantic functions in CSF that are able to generate x, producing a new Multimodal Attribute Grammar G'
Multimodal Attribute Grammar G'
STEP 2
Evaluate the description length DL of the grammar G'
Apply the learning operators for generating equivalent grammar descriptions, evaluate the description lengths of these grammars, and select the multimodal attribute grammar G'' with minimum description length DL'< DL.
Multimodal Attribute Grammar G''

Figure 5.9: Workflow of the proposed grammar inference algorithm

These two steps are described in detail in the following sections. In particular, next section illustrates the first step of the algorithm, which generates the MAG production rules and the associated semantic functions to parse positive examples, while Section 5.5.2

presents the second step, which applies the learning operators and provides a measurement of the accuracy of the grammar for improve the grammar description.

5.5.1 First Step: the MAG Generation from Positive Examples

The proposed grammatical inference algorithm enhances the inductive CYK algorithm along two main dimensions. The greatest disadvantage of the application of the inductive CYK algorithm to MLP is associated with the fact that it learns from an informant (see Section 4.4.1). In fact, it is difficult to obtain a set of negative examples in NLP and MLP. To be more precise, the user is unable to specify all sentences that have not to be included into the grammar, while s/he can easily define sentences that have to be included. In the literature, the majority of the learning algorithms applied to NLP, in fact, infers grammars solely from positive examples. Therefore, an adaptation of inductive CYK algorithm is necessary in order to learn from positive examples only. However, this requires the introduction in the second step of some heuristics to avoid the over-generalization problem. Another significant improvement relates to handle Multimodal Attribute Grammars rather than CFGs. Therefore, not only the set of production rules has to be generated by the algorithm, but also the collection of associated semantic functions for evaluating the attribute values.

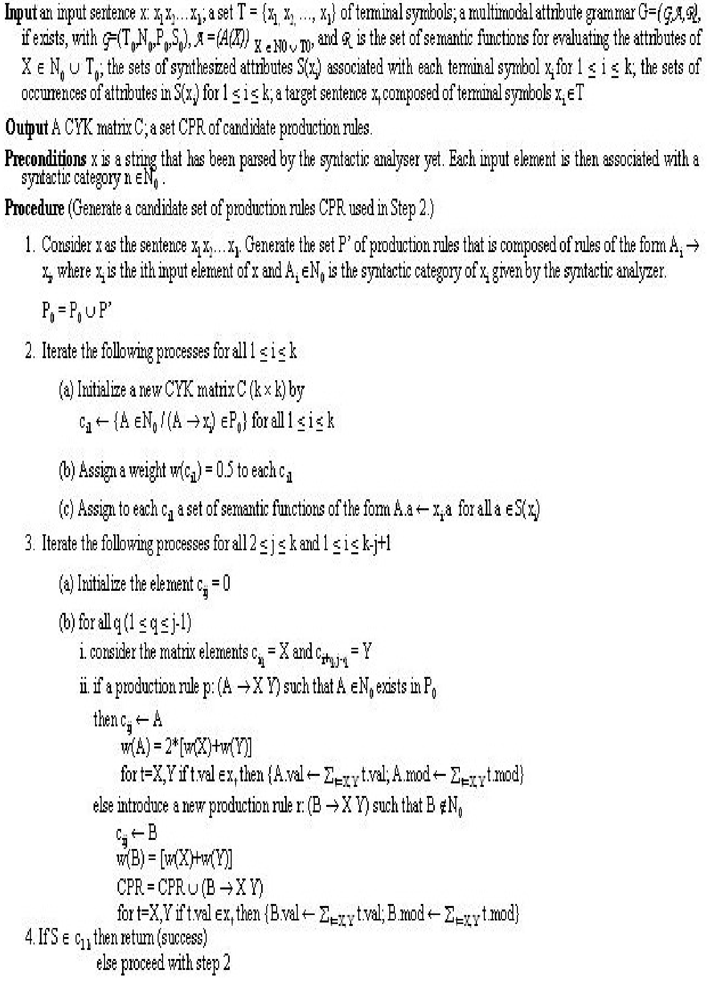
For the sake of convenience, the revised CYK algorithm has been split in two consecutive steps, whose detailed descriptions are given in Figure 5.10 and 5.11, respectively.

The first step takes as input:

- the linearized sentence $x= x_1x_2...x_k$, with $x_i \in T$;
- the current multimodal grammar G=$(\mathcal{G},\mathcal{A},\mathcal{R})$, if exists, with $\mathcal{G}$=(T_0,N_0,P_0,S_0), $\mathcal{A} =(A(X))_{X \in N0 \cup T0}$, and $\mathcal{R}$ is the set of semantic functions for evaluating the attributes of $X \in N_0 \cup T_0$;
- the sets of synthesized attributes $S(x_i)$ associated with each terminal symbol x_i;
- the sets of occurrences of attributes in $S(x_i)$ for $1 \leq i \leq k$;

- a target sentence x_t composed of terminal symbols $x_i \in T$.The output is a CYK matrix, in which each position c_{ij} contains the non-terminals, the associated weights and the semantic functions for evaluating the attribute occurrences of the non-terminals. At the same time, a structure that contains all the candidate production rules (with the associated weights) generated by the algorithm is created and updated. The first step ends whether the sentence *x* is generated by the current grammar. Otherwise the algorithm proceeds with the second step.

During this step the analysis of the structures generated during the first step, that are the CYK matrix C and set CPR of candidate production rules is performed. In particular, the algorithm selects the candidate derivations with the highest weights. Non-terminal symbols, which belong to the set N_0, do not need any processing, while those symbols that are created during the first phase for simulating the generation of some productions, need to be definitely included into the grammar. Consequently, non-terminals that make part of the production rule inserted into the grammar, have to be re-defined until all symbols belong to the grammar. Therefore, the output of the revised CYK algorithm is a new multimodal attribute grammar G’=$(\mathcal{G}',\mathcal{A}',\mathcal{R}')$, where $\mathcal{G}'$=(T’,N’,P’,S’), $\mathcal{A}' = (A(X))_{X \in N \cup T}$, and $\mathcal{R}$ is the set of semantic functions for evaluating the attributes of $X \in N \cup T$.



Input an input sentence x: $x_1 x_2 \ldots x_1$; a set $T = \{x_1, x_2, \ldots, x_1\}$ of terminal symbols; a multimodal attribute grammar $G=(\mathcal{G},\mathcal{A},\mathcal{R})$, if exists, with $\mathcal{G}=(T_0,N_0,P_0,S_0)$, $\mathcal{A} = (A(X))_{X \in N0 \cup T0}$, and $\mathcal{R}$ is the set of semantic functions for evaluating the attributes of $X \in N_0 \cup T_0$; the sets of synthesized attributes $S(x_i)$ associated with each terminal symbol x_i for $1 \leq i \leq k$; the sets of occurrences of attributes in $S(x_i)$ for $1 \leq i \leq k$; a target sentence x_t composed of terminal symbols $x_i \in T$

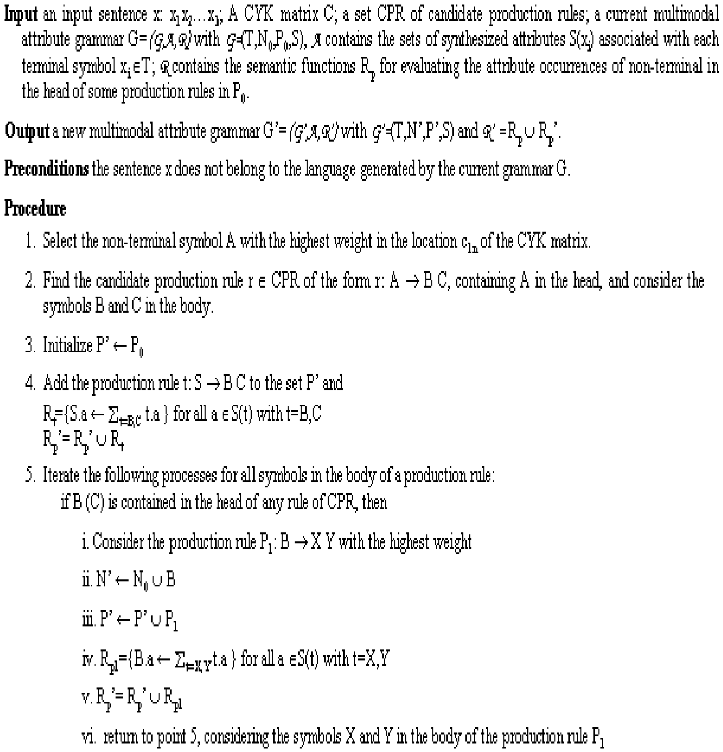
Output A CYK matrix C; a set CPR of candidate production rules.

Preconditions x is a string that has been parsed by the syntactic analyser yet. Each input element is then associated with a syntactic category $n \in N_0$.

Procedure (Generate a candidate set of production rules CPR used in Step 2.)

```
1. Consider x as the sentence $x_1 x_2 \ldots x_1$. Generate the set P' of production rules that is composed of rules of the form $A_i \rightarrow x_i$, where $x_i$ is the ith input element of x and $A_i \in N_0$ is the syntactic category of $x_i$ given by the syntactic analyzer.
   $P_0 = P_0 \cup P'$
2. Iterate the following processes for all $1 \leq i \leq k$
   (a) Initialize a new CYK matrix C ($k \times k$) by
       $c_{i1} \leftarrow \{A \in N_0 / (A \rightarrow x_i) \in P_0\}$ for all $1 \leq i \leq k$
   (b) Assign a weight $w(c_{i1}) = 0.5$ to each $c_{i1}$
   (c) Assign to each $c_{i1}$ a set of semantic functions of the form $A.a \leftarrow x_i.a$ for all $a \in S(x_i)$
3. Iterate the following processes for all $2 \leq j \leq k$ and $1 \leq i \leq k-j+1$
   (a) Initialize the element $c_{ij} = 0$
   (b) for all q ($1 \leq q \leq j-1$)
       i. consider the matrix elements $c_{iq} = X$ and $c_{i+q,j-q} = Y$
       ii. if a production rule p: ($A \rightarrow X\ Y$) such that $A \in N_0$ exists in $P_0$
         then $c_{ij} \leftarrow A$
              $w(A) = 2*[w(X)+w(Y)]$
              for t=X,Y if $t.val \in x_t$ then $\{A.val \leftarrow \Sigma_{t=X,Y} t.val; A.mod \leftarrow \Sigma_{t=X,Y} t.mod\}$
         else introduce a new production rule r: ($B \rightarrow X\ Y$) such that $B \notin N_0$
              $c_{ij} \leftarrow B$
              $w(B) = [w(X)+w(Y)]$
              $CPR = CPR \cup (B \rightarrow X\ Y)$
              for t=X,Y if $t.val \in x_t$ then $\{B.val \leftarrow \Sigma_{t=X,Y} t.val; B.mod \leftarrow \Sigma_{t=X,Y} t.mod\}$
4. If $S \in c_{11}$ then return (success)
         else proceed with step 2
```

Figure 5.10: First step of the revised CYK algorithm



Input an input sentence x: $x_1 x_2 \ldots x_j$; A CYK matrix C; a set CPR of candidate production rules; a current multimodal attribute grammar G=$(\mathcal{G},\mathcal{A},\mathcal{R})$ with $\mathcal{G}$=(T,N_0,P_0,S), $\mathcal{A}$ contains the sets of synthesized attributes S(x_j) associated with each terminal symbol $x_i \in$T; $\mathcal{R}$ contains the semantic functions R_p for evaluating the attribute occurrences of non-terminal in the head of some production rules in P_0.

Output a new multimodal attribute grammar G'= $(\mathcal{G}',\mathcal{A},\mathcal{R}')$ with $\mathcal{G}'$=(T,N',P',S) and $\mathcal{R}'$ =$R_p \cup R_p$'.

Preconditions the sentence x does not belong to the language generated by the current grammar G.

Procedure

```
1. Select the non-terminal symbol A with the highest weight in the location $c_{1n}$ of the CYK matrix.
2. Find the candidate production rule r ∈ CPR of the form r: A → B C, containing A in the head, and consider the symbols B and C in the body.
3. Initialize P' ← $P_0$
4. Add the production rule t: S → B C to the set P' and
   $R_t$={S.a ← $\Sigma_{t=B,C}$ t.a } for all a ∈S(t) with t=B,C
   $R_p$'= $R_p$' ∪ $R_t$
5. Iterate the following processes for all symbols in the body of a production rule:
     if B (C) is contained in the head of any rule of CPR, then
         i. Consider the production rule $P_1$: B → X Y with the highest weight
         ii. N' ← $N_0$ ∪ B
         iii. P' ← P' ∪ $P_1$
         iv. $R_{p1}$={B.a ← $\Sigma_{t=X,Y}$t.a } for all a ∈S(t) with t=X,Y
         v. $R_p$'= $R_p$' ∪ $R_{p1}$
         vi. return to point 5, considering the symbols X and Y in the body of the production rule $P_1$
```

Figure 5.11: Second step of the revised CYK algorithm

More in details, the first step of the revised CYK algorithm (shown in Figure 5.10) works in the following way. Assume we have a multimodal linearized sentence *x*, that is composed of input elements x_i, for $1 \le i \le k$ that are the terminal symbols of the grammar. Each terminal symbol is associated with a syntactic category, which corresponds to a non-terminal symbol of the set N_0. If *x* is not the first multimodal sentence inputted to the editor, a multimodal attribute grammar G exists yet and it is given as input to the algorithm. This initial grammar G is composed of $(\mathcal{G},\mathcal{A},\mathcal{R})$, with $\mathcal{G}$=(T_0,N_0,P_0,S_0), $\mathcal{A}$ =$(A(X))_{X \in N0 \cup T0}$, and $\mathcal{R}$ is the set of semantic functions for evaluating the attributes of $X \in N_0 \cup T_0$.

Moreover, for each terminal symbol x_i assume that a set of synthesized attribute $S(x_i)$={val, mod, synrole, coop} is specified by the user. Finally, assume that the user specifies a target sentence x_t in NL, that is a sentence equivalent to the linearized sentence x, i.e. conveys the same information, and it is composed of a subset of terminal symbols of x. The target sentence is necessary for associating the meaning the user wants to give to the multimodal sentence x.

For each terminal symbol x_i, for $1 \leq i \leq k$, the revised CYK algorithm considers the set P' of production rules of the form $A_i \rightarrow x_i$, which have the i^th terminal symbol in the body and the corresponding syntactic category $A_i \in N_0$ in the head. This set P' is added to the initial set P_0. Afterwards, the CYK matrix C (that has dimension k) is initialized and its first row is computed as follows: the elements of the i^th entry of the matrix are the heads A_i of the production rules in P', for $1 \leq i \leq k$. Moreover, the algorithm assigns both a weight to each non-terminal symbol A_i, which is equal to 0.5, and a set of semantic functions R_i, which allow to compute the occurrences of attributes of A_i. In particular, each function $A_i.b \leftarrow x_i.b$ assigns the occurrences of the synthesized attributes b={val, mod, synrole, coop} of the terminal symbols x_i to the corresponding attributes of A_i.

Therefore, the algorithm computes all the remaining rows in the following way. Let i be the column number and j be the row number. The element c_{ij} of the matrix is equal to the head of a production rule $p{:}\ A \rightarrow X\ Y$ such that $X=c_{iq}$ and $Y=c_{i+q,j-q}$ for all $q=1, \ldots, j-1$. If A is a non-terminal symbol in N_0 then the algorithm assigns it to the entry c_{ij} of the matrix C. Otherwise, it creates an appropriate non-terminal symbol B (not in N_0) in order to simulate the generation of the non-terminal symbols X and Y and assigns it to the entry c_{ij} of the matrix C. Moreover, the algorithm assigns a weight to each non-terminal symbol, that is the sum of the weights assigned to X and Y. If $A \in N_0$ the weight is doubled. Furthermore, two semantic functions are assigned to the non-terminal A (or B). The first function computes the occurrence of the attribute *val* of the non-terminal A (or B) as the sum of the occurrences of the attribute *val* of the non-terminal X and/or Y, whose occurrences of the attribute *val* appear as terminal symbols in the target sentence

x_t. The second function computes the occurrence of the attribute *mod* of the non-terminal *A* (or *B*) as the sum of the occurrences of the attribute *mod* of the non-terminal *X* and *Y*.

When all rows of the CYK matrix are computed, the algorithm checks if the cell c_{1k}, associated with the overall sentence *x*, contains the start symbol *S*. In the positive case the sentence *x* can be parsed by the actual grammar and there is no need to update the grammar. Otherwise, the algorithm proceeds with the second step (shown in Figure 5.11) for identifying a further start symbol. In particular, the algorithm chooses the non-terminal symbol *A* with the highest weight in the location c_{1k} of the CYK matrix. Then it considers the production rule $A \rightarrow B\ C$ containing *A* in the head and adds the symbols *B* and *C* as a further derivation of the start symbol *S*. At this point, the new derivation tree can be created by selecting the derivation p_1 (within the set CPR of candidate production rules) that contain *B* (and *C*) in the head. The set of non-terminal symbols, production rules and semantic functions are consequently updated, by adding *B* (and *C*) to N_0, the rule p_1 to P_0, and the functions for computing the occurrence of the attributes of the non-terminal *B* (and *C*) to R_p.

Comparing the original CYK algorithm, that has been introduced in Section 4.4.1 (see Figure 4.2) with the revised CYK algorithm proposed in this thesis, there are basically two main differences in the first step. Firstly, the revised CYK algorithm introduces a weight (see point 2.b and 3.b.ii in Figure 5.10) associated with each element of the CYK matrix for choosing the appropriate candidate production rules during the second step, without the necessity of backtracking. Secondly, as the MAG notation requires that a set of semantic functions is associated to each production rules, the algorithm provides a way to evaluate these functions (see point 2.c and 3.b.ii in Figure 5.10) and stores them in the CYK matrix along the non-terminal symbols. Analogously, in the second step the revised CYK algorithm differs from the original CYK algorithm for two reasons. First of all, the choice of the candidate production rules is based on the weight that each rule has into the grammar (see point 1 in Figure 5.11), instead of the similarity with the form of the rules selected by the user (see point 1 in Step 2 of Figure 4.2). This allows to obtain a more accurate and weighted choice of the production rules to be inserted into the grammar. Finally, in addition to the set of production rules, the algorithm outputs a set of opportunely defined semantic functions associated with each production rule (see point 4 and 5.iv in Figure 5.11).

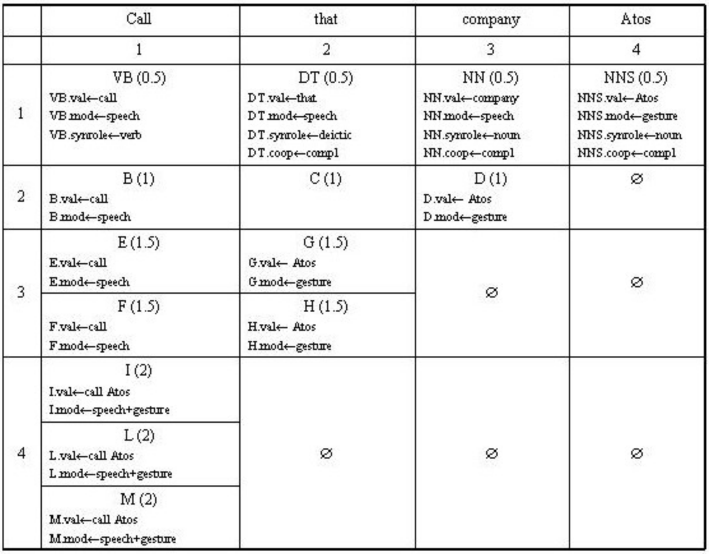
Example 5.2. Consider the multimodal sentence composed of the speech "*call that company*" and the pointing gesture on the company icon on a touch-screen display. The consequent linearized sentence has the form:

SP(Call) SP(that) SP(company) G(Atos).

The initial set of production rules *P'* contains the following rules:

P' = {VB→ Call; DT→ that; NN→ company; NNS→Atos }

Table 5.2: CYK matrix for the example



	Call	that	company	Atos
	1	2	3	4

1	VB (0.5)
	VB.val←call
	VB.mod←speech
	VB.synrole←verb
		DT (0.5)
		DT.val←that
		DT.mod←speech
		DT.synrole←deictic
		DT.coop←compl
			NN (0.5)
			NN.val←company
			NN.mod←speech
			NN.synrole←noun
			NN.coop←compl
				NNS (0.5)
				NNS.val←Atos
				NNS.mod←gesture
				NNS.synrole←noun
				NNS.coop←compl

2	B (1)
	B.val←call
	B.mod←speech
		C (1)	D (1)
			D.val← Atos
			D.mod←gesture
				∅

3	E (1.5)
	E.val←call
	E.mod←speech
		G (1.5)
		G.val← Atos
		G.mod←gesture
			∅	∅

	F (1.5)
	F.val←call
	F.mod←speech
		H (1.5)
		H.val← Atos
		H.mod←gesture		

4	I (2)
	I.val←call Atos
	I.mod←speech+gesture
		∅	∅	∅

	L (2)
	L.val←call Atos
	L.mod←speech+gesture			

	M (2)
	M.val←call Atos
	M.mod←speech+gesture			

The CYK matrix consequent to the running of the first step of the revised CYK algorithm is shown in Table 5.2. Furthermore, the set of candidate production rules CPR contains the following rules:

CPR = {B→ VB DT; C→ DT NN; D→ NN NNS; E →VB C; F → B NN; G →DT D; H→ C NNS; I→ VB G; L→ B D; M→ E NNS}

During the second step, the algorithm chooses randomly one of the symbols in the location c_{14} of the matrix, as they have the same weight (equal to 2). For instance, suppose that the symbol *I* is chosen, which consequently becomes the new start symbol *S* of the grammar. Moreover, the production rule *S → VB G* is added to the set *P'*, along with the corresponding set of semantic functions R_S={S.val←VB.val+G.val; S.mod←VB.mod+G.mod}. At this point the symbols in the body of the selected rule are taken into account: as VB ∈ N_0, it is included in the grammar yet, while *G* is an additional symbol that is not included in the grammar yet. Therefore, the algorithm selects the rule in CPR that contains *G* in the head (i.e. *G →DT D*) and adds this rule to the set *P'*. Moreover, it adds *G* to N_0 and the set of semantic functions R_G={G.val← D.val; G.mod←DT.mod+D.mod} to R_p*'*. Analogously, considering the symbols in the body of the selected rule *G →DT D*, *D* is not included in the grammar yet. Therefore, the algorithm selects the rule in CPR that contains *D* in the head (i.e. *D→ NN NNS*) and adds this rule to the set *P'*. Moreover, it adds *D* to N_0 and the set of semantic functions R_D={D.val← NNS.val; D.mod←NN.mod+NNS.mod} to R_p*'*.

At the end, the grammar *G'* is composed of the following production rules and semantic functions:

P1) S → VB G
  R1.1) S.val←VB.val+G.val
  R1.2) S.mod←VB.mod+G.mod
P2) G →DT D
  R2.1) G.val← D.val
  R2.2) G.mod←DT.mod+D.mod
P3) D→ NN NNS
  R3.1) D.val← NNS.val
  R3.2) D.mod←NN.mod+NNS.mod
P4) VB → Call
  R4.1) VB.val ← call

R4.2) VB.mod ← speech
R4.3) VB.synrole ← verb
P5) DT → That
R5.1) DT.val ← that
R5.2) DT.mod ← speech
R5.3) DT.synrole ← deictic
R5.4) DT.coop ← complementary
P6) NN → Company
R6.1) NN.val ← company
R6.2) NN.mod ← speech
R6.3) NN.synrole ← noun
R6.4) NN.coop ← complementary
P7) NNS → Atos
R7.1) NNS.val ← Atos
R7.2) NNS.mod ← gesture
R7.3) NNS.synrole ← noun
R7.4) NNS.coop ← complementary

At the end of the revised CYK algorithm a new multimodal attribute grammar *G'* that is able to generate the multimodal linearized sentence *x* is available.

However, as the revised CYK algorithm does not use negative examples (i.e. sentences that should not be recognized by the grammar) for limiting the extent of generalization, an additional criterion is needed to avoid the generation of a trivial grammar that accepts any example.

Therefore, during the second step of the proposed grammar inference method, a heuristic based on the minimum description length [Ris78] of the grammar is introduced for computing the "simplicity" of the grammar and a set of two learning operators is applied for improving the grammar description towards more "simple" grammars.

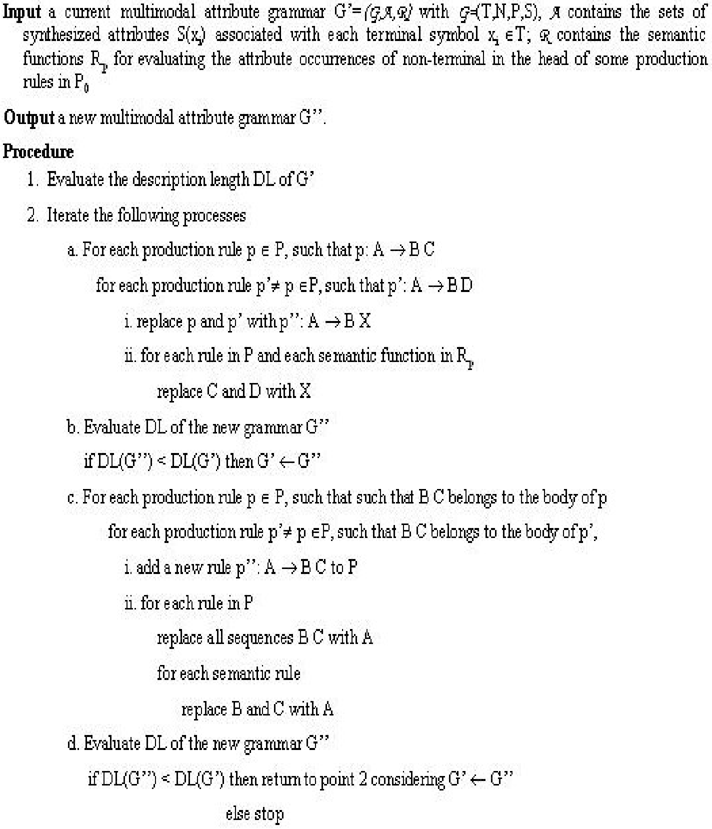
5.5.2 Second Step: Improving the Grammar Description for Avoiding the Over-Generalization Problem

The goal of the second step of the proposed grammar inference method is to update the multimodal attribute grammar *G'*, outputted by the first step, by evaluating its description length and applying to it the learning operators for producing equivalent grammar descriptions that are more "simple" with respect to the description length of the grammar.

In order to handle Multimodal Attribute Grammars, it is necessary to adapt the evaluation of the description length, as well as the *merge* and *create* operators, defined by Petasis et al. [PKK04] and applied to CFGs. Therefore, the significant improvement introduced in this step of the proposed grammar inference method relates to handle not only the set of production rules but also semantic functions both in the description length evaluation and in the learning operator definition.

Roughly, the second step of the grammar inference method, named briefly grammar updating step, works in the following way. It takes as input the multimodal attribute grammar $G'=(\mathcal{G}',\mathcal{A}',\mathcal{R}')$ generated during the first step, where $\mathcal{G}'$=(T',N',P',S'), $\mathcal{A}' =(A(X))_{X \in N \cup T}$, and $\mathcal{R}$ is the set of semantic functions for evaluating the attributes of $X \in N \cup T$.

First of all, the algorithm evaluates the description length *DL'* of *G'*. Afterwards, it repeatedly applies the merge operator considering all ways of unifying non-terminal symbols of *G'*. The resulting grammar *G''* is evaluated by computing the description length *DL''*. If *G''* scores better than the current grammar *G'*, then *G''* replaces the current grammar *G'*, otherwise *G'* remains the current grammar. At this point, the algorithm continues by considering all ways of creating new non-terminal symbols from pairs of symbols that occur in sequence within the grammar repeatedly applying the create operator to the current grammar. Again, the description length of this resulting grammar is evaluated and the grammar that has the lowest score is selected as current grammar. The algorithm iterates the application of the merge and create operators until it is unable to produce a grammar that scores better than the current grammar. A detailed description of the grammar updating step is given in Figure 5.12. Note that the loop identified by *a* in the figure represents the merge operator, while the loop identified by *c* in the figure represents the create operator.



```
Input a current multimodal attribute grammar $G'=(\mathcal{G},\mathcal{A},\mathcal{R})$ with $\mathcal{G}=(T,N,P,S)$, $\mathcal{A}$ contains the sets of
   synthesized attributes $S(x_i)$ associated with each terminal symbol $x_i \in T$; $\mathcal{R}$ contains the semantic
   functions $R_p$ for evaluating the attribute occurrences of non-terminal in the head of some production
   rules in $P_0$

Output a new multimodal attribute grammar G''.

Procedure
   1. Evaluate the description length DL of G'
   2. Iterate the following processes
         a. For each production rule $p \in P$, such that $p: A \rightarrow B\ C$
               for each production rule $p' \neq p \in P$, such that $p': A \rightarrow B\ D$
                  i. replace p and p' with $p'': A \rightarrow B\ X$
                  ii. for each rule in P and each semantic function in $R_p$
                        replace C and D with X
         b. Evaluate DL of the new grammar G''
            if DL(G'') < DL(G') then G' ← G''
         c. For each production rule $p \in P$, such that such that B C belongs to the body of p
                 for each production rule $p' \neq p \in P$, such that B C belongs to the body of p',
                  i. add a new rule $p'': A \rightarrow B\ C$ to P
                  ii. for each rule in P
                        replace all sequences B C with A
                        for each semantic rule
                           replace B and C with A
         d. Evaluate DL of the new grammar G''
            if DL(G'') < DL(G') then return to point 2 considering G' ← G''
                                 else stop
```

Figure 5.12: Grammar updating step

Some more details about the minimum description length model and the learning operators are given in the following two sub-paragraphs.

5.5.2.1 Description Length of a MAG

The minimum description length (MDL) principle, as introduced by Rissanen [Ris78], is a general principle of statistics that allows to seek the shortest possible representation of data expressed through a representation language. In natural language, this principle has been extensively used for grammar inference from positive examples [Wol82] [Gru96] [KeL97] [PPK04] as a heuristic for comparing grammars and selecting the one that is more “compact” with respect to both the length of the grammar as well as the encoding of the training set by the grammar. The choice of the MDL heuristic in our approach is motivated by the encouraging results of the application of the MDL approach to the inference of natural language grammars. In fact, several works [LaS00] [PPK04] show how the use of this heuristic helps in avoiding the over-generalization problem guiding the search process towards the optimal grammar. However, the application of the MDL heuristic to our grammar inference method needs an adaptation as it fits well for CFGs but it can not be applied to MAGs as is.

Before explaining how the MDL principle has been adapted to MAGs, some preliminary definitions have to be given.

Following the approach proposed by Petasis et al. [PPK04], given a context-free grammar *G* and a set of positive examples E, the *description length DL* of *G* is the sum of two independent lengths:

DL = GDL + DDL, where:

- GDL is the Grammar Description Length, i.e. the bits required to encode the grammar rules and transmit them to a recipient who has minimal knowledge of the grammar representation, and
- DDL is the Derivation Description Length, i.e. the bits required to encode and transmit all examples in the set E, provided that the recipient already knows the grammar *G*.

Searching for grammars with minimum description length allows to avoid trivial grammar that has a separate rule for each training sentence, as this grammar will have a large GDL, but also overly general grammars, which will have a large DDL. In fact, the DDL of the language is expected to vary proportionally with its derivation power. As a general grammar involves several rules in the derivation of a single sentence, requiring substantial effort to track all the rules involved in the generation of the sentence, its derivational power has the worst score, and consequently it has the highest DDL.

The evaluation of the two components GDL and DDL, as presented in [PPK04], fits well for CFGs but it needs to be adapted for being applied to MAGs.

Starting from the calculation of the GDL, in order to count the bits required to transmit a multimodal attribute grammar *G'* to a recipient, not only grammar rules have to be encoded but also semantic functions associated with each rule. Similarly to the approach of Petasis et al. [PPK04], a separation of the grammar rules into three independent subsets is applied: the start symbol subset, that contains all the rules whose head is the start symbol *S*; the terminal symbol subset (corresponding to the set of target rules P_T (see Section 5.4)) that contains all the rules of the form $A \rightarrow B$; the non-terminal symbol subset that contains all the rules that are not in the first two subsets.

In addition to the bits required to encode the production rules of the three subsets, that are evaluated by Petasis et al. [PPK04, p.7], the evaluation of the bits required to encode semantic functions is necessary. For this purpose, the following expression can be used:

$$Bits_{SemFunc} = Bits_{Head} + Bits_{Body} + Bits_{Stop}$$

In other words, the total number of bits needed for encoding a semantic function is the sum of the bits required to encode the head ($Bits_{Head}$) and its body ($Bits_{Body}$), similarly to the production rule. Furthermore, a stop symbol should be appended for signaling the end of the semantic function.

For each production rule *p*, the set of occurrences *Occ(p)* and its subsets *Output(p)* and *Input(p)* are considered. Therefore, the bits required for encoding the head of the semantic function is:

$$Bits_{Head} = log_2 (|Output(p)|)$$

and the bits required to encode each term of the body of the semantic function is:

$$Bits_{BodyTerm}= log_2 (|Occ(p)| - 1)$$

Furthermore, in order to encode a semantic function associated with a production rule *p*, the following expression can be used:

$$Bits_{SemFunc} = \log_2\left(\left|Output(p)\right|\right) + \sum_{\substack{\forall\ term\ in\\ rule\ body}} \log_2\left(\left|Occ(p)\right| - 1\right) + Bits_{Stop}$$

The total GDL is therefore given by the sum of the bits required to encode each production rule and its set of semantic functions, for each one of the three subsets, plus two additional stop symbols required to separate the three subsets, as expressed in the following equation.

$$GDL = \sum_{\substack{\forall\ rule\ in\\ Start\ Symbol\\ Subset}} \left(\left(\sum_{\substack{\forall\ NT\\ in\ rule\ body}} Bits_{NT} + Bits_{Stop} \right) + \sum_{\substack{\forall\ SemFunc\\ of\ the\ rule}} Bits_{SemFunc} \right) +$$

$$Bits_{Stop} +$$

$$\sum_{\substack{\forall\ rule\ in\\ Ter\min al\\ Subset}} \left(\left(Bits_{NT} + Bits_{T}\right) + \sum_{\substack{\forall\ SemFunc\\ of\ the\ rule}} Bits_{SemFunc} \right) +$$

$$Bits_{Stop} +$$

$$\sum_{\substack{\forall\ rule\ in\\ Non-Ter\min al\\ Subset}} \left(\left(Bits_{NT} + \sum_{\substack{\forall\ NT\\ in\ rule\ body}} Bits_{NT} + Bits_{Stop} \right) + \sum_{\substack{\forall\ SemFunc\\ of\ the\ rule}} Bits_{SemFunc} \right)$$

In the formula $Bits_{NT}$ and $Bits_{T}$ are the bits required to encode a single instance of a non-terminal and a terminal, respectively, which are computed by using the following expressions from Petasis et al. [PPK04, p.7]:

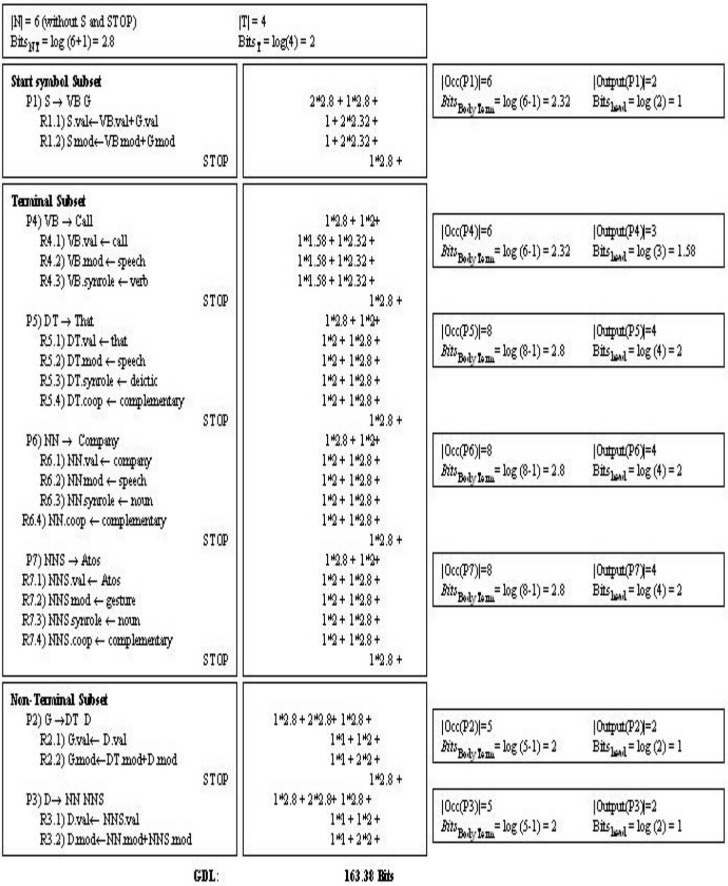
$Bits_{NT} = \log_2(|\mathrm{N}|+1)$

$Bits_{T} = \log_2(|\mathrm{T}|)$

Note that the *Stop* symbol is required for signaling the end of the rules and the semantic functions since their bodies can have variable lengths. This symbol is treated as a non-terminal one, requiring $Bits_{NT}$ to be encoded.

An example of calculating the GDL of the grammar in Example 5.2 of Section 5.5.1 is shown in Table 5.3.

Table 5.3: Calculating the GDL



|N| = 6 (without S and STOP) $Bits_{NT}$ = log (6+1)= 2.8	|T| = 4 $Bits_{T}$ = log(4) = 2	
Start symbol Subset		|Occ(P1)|=6 |Output(P1)|=2 $Bits_{Body\ Term}$ = log (6-1) = 2.32 $Bits_{head}$ = log (2) = 1
P1) S → VB G	2*2.8 + 1*2.8 +	
R1.1) S.val←VB.val+G.val	1 + 2*2.32 +	
R1.2) S.mod←VB.mod+G.mod	1 + 2*2.32 +	
STOP	1*2.8 +	
Terminal Subset		
P4) VB → Call	1*2.8 + 1*2+	|Occ(P4)|=6 |Output(P4)|=3 $Bits_{Body\ Term}$ = log (6-1) = 2.32 $Bits_{head}$ = log (3) = 1.58
R4.1) VB.val ← call	1*1.58 + 1*2.32 +	
R4.2) VB.mod ← speech	1*1.58 + 1*2.32 +	
R4.3) VB.synrole ← verb	1*1.58 + 1*2.32 +	
STOP	1*2.8 +	
P5) DT → That	1*2.8 + 1*2+	|Occ(P5)|=8 |Output(P5)|=4 $Bits_{Body\ Term}$ = log (8-1) = 2.8 $Bits_{head}$ = log (4) = 2
R5.1) DT.val ← that	1*2 + 1*2.8 +	
R5.2) DT.mod ← speech	1*2 + 1*2.8 +	
R5.3) DT.synrole ← deictic	1*2 + 1*2.8 +	
R5.4) DT.coop ← complementary	1*2 + 1*2.8 +	
STOP	1*2.8 +	
P6) NN → Company	1*2.8 + 1*2+	|Occ(P6)|=8 |Output(P6)|=4 $Bits_{Body\ Term}$ = log (8-1) = 2.8 $Bits_{head}$ = log (4) = 2
R6.1) NN.val ← company	1*2 + 1*2.8 +	
R6.2) NN.mod ← speech	1*2 + 1*2.8 +	
R6.3) NN.synrole ← noun	1*2 + 1*2.8 +	
R6.4) NN.coop ← complementary	1*2 + 1*2.8 +	
STOP	1*2.8 +	
P7) NNS → Atos	1*2.8 + 1*2+	|Occ(P7)|=8 |Output(P7)|=4 $Bits_{Body\ Term}$ = log (8-1) = 2.8 $Bits_{head}$ = log (4) = 2
R7.1) NNS.val ← Atos	1*2 + 1*2.8 +	
R7.2) NNS.mod ← gesture	1*2 + 1*2.8 +	
R7.3) NNS.synrole ← noun	1*2 + 1*2.8 +	
R7.4) NNS.coop ← complementary	1*2 + 1*2.8 +	
STOP	1*2.8 +	
Non-Terminal Subset		
P2) G →DT D	1*2.8 + 2*2.8+ 1*2.8 +	|Occ(P2)|=5 |Output(P2)|=2 $Bits_{Body\ Term}$ = log (5-1) = 2 $Bits_{head}$ = log (2) = 1
R2.1) G.val← D.val	1*1 + 1*2 +	
R2.2) G.mod←DT.mod+D.mod	1*1 + 2*2 +	
STOP	1*2.8 +	
P3) D→ NN NNS	1*2.8 + 2*2.8+ 1*2.8 +	|Occ(P3)|=5 |Output(P3)|=2 $Bits_{Body\ Term}$ = log (5-1) = 2 $Bits_{head}$ = log (2) = 1
R3.1) D.val← NNS.val	1*1 + 1*2 +	
R3.2) D.mod←NN.mod+NNS.mod	1*1 + 2*2 +	
GDL:	**163.38 Bits**	

For the calculation of the DDL, in order to count the bits required to encode and transmit the set of training sentences, as recognized by a multimodal attribute grammar *G'* (provided that the recipient already knows the grammar), the equation proposed by Petasis et al. [PPK04, p.9] has been followed, which is quoted below for the sake of thoroughness.

$$DDL = \sum_{\substack{\forall\, rule\, in \\ Start\, Symbol\, Subset}} \left(\log\left(H_{Start\, Symbol}\right) + \sum_{\substack{\forall\, X\, in \\ rule\, body}} \log\left(H_X\right) \right) \cdot F_{rule} +$$

$$\sum_{\substack{\forall\, rule\, in \\ Non\text{-}Terminal\, Subset}} \left(\sum_{\substack{\forall\, X\, in \\ rule\, body}} \log\left(H_X\right) \cdot F_{rule} \right)$$

where:

- $H_X = \begin{cases} Number\ of\ times\ X\ appears\ as\ Head\ of\ a\ rule \\ 1\ \ if\ X\ does\ not\ appear\ as\ Head\ of\ a\ rule \end{cases}$
- F_{rule} is the rule frequency, i.e. the number of sentences from the training set in which the rule is involved in parsing.

An example of calculating the DDL of the grammar in Example 5.2 of Section 5.5.1 is shown in Figure 5.13.

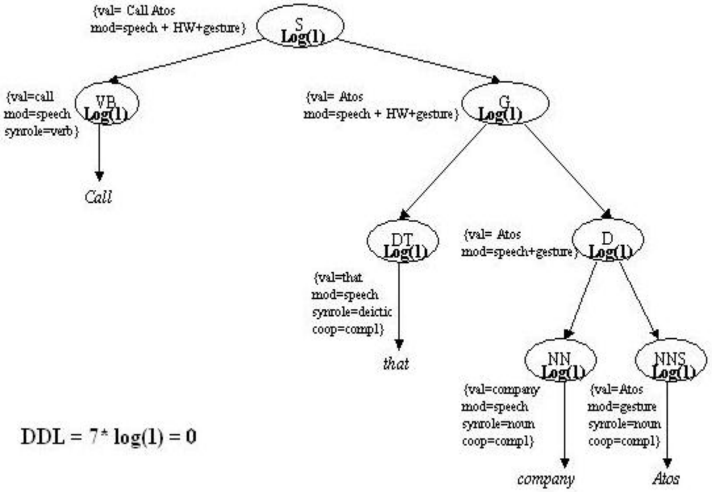


Figure 5.13: Calculating the DDL

Therefore, the description length of the grammar in Example 5.2 of Section 5.5.1 is:

DL(G') = GDL + DDL = 163.38 + 0 = 163.38 Bits

5.5.2.2 Learning Operators

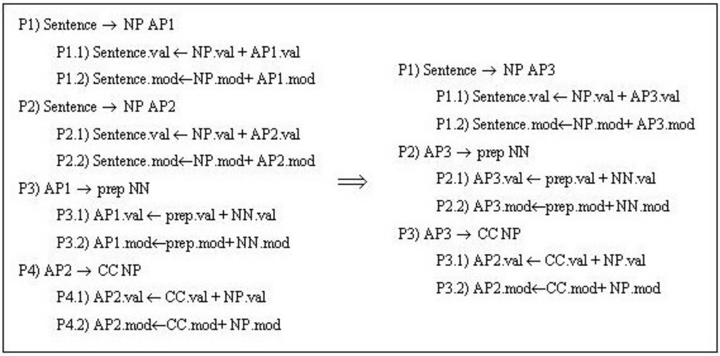
After evaluating the description length of the multimodal attribute grammar *G'*, outputted by the revised CYK algorithm during the first step, the learning operators, proposed by Petasis et al. [PPK04], are applied for computing equivalent grammar descriptions, among which the grammar with minimum description length is selected as output of the proposed grammatical inference method.

Again, the application of the learning operators to our grammar inference method needs an adaptation as they fit well for CFGs but they can not be applied to MAGs as are. Therefore, not only the production rules have to be created or unified but also the associated semantic functions.

A brief description of these learning operators is given below.

The *merge* operator unifies two non-terminals *A* and *B* ∈ *N* into a new non-terminal *Z*, replacing all occurrence of *A* and *B* and the corresponding rule heads. An example of the application of the *merge* operator is shown in Table 5.4. The *merge* operator decreases the GDL of the grammar, since fewer bits are required to encode one non-terminal instead of two. Moreover, the DDL can either increase or decrease.

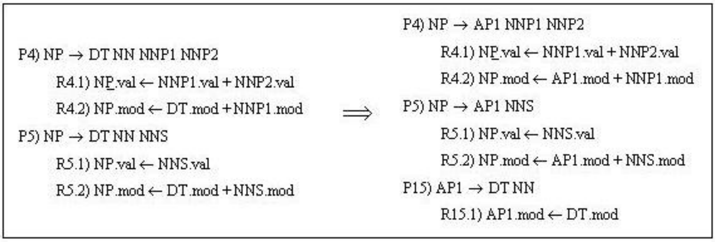
Table 5.4: The effect of the Merge operator



```
P1) Sentence → NP AP1
    P1.1) Sentence.val ← NP.val + AP1.val
    P1.2) Sentence.mod←NP.mod+ AP1.mod
P2) Sentence → NP AP2
    P2.1) Sentence.val ← NP.val + AP2.val
    P2.2) Sentence.mod←NP.mod+ AP2.mod
P3) AP1 → prep NN
    P3.1) AP1.val ← prep.val + NN.val
    P3.2) AP1.mod←prep.mod+NN.mod
P4) AP2 → CC NP
    P4.1) AP2.val ← CC.val + NP.val
    P4.2) AP2.mod←CC.mod+ NP.mod

⟹

P1) Sentence → NP AP3
    P1.1) Sentence.val ← NP.val + AP3.val
    P1.2) Sentence.mod←NP.mod+ AP3.mod
P2) AP3 → prep NN
    P2.1) AP3.val ← prep.val + NN.val
    P2.2) AP3.mod←prep.mod+NN.mod
P3) AP3 → CC NP
    P3.1) AP2.val ← CC.val + NP.val
    P3.2) AP2.mod←CC.mod+ NP.mod
```

The *create* operator creates a new non terminal symbol *Z* that is composed of two sequential non-terminals *A* and *B* ∈ *N*. This implies that a new rule "$Z \rightarrow A\ B$" with the associated semantic function(s) is introduced into the grammar and all occurrences of the sequence *A B* are replaced by the symbol *Z* in the grammar rules. An example of the application of the *create* operator to the grammar of the Example 5.1 (see Section 5.4) is shown in Table 5.5. Moreover, the *create* operator has no effect on the DDL of the grammar, as the derivational power of the grammar remains the same, while it increases the GDL, since more bits are needed in order to represent a further non-terminal.

Table 5.5: The effect of the Create operator



P4) NP → DT NN NNP1 NNP2
R4.1) NP.val ← NNP1.val + NNP2.val
R4.2) NP.mod ← DT.mod + NNP1.mod
P5) NP → DT NN NNS
R5.1) NP.val ← NNS.val
R5.2) NP.mod ← DT.mod + NNS.mod

⟹

P4) NP → AP1 NNP1 NNP2
R4.1) NP.val ← NNP1.val + NNP2.val
R4.2) NP.mod ← AP1.mod + NNP1.mod
P5) NP → AP1 NNS
R5.1) NP.val ← NNS.val
R5.2) NP.mod ← AP1.mod + NNS.mod
P15) AP1 → DT NN
R15.1) AP1.mod ← DT.mod

5.6 Final Discussion

This chapter is motivated by the idea that a grammar-based paradigm is the most natural and coherent with the human-human communication, and, therefore, to provide a multimodal system that relies on a grammar for parsing and interpreting the sentence expressed by the user enables a more flexible and natural interaction. Moreover, a large number of grammars has been defined for natural language processing, which represent a valuable and standardize starting point toward the extension to multimodal input.

To facilitate the grammar definition a "by example" paradigm can be adopted, which allows the end user to provide concrete examples of multimodal sentences that have to be recognized, and the system automatically generates the grammar rules to parse those examples. This implies that a grammar inference method has to be implemented.

Therefore, in this chapter the main achieved results are the definition of a Multimodal Attribute Grammar (MAG) and an algorithm for the inference of this grammar.

The strength of the MAG is the capability to manage whatever modalities and to represent temporal constraints into the grammar rules. Moreover, it provides a good compromise between the context-free paradigm and the necessity to represent semantic and temporal aspects of multimodal input.

The proposed grammar inference method, following an approach "by example", allows to generate the MAG production rules and the associated semantic functions starting from the acceptable multimodal sentences (positive examples only) in polynomial time. The strength of this algorithm relies on its efficiency, simplicity and capability of avoiding the over-generalization problem through the introduction of a heuristic based on the simplicity of the grammar description.

Both the grammar and the inference method represent the key elements on which the development of the grammar editor, described in the following section, is based.

Chapter 6

Multimodal Grammar Editor Design

This chapter, starting from the description of the general architecture of the Multimodal Language Processing (M2LP) framework, details the design process of the Multimodal Grammar Editor (MGE), on which this thesis is focused. Even in its general validity, the design description of the Multimodal Grammar Editor has been carried out using outputs of the unimodal recognizers for speech, gesture, handwriting and sketch, and involving concepts implied by multimodal inputs. Finally, the sequence diagram of the MGE synthetically shows its functioning.

6.1 Introduction

Editing grammars is a difficult and error-prone activity, mainly for users not highly skilled in computational linguistics. Furthermore, a grammar editor represents a useful software tool that enables to define grammars interactively. This chapter describes the Multimodal Grammar Editor (MGE), i.e. a software tool that assists language designers in defining and editing multimodal grammars. The MGE proposed in this thesis allows to define the left and right-hand side of production rules of the grammar, as well as the attribute constraints.

The MGE allows the language designer both to express concrete examples of multimodal sentences, which s/he wants the system recognizes, and to define all the opportune constraints on syntactic roles and types of cooperation among modalities. Afterwards, the editor applies the grammar inference method, described in Section 5.5, for generating the set of production rules and the associated semantic functions, expressed following the MAG notation, described in Section 5.4. The generated grammar serves as input for the multimodal interpreter that applies the grammar production rules and the semantic functions for parsing the sentence and outputting the appropriate interpretation.

A detailed description of the architecture of the Multimodal Language Processing (M2LP) framework, in which the MGE is inserted, is given in Section 6.2, along with a description of the architecture of the MGE. The design process of the MGE is illustrated in Section 6.3. Finally, the functioning of the MGE is showed by means of the sequence diagram given in Section 6.4.

6.2 Overall System Architecture

The Multimodal Grammar Editor (MGE) is a component of the Multimodal Language Processing (M2LP) framework, a system able to acquire information derived from whatever input modalities according to their different representations, to appropriately interpret these inputs with a shared meaning, to integrate these different interpretations into a joint semantic interpretation, and to understand which is the better way to react to the interpreted multimodal sentence by activating the most appropriate output devices.

The overall architecture of the (M2LP) framework is depicted in Figure 6.1. This framework is integrative, configurable, scalable, and adaptive to several application scenarios in order to efficiently manage multimodal communication between people and computational systems.

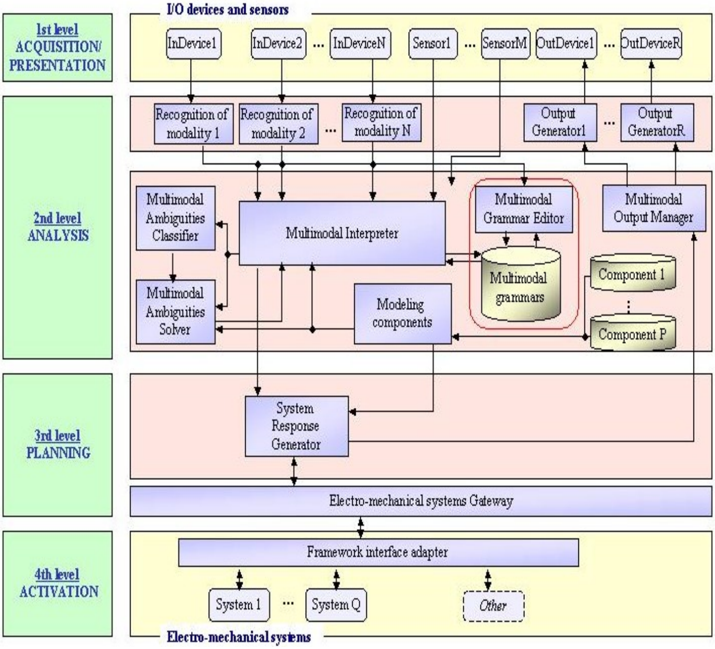

I/O devices and sensors
1st level ACQUISITION/ PRESENTATION
InDevice1
InDevice2
InDeviceN
Sensor1
SensorM
OutDevice1
OutDeviceR
Recognition of modality 1
Recognition of modality 2
Recognition of modality N
Output Generator1
Output GeneratorR
2nd level ANALYSIS
Multimodal Ambiguities Classifier
Multimodal Interpreter
Multimodal Grammar Editor
Multimodal Output Manager
Multimodal grammars
Component 1
Component P
Multimodal Ambiguities Solver
Modeling components
3rd level PLANNING
System Response Generator
Electro-mechanical systems Gateway
4th level ACTIVATION
Framework interface adapter
System 1
System Q
Other
Electro-mechanical systems

Figure 6.1: Architecture of the M2LP framework

As shown in Figure 6.1, four different architectural levels make up the framework:

- The acquisition/presentation level: This level includes the specific I/O devices, such as, for example, display, cameras, microphone, loudspeakers, and input sensors.
- The analysis level: This includes both the unimodal input recognizers, as for example the *Automatic Speech Recognizer* and the gesture recognizer, and the output generators, as the *Speech Synthesizer*. The *Multimodal Interpreter* component integrates the recognized inputs, assigning them the appropriate values for the attributes, as required by the multimodal grammar notation, and applies the production rules stored in the *Multimodal Grammar Repository*, to parse the multimodal input. When the *Multimodal Interpreter* produces multiple interpretations of the same sentence, the linearised multimodal sentence

and the different interpretations are sent to the *Multimodal Ambiguities Classifier* that intercepts the class of ambiguity, which will be solved by the *Multimodal Ambiguities Solver*. The framework acquires the set of production rules of the grammar through the *Multimodal Grammar Editor* component, whose design and development are addressed in this thesis. The analysis level contains also the *Modeling components*, that are aimed at capturing some information used during the interpretation and disambiguation phases for leading up to the most probable interpretation of the user input. Examples of modeling components that can be integrated in the framework can be the user, content and context modeling components. Finally, the analysis level includes the *Multimodal Output Manager* for generating appropriate output information, through the available output modalities (multimodal fission).

- The planning level: This includes the *System Response Generator*, whose main tasks is the understanding of which is the better way to react to the user input (either directly intervening on the electro-mechanical systems, through the *electro-mechanical systems Gateway*, and/or providing specific sensorial feedback) and the consequent adaptation of the human-machine interaction, taking into account also the outputs of the *Modeling Components*. This level contains also the *Electro-mechanical systems Gateway* that provides the link with the electro-mechanical systems. Proper solutions shall be applied to ensure safe interfacing and communication between the two levels.
- The activation level: This level contains the electro-mechanical components offering specific functionalities to the user. It includes a *framework interface adapter* offering specific functions such as communicating to the framework through the electro-mechanical systems gateway.

As the topic of this thesis is the Multimodal Grammar Editor (MGE) component (highlighted by a red rectangle in Figure 6.1), the focus of this chapter will be hereafter the design of this component, starting from its architecture described in the following section.

6.2.1 The Multimodal Grammar Editor Architecture

A block diagram of the architecture of the MGE is shown in Figure 6.2.

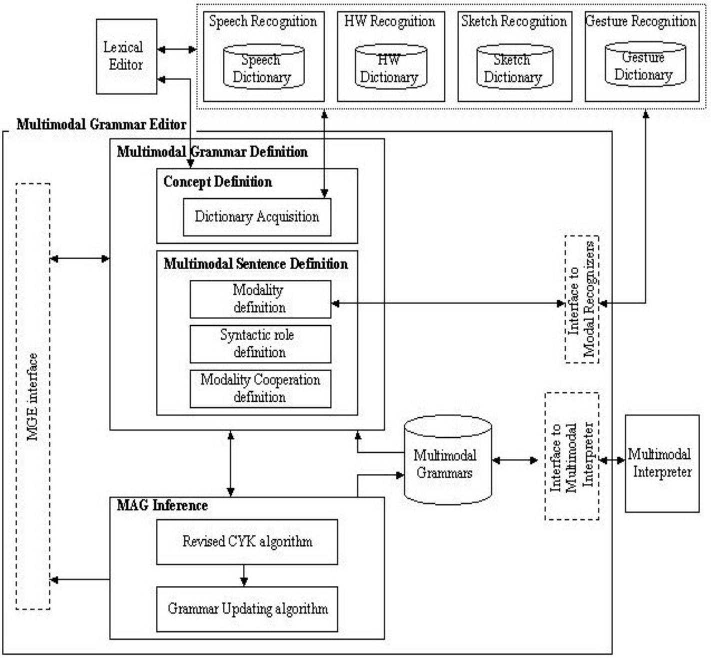


Figure 6.2: Architecture of the Multimodal Grammar Editor

Using the MGE, the language designer, which is the primary user of this component, can create all up the intended grammar or update an existing one. The MGE contains the *Multimodal Grammar Definition* and the *MAG Inference* components and a *MGE interface*. The *MGE interface* is a Multimodal User Interface (MUI) responsible of the interaction between the language designer and both the *Multimodal Grammar Definition* and the *MAG*

Inference components. This interface allows the acquisition of the data to be used for inferring the grammar, i.e. the (positive) examples of sentences and the concepts used for expressing these sentences, since the grammar definition follows a “by example” approach. Moreover, it presents a view onto the multimodal grammar resulting from the MAG inference stage.

The *Multimodal Grammar Definition* sets the grammar that the language designer wants to define by either instantiating a new grammar or selecting an existing grammar from the *Multimodal Grammars* repository, according to the designer’s choice. Furthermore, the *Multimodal Grammar Definition* is responsible for the linearization process, i.e. it takes the elements of the unimodal sentences, coming from the *Multimodal Sentence Definition* sub-component, and combines them opportunely, in order to generate a linear sequence of elements. Such a sequence represents the multimodal linearized sentence that is sent to the *MAG Inference* component for grammar inference.

The *Concept Definition* allows to define the set of terminal symbols, by selecting the appropriate elements from the dictionaries (i.e. the lexicon) of the unimodal recognizers. For this purpose, the *Dictionary Acquisition* requests the dictionaries to the recognizers of the modalities involved in the sentence definition.

The *Multimodal Sentence Definition* allows to specify the positive examples of sentences along with all information needed for the linearization process. In particular, for each example of sentence, this component is responsible of three main tasks. Firstly, it requests to the language designer to select the modalities involved in the example of sentence in order to enable the connection with the appropriate modality recorders. This task is mainly performed by the *Modality definition* component. Secondly, it requests to the language designer to identify the syntactic role that each element has within the sentence. This task is mainly performed by the *Syntactic role definition* component. Finally, it requests to the language designer to define the kind of cooperation among the elements of the inserted sentence. This task is mainly performed by the *Modality Cooperation definition* component.

The *Interface to Modal Recognizers* is responsible for enabling communication between the recorders of the defined modalities, which capture unimodal inputs, and the related recognizers, which convert the captured inputs to recognized concepts.

The *MAG Inference* takes as input the linearized sentence coming from the *Multimodal Grammar Definition*. Furthermore, it applies the grammatical inference method for generating the production rules and the semantic functions (following the MAG notation) that are able to parse the sentence. In particular, the *Revised CYK* and the *Grammar Updating algorithms* (that are described in Sections 5.5.1 and 5.5.2, respectively) are consecutively applied: the former for generating the multimodal attribute grammar that is able to parse the input sentence; the latter for improving the grammar description and avoiding the over-generalization problem. When both the algorithms are applied, the *MAG Inference* stores the generated/updated grammar into the *Multimodal Grammars* repository.

The *Interface to Multimodal Interpreter* provides the bridge between the *Multimodal Grammars* repository and the *Multimodal Interpreter*.

6.3 Design of the Multimodal Grammar Editor

This section describes the design process that has been followed to create the MGE within the M2LP framework.

The MGE aims at providing the language designer all tools for defining a multimodal language. Therefore, the MGE has to satisfy the following requirements:

- to provide a multimodal user interface that interactively leads the designer toward the correct definition of a multimodal grammar by adopting a “by example” approach;
- to allow the acquisition of the lexicon of the grammar, i.e. the terminal symbols that the designer can use for expressing the multimodal sentence;
- to allow the definition of the positive examples of multimodal sentences, by providing the tools for acquiring each unimodal input (e.g., speech, handwriting, sketch, and gesture recorders) and for converting it into a recognized meaningful concept (e.g., using speech, handwriting, sketch, and gesture recognition);

- to provide the way of incrementally updating the grammar description by automatically generating the production rules for parsing the defined examples of multimodal sentences.

These requirements suggested the following steps to design the MGE:

1) create a multimodal user interface (MUI) that allows the language designer to input the data needed for inferring the grammar, according to a "by example" paradigm;
2) use the multimodal user interface for acquiring/defining the lexicon of the grammar;
3) use the multimodal user interface for specifying concrete examples of multimodal sentences and all the opportune constraints on syntactic roles and types of cooperation among modalities;
4) implement a grammar inference algorithm that automatically generates MAG production rules and the associated semantic functions to parse the defined examples of multimodal sentences.

Each step of this design process is explored in more detail in the following sub-sections.

6.3.1 Creating the MUI of the Multimodal Grammar Editor

The usually adopted approach to define grammars is by textually specifying the rules in some descriptive language, such as the Backus-Normal Form (BNF) syntax. The use of textual description, however, is rather difficult for two main reasons. Firstly, it requires the designer to learn the syntax of the descriptive language, i.e. the designer has to have a special skill in computational linguistics for writing the grammar. Secondly, it is very easy to make mistakes, especially if the size of the grammar increases. The use of a MUI, in conjunction with the adoption of a "by example" approach, facilitates the language definition. In fact, in this way the user is not forced to learn a language for manually writing the grammar rules, but s/he has only to think about concrete examples that the grammar has to generate. This makes the grammar definition much more intuitive and less error-prone. In such a way, no skilled language designers are needed, but even non-expert users can define multimodal grammars.

Therefore, a multimodal user interface that leads the language designer to accomplish all steps required for the grammar definition/updating, by allowing her/him to see and verify any moment the choices made till then, has been adopted in the MGE.

The idea of this multimodal user interface is that it has to interactively acquire all data, necessary for generating the grammar, from the language designer, and it has to display the results of the grammatical inference process to the designer. Therefore, the MUI components that can be envisaged are the following:

- a panel for the initial grammar selection, where the language designer can either select the grammar s/he wants to update from a list of existing grammars (that are stored in a multimodal grammar repository), or define a new grammar, if the domains of interest of the existing grammars do not match the need of the designer;
- a panel for the acquisition of the lexicon of the initial grammar (see Section 6.3.2 for more details);
- a panel for the specification of the examples of multimodal sentences (see Section 6.3.3 for more details);
- a panel for setting the opportune constraints on syntactic roles and types of cooperation among modalities (see Section 6.3.3 for more details);
- a panel for displaying both the initial and the grammar generated by the inference method.

6.3.2 Acquiring the Lexicon of the Grammar

The lexicon of a language consists of its vocabulary, including its words and expressions. The lexicon can hardly be complete for any language, since new words are being added all the time, and sometimes old words start to get inflected with new paradigms.

In order to allow that the system “understands” the multimodal sentence specified by the language designer, the elements of the sentence need to be just words from a lexicon known by the system. In other words, the system needs to have a vocabulary specific for the application domain of the intended grammar.

By default, this vocabulary will be composed of the vocabularies of the unimodal input recognizers. If the language designer wants to use a special vocabulary that contains some words not included in the dictionaries of the recognizers, it is necessary that s/he edits the vocabulary by using the lexical editor component (see Figure 6.2), which provides a good support for adding new words to the vocabulary. The design and development of this component is out the scope of this thesis and is one of the topics of future work. However, for the current implementation of the MGE the standard vocabularies of speech, handwriting and sketch recognizers will be used according to the fact that the current implementation of the editor supports these three modalities, even if other modalities can be integrated into the M2LP framework by adding the appropriate input devices and recognizers.

A brief description of the vocabularies of the recognizers is given below.

The standard vocabulary used by the speech recognizer is based on the CMU Pronouncing Dictionary [CMU], which provides pronunciations for words, each one breaking words into sequences of sub-word units.

The vocabulary used by the handwriting recognizer is a small dictionary of lower case Latin letters. However, the lexical editor allows to build further dictionaries according to the need of the designer.

The standard vocabulary used by the sketch recognizer is a small library of geometric objects. Similarly to the handwriting, the lexical editor allows both to build further objects to add to the standard vocabulary, or to define a completely new vocabulary.

6.3.3 Specifying Examples of Multimodal Sentences

In order to start with the grammar inference process, a set of examples of multimodal sentences has to be inputted by the language designer.

As a multimodal sentence consists of different inputs expressed through one or more modalities, each input modality has to be acquired by the editor through an appropriate input device (e.g., a microphone for speech, an editable area for sketch and handwriting). Moreover, the editor has to convert the acquired input data into “meaningful” concepts through the appropriate input recognizers (e.g., speech, sketch and handwriting recognizers).

Once the unimodal inputs are recognized, the language designer has to complete the multimodal sentence specification by defining the opportune constraints both on syntactic roles and types of cooperation among modalities.

Concerning syntactic roles, it is necessary that the designer tags the input elements (previously recognized) with the syntactic category the element belongs to. To support the designer in this task, a syntactic analyzer is used, which provides the possible associations between the input elements and the syntactic categories of the Penn Treebank [MSM94]. In the current implementation of the MGE, the Stanford Log-linear Part-Of-Speech Tagger [TKM03] has been used. Note that this tagger is addressed to NL expression and reads text. Therefore, it can be applied in our editor to unimodal sentences (coming from the recognizers), which are represented by text.

In order to define the types of cooperation among modalities, the designer needs to specify the relations (e.g., complementarity, redundancy) among the input elements (if necessary).

When the multimodal sentence is completely specified, the editor performs the linearization process that translates the unimodal inputs into a linear sequence of elements. This sequence represents the multimodal sentence that will be used as positive sample during the grammar inference process.

For the current implementation of the MGE the following existing recognizers are used, but any other recognizer can also be used.

The SPHINX speech recognizer

The first SPHINX speech recognition system was developed at Carnegie Mellon University in the 1990s. The version used in the MGE, Sphinx-4 [WLK04], is a flexible, modular and pluggable framework based on Hidden Markov Model (HMM). It is composed of three primary modules: the *FrontEnd*, the *Decoder*, and the *Linguist*. The FrontEnd takes one or more input signals and parameterizes them into a sequence of features. The Linguist translates any type of standard language model, along with pronunciation information from the dictionary and structural information from one or more sets of *AcousticModels*, into a *SearchGraph*. The Decoder uses the features from the FrontEnd and the SearchGraph from the Linguist to perform the actual decoding, generating *Results*.

The JARNAL handwriting recognizer

The JARNAL handwriting recognition system [JARNAL] is an open-source application for notetaking, sketching, annotating a document by using a stylus, mouse or keyboard. The version used in the MGE, Jarnal 2.75, works in the following way. It normalizes each strokes to a common scale. Then, a whole bunch of different methods are used to try to determine which stroke/strokes in the dictionary match whatever the user is drawing, and they are averaged together to get a score. Afterwards, a table of probabilities of pairs of letters is applied to guess which the next letter is going to be. Spacing is handled by seeing how far apart you drew your letters, and also using the table of letter pairs.

The sketch recognizer

For the sketch recognition, a system developed by the MultiModal Laboratory of the Italian National Research Council has been used [AFG08]. This recognizer works with SVG files (the objects of the library are SVG files), therefore, the sketch of the user is translated into an SVG representation. Afterwards, a matching algorithm is applied, which performs an exhaustive research of the sketched object into the library, in order to select the right object providing the correct interpretation of the sketch. To achieve that, both the sketch of the user and the objects of the library are represented by a set of nodes (identifying the opposite ends of the strokes that form the sketch) and a set of relationships among these nodes. The algorithm sequentially examines all nodes of the sketch and all relationships searching for the library objects that have the same nodes and relationships. The matching algorithm ends by ranking the library objects according to the similarity of relationships and nodes.

6.3.4 Implementing the Grammar Inference Algorithm

As described in Section 5.5.1, the grammar inference algorithm takes as input:

- the linearized sentence, that is generated by the editor after the language designer has completely specified the multimodal sentence in the MUI;
- the current multimodal grammar, that is selected by the designer through the MUI;
- the sets of occurrences of the synthesized attributes (i.e. actual value, modality, syntactic role, and modality cooperation) associated with each element of the sentence. Actual values of the elements are provided by the recognizers, while the occurrences of all the other attributes are provided by the designer during the specification of the multimodal sentence (see Section 6.3.3).

When all these necessary inputs are acquired by the editor, it applies the grammar inference method, described in Section 6.5, for generating the set of production rules and the associated semantic functions, expressed following the MAG notation. The results of the inference process are displayed in an appropriate panel of the MUI.

6.4 MGE Sequence Diagram

In this section a brief description of the interaction between the language designer and the MGE is illustrated.

Figure 6.3 shows the UML sequence diagram that clarifies how execution switches from the designer to the involved components of the grammar editor.

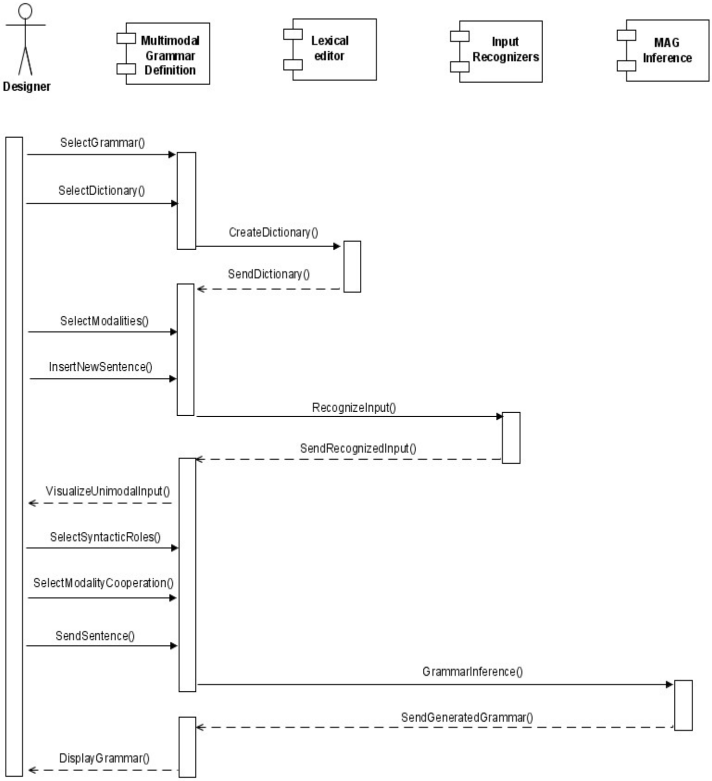


Figure 6.3: Sequence diagram of the MGE

All the classes shown in the diagram realize the functionalities of the MGE described in Sections 6.3.1, 6.3.2, 6.3.3, and 6.3.4. The implementation of this classes is illustrated in the next chapter.

6.5 Summary

This chapter has presented the design process as well as architectural details of the multimodal grammar editor.

Starting from an analysis of requirements that a grammar editor should follow in order to be a useful tool, four main steps to design the editor have been envisaged. First of all, a multimodal user interface for the acquisition of examples of sentences has been designed. Secondly, the acquisition of the vocabularies, containing the terminal symbols that the designer can use for expressing the multimodal sentence, has been discussed. Thirdly, the acquisition of the multimodal sentence through the multimodal user interface has been examined. Finally, the functioning of the grammar inference algorithm has been discussed.

In the following chapter the requirements and design choices explained in the previous sections will be used to implement the multimodal grammar editor.

Chapter 7

Multimodal Grammar Editor Implementation

This chapter describes the implementation process that has been followed to develop the multimodal grammar editor. For explaining the software classes implemented in the prototype, the class diagrams of the main packages are presented following the standard Unified Modeling Language (UML) notation [OMG01]. The editor is implemented using the Java language due to its portability in order to maximize the system independence and to make possible to deploy it on the World Wide Web.

7.1 Introduction

In this chapter the steps towards the development of a prototype of the Multimodal Grammar Editor (MGE) are presented. This prototype, according to the requirements and design choices described in Section 6.3, is composed of two main components. The former is the *Multimodal Grammar Definition* component that performs the acquisition of all data necessary for defining/updating a multimodal grammar, i.e. the (positive) examples of sentences, the concepts used for expressing these sentences, and all the constraints on syntactic roles and types of cooperation among modalities. The acquisition of this data from the language designer is performed through a multimodal user interface, whose design features have been illustrated in Section 6.3.1. The latter is the

MAG Inference component that implements the grammar inference algorithm, described in Section 6.5. This component generates the production rules and semantic functions that are able to parse the acquired examples of sentences.

The prototype has been implemented by using the Java Platform Standard Edition 6 and Netbeans IDE 6.1 as programming environment. The choice of the Java language is due to its portability in order to maximize the system independence and to make possible to deploy it on the World Wide Web.

In the next sections an overview of the implementation of the MGE is given, starting from illustrating the software class design by using the UML class diagrams. Then, a brief description of the main implemented classes is presented. Finally, an example of use of the MGE is described with the help of some screenshots of the main interface of the MGE.

7.2 Software Class Design

The multimodal grammar editor is composed of four main packages, each one implementing a component of the editor, as described in the architecture of Section 6.2.1. These packages, shown in Figure 7.1, are described below:

Multimodal User Interface (MUI). This package manages the interaction between the language designer and the MGE. It provides all functionalities for reading and managing inputs specified by the designer and for writing and displaying the grammar inference output. During the multimodal sentence acquisition it activates the connection with the recorders of the defined modalities and the related recognizers by exploiting the classes of the *interfaceToModalRecognizer* package.

Multimodal Attribute Grammar (MAG). This package contains all classes that implement the data structure of the grammar and the methods for manipulating it.

Multimodal Sentence. This package provides the data structure of the sentence. As the language designer inserts all required data in the MUI, the classes of this package store these data and process them for building the linearized sentence, which is sent as input to the grammar inference algorithm.

Grammar Inference. This package is responsible of the grammar inference algorithm.

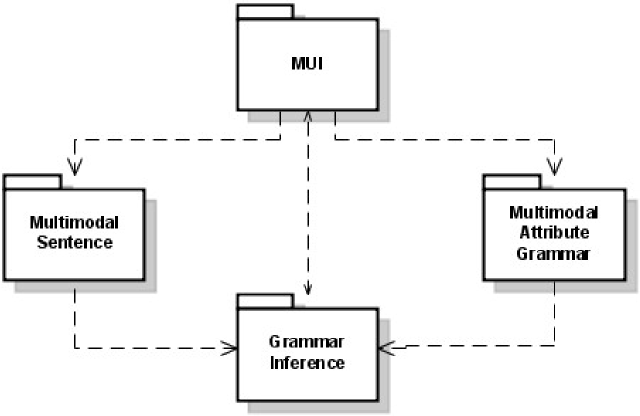


Figure 7.1: General diagram of packages

An explanation of the responsibilities and purposes of each package is given in the following sections.

7.2.1 Multimodal User Interface

Figure 7.2 shows the class diagram of the conceptually most important classes of the *MUI* package.

The main class of this package is the *GrammarEditorMain*, which contains the runnable interface of the MGE. This interface allows both the acquisition of all data, necessary for generating the grammar, from the language designer, and the visualization of the output of the grammatical inference process. For this purpose, several panels are placed in the main window of the interface, each one aimed at allowing a dialogue with the user for acquiring the required pieces of information, such as the current grammar that has to be updated, the modalities used for expressing the sentence, the multimodal sentence itself, the lexicon, and so on.

The class *SyntacticRoleDefinition* provides the interface for the acquisition of the syntactic roles of the input elements.

Analogously, the class *ModCooperationDefinition* provides the interface for defining the type of cooperation among input elements.

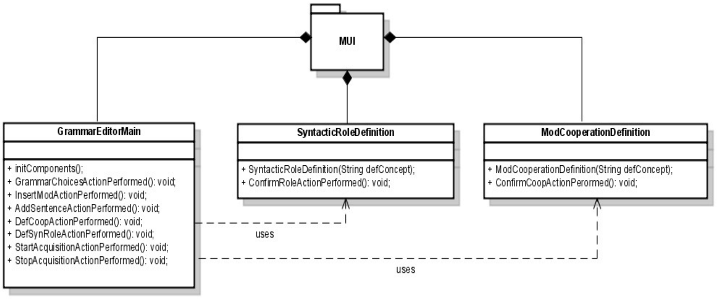

MUI
GrammarEditorMain
+ initComponents();
+ GrammarChoicesActionPerformed(): void;
+ InsertModActionPerformed(): void;
+ AddSentenceActionPerformed(): void;
+ DefCoopActionPerformed(): void;
+ DefSynRoleActionPerformed(): void;
+ StartAcquisitionActionPerformed(): void;
+ StopAcquisitionActionPerformed(): void;
SyntacticRoleDefinition
+ SyntacticRoleDefinition(String defConcept);
+ ConfirmRoleActionPerformed(): void;
ModCooperationDefinition
+ ModCooperationDefinition(String defConcept);
+ ConfirmCoopActionPerormed(): void;
uses
uses

Figure 7.2: Class diagram of the *MUI* package

7.2.2 Multimodal Attribute Grammar

The multimodal attribute grammar (MAG), described in Section 5.4, has been implemented by using an object-oriented structure in the *MultimodalAttributeGrammar* package. In particular, the class hierarchy for the components of this structure is shown in Figure 7.3.

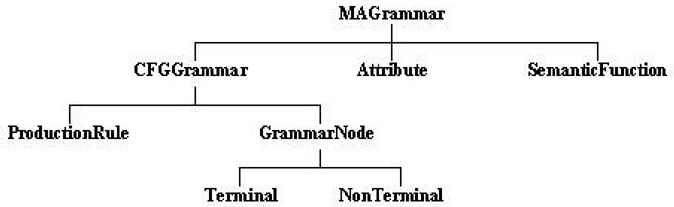


Figure 7.3: Class hierarchy for the MAG

The *MAGrammar* is the root class that provides to external entities all functionalities for creating and manipulating grammar objects. These objects are composed of production rules and grammar nodes, which are implemented by the homonym classes. The class *SemanticFunction* represents and manages the semantic functions associated to the production rules. The class *grammarNode* manipulates *Terminal* and *NonTerminal* objects, which have a set of attributes describing the synthesized and inherited attributes of the MAG notation.

The class diagram of the conceptually most important classes of the *MultimodalAttributeGrammar* package are depicted in Figure 7.4.

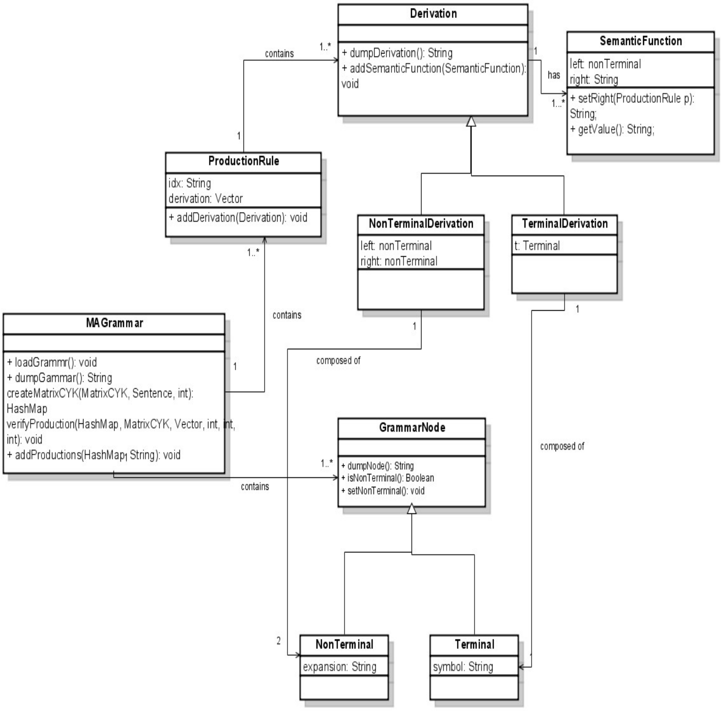


Figure 7.4: Class diagram of the *MultimodalAttributeGrammar* package

7.2.3 Multimodal Sentence

Figure 7.5 shows the class diagram of the conceptually most important classes of the *MultimodalSentence* package.

Sentence is the main class of this package, which provides all functionalities for creating and manipulating sentence objects.

According to the theoretical foundations discussed in Section 5.3.1, these objects are composed of input elements, which are implemented by the class *SentenceElem*. Each element of this class has a set of synthesized attributes, representing the actual value (*token*), modality (*mod*), syntactic role (*syntacticCat*), and modality cooperation (*cooperation*). The actual values of the elements are provided by the recognizers, while the occurrences of all the other attributes are provided by the designer during the specification of the multimodal sentence through the MUI.

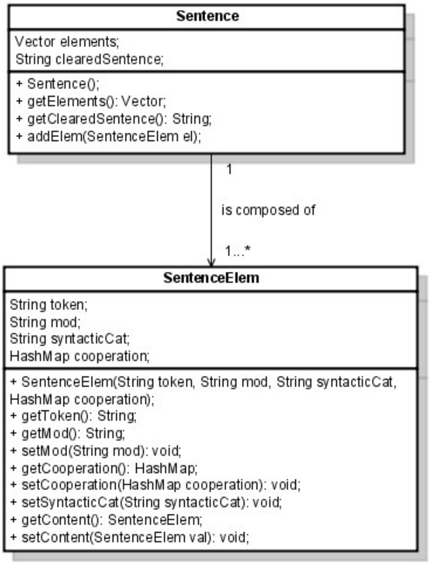

Sentence
Vector elements;
String clearedSentence;
+ Sentence();
+ getElements(): Vector;
+ getClearedSentence(): String;
+ addElem(SentenceElem el);
1
is composed of
1...*
SentenceElem
String token;
String mod;
String syntacticCat;
HashMap cooperation;
+ SentenceElem(String token, String mod, String syntacticCat, HashMap cooperation);
+ getToken(): String;
+ getMod(): String;
+ setMod(String mod): void;
+ getCooperation(): HashMap;
+ setCooperation(HashMap cooperation): void;
+ setSyntacticCat(String syntacticCat): void;
+ getContent(): SentenceElem;
+ setContent(SentenceElem val): void;

Figure 7.5: Class diagram of the *MultimodalSentence* package

7.2.4 Grammar Inference

Figure 7.6 shows the class diagram of the conceptually most important classes of the *GrammarInference* package.

GrammarInference is the main class of this package, which implements the grammar inference algorithm. In particular, the method *upgradeGrammar* in this class is responsible of the control of the algorithm (see Section 7.3.3). As discussed in Section 5.5.1, the first step of the algorithm requires that a CYK matrix is built.

For this purpose, the class *MatrixCYK* is implemented for creating and managing CYK matrix objects. The class *MatrixCYKElement* provides all functionalities for manipulating the elements of the CYK matrix. Each element, implemented by the class *MatrixCYKElementComp,* is an object composed of a non-terminal, its associated weight, and the semantic functions for evaluating the values of the attributes of the non-terminal.

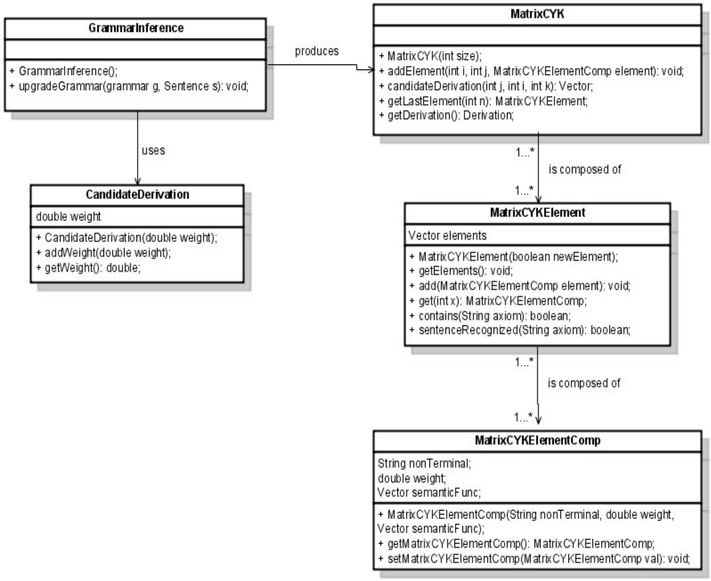

GrammarInference
+ GrammarInference();
+ upgradeGrammar(grammar g, Sentence s): void;
produces
uses
CandidateDerivation
double weight
+ CandidateDerivation(double weight);
+ addWeight(double weight);
+ getWeight(): double;
MatrixCYK
+ MatrixCYK(int size);
+ addElement(int i, int j, MatrixCYKElementComp element): void;
+ candidateDerivation(int j, int i, int k): Vector;
+ getLastElement(int n): MatrixCYKElement;
+ getDerivation(): Derivation;
1...*
is composed of
1...*
MatrixCYKElement
Vector elements
+ MatrixCYKElement(boolean newElement);
+ getElements(): void;
+ add(MatrixCYKElementComp element): void;
+ get(int x): MatrixCYKElementComp;
+ contains(String axiom): boolean;
+ sentenceRecognized(String axiom): boolean;
1...*
is composed of
1...*
MatrixCYKElementComp
String nonTerminal;
double weight;
Vector semanticFunc;
+ MatrixCYKElementComp(String nonTerminal, double weight, Vector semanticFunc);
+ getMatrixCYKElementComp(): MatrixCYKElementComp;
+ setMatrixCYKElementComp(MatrixCYKElementComp val): void;

Figure 7.6: Class diagram of the *GrammarInference* package

7.3 Main Software Classes of the System

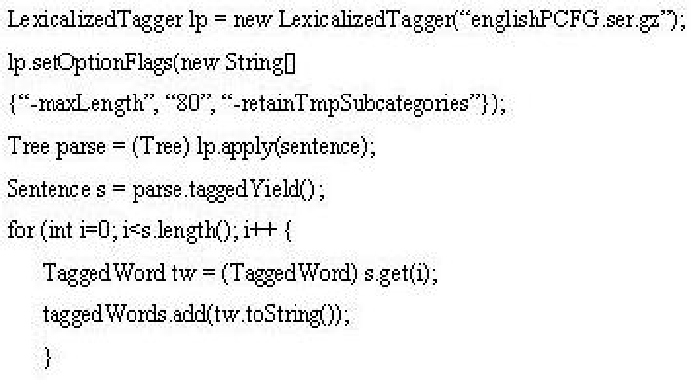
In this section a general overview of the main classes of the MGE is provided. In particular, the class *Tagger* used by the class *SyntacticRoleDefinition* in the *MUI* package, and the class *GrammarInference* in the *GrammarInference* package are described.

7.3.1 Defining Syntactic Roles

The syntactic role of the recognized input elements is a fundamental information for building the CYK matrix during the grammar inference step. To support the language designer in the definition of syntactic roles that each input element has within the sentence, the tagger of Stanford has been used.

The class *Tagger* is devoted to achieve this support. In particular, for each input modality an object *Tagger* is initialized with the unimodal input elements, outputted by the specific recognizer. The output of the method *Tagging()*, in the class *Tagger*, is a list composed of the syntactic categories corresponding to the input elements of the sentence. The method *TagView()* takes this list and opportunely display the association between input elements and syntactic categories in a panel of the window for the syntactic category definition.

Figure 7.7 shows a code excerpt from the method *Tagging()*. The first three rows allows to opportunely initialize the tagger. Afterwards, the tree of the unimodal input elements is created and the method *taggedYield()* is applied to this tree, which generates an object *Sentence* that is the sequence of syntactic categories corresponding to the input elements. The tagging ends by adding the categories in a list, *TaggedWord*, which is used by the method *TagView()* for displaying the output of the tagger to the designer.

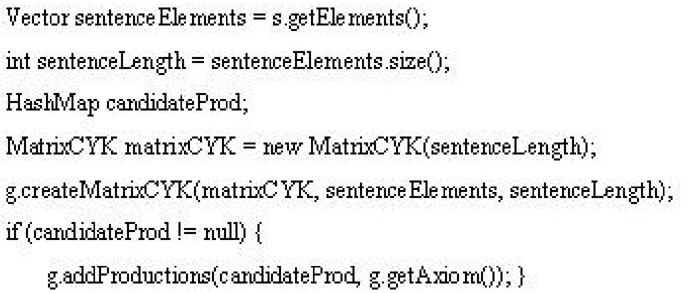


```
LexicalizedTagger lp = new LexicalizedTagger(“englishPCFG.ser.gz”);
lp.setOptionFlags(new String[]
{“-maxLength”, “80”, “-retainTmpSubcategories”});
Tree parse = (Tree) lp.apply(sentence);
Sentence s = parse.taggedYield();
for (int i=0; i<s.length(); i++ {
    TaggedWord tw = (TaggedWord) s.get(i);
    taggedWords.add(tw.toString());
    }
```

Figure 7.7: A code excerpt from the method *Tagging()*

7.3.2 Building of the CYK Matrix

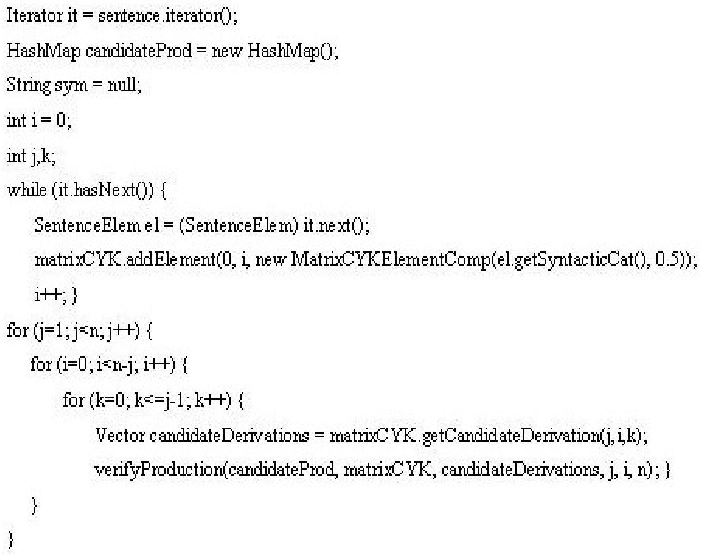
When the language designer has inputted all needed information through the MUI, the controller of the MUI activates the class *GrammarInference* for the generation/updating of the grammar. In particular, the method *upgradeGrammar(Grammar g, Sentence s)* is invoked, which takes as parameter the current MAG grammar *g* and the linearized sentence *s*. A fragment of code of this method is shown in Figure 7.8.



```
Vector sentenceElements = s.getElements();
int sentenceLength = sentenceElements.size();
HashMap candidateProd;
MatrixCYK matrixCYK = new MatrixCYK(sentenceLength);
g.createMatrixCYK(matrixCYK, sentenceElements, sentenceLength);
if (candidateProd != null) {
     g.addProductions(candidateProd, g.getAxiom()); }
```

Figure 7.8: A code excerpt from the method *upgradeGrammar(Grammar g, Sentence s)*

First of all, the elements of the linearized sentence are extracted and the number of these elements is evaluated (through the method *size()*) for defining the dimension of the CYK matrix that will be created. The object *candidateProd*, that is a map, is created for storing the candidate productions that will be inserted into the grammar. Afterwards, the creation of the CYK matrix is delegated to the class *MAGrammar* that invokes the method *createMatrixCYK(matrixCYK, sentenceElements, sentenceLength)* for creating the CYK matrix. An excerpt from this method is shown in Figure 7.9.

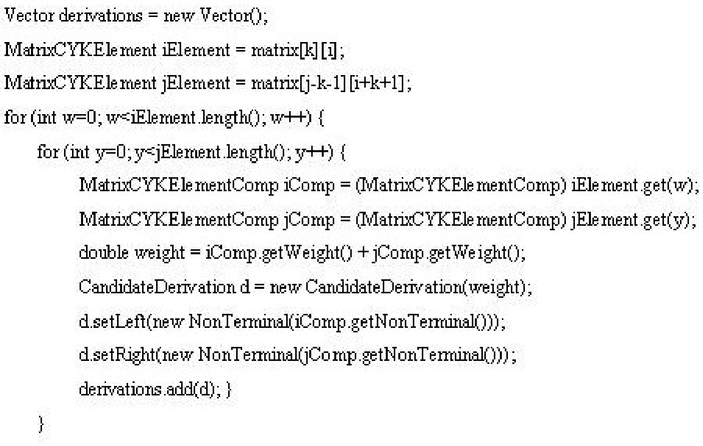


```
Iterator it = sentence.iterator();
HashMap candidateProd = new HashMap();
String sym = null;
int i = 0;
int j,k;
while (it.hasNext()) {
    SentenceElem el = (SentenceElem) it.next();
    matrixCYK.addElement(0, i, new MatrixCYKElementComp(el.getSyntacticCat(), 0.5));
    i++; }
for (j=1; j<n; j++) {
   for (i=0; i<n-j; i++) {
        for (k=0; k<=j-1; k++) {
             Vector candidateDerivations = matrixCYK.getCandidateDerivation(j,i,k);
             verifyProduction(candidateProd, matrixCYK, candidateDerivations, j, i, n); }
   }
}
```

Figure 7.9: A code excerpt from the method *createMatrixCYK(matrixCYK, sentenceElements, sentenceLength)*

The while loop is responsible for loading the elements of the first row of the CYK matrix. As described in Section 5.5.1, these elements are the syntactic categories (obtained by the tagger (see Section 7.3.1)) of the non-terminals that generate directly the terminal symbols of the grammar in the production rules. Afterwards, three nested for loops are implemented. For each element of the matrix, a vector *candidateDerivations* is created, which contains the non-terminals with the related weights. To achieve that, the method *getCandidateDerivation(j,i,k)* of the class *MatrixCYK* is invoked, which is briefly described below (see Figure 7.10). Finally, the method *verifyProduction(candidateProd, matrixCYK, candidateDerivations, j, i, n)* verifies the candidate production rules to be added to the grammar. This method firstly checks if a non-terminal of a candidate rule is present in the body of some grammar rules yet. If it is so, the non-terminal is added in the appropriate location of the CYK matrix and the inspection of the vector of candidate production rules ends. Otherwise, the new symbol is inserted as key of the map (called *candidateProd*) containing the candidate rules and it is subsequently added in the appropriate location of the CYK matrix.

As said before, another method relevant for building the CYK matrix is *getCandidateDerivation(j,i,k)* of the class *MatrixCYK*. An excerpt from this method is shown in Figure 7.10. This method firstly extracts the non-terminals from the CYK matrix. As more than one element can be contained in a location of the matrix, a for loop (with variable *w*) is used for extracting all the non-terminals of a location. Afterwards, the weight of the candidate production is computed. Finally, a new rule is created by initializing an object *CandidateDerivation* that opportunely includes the elements extracted from the CYK matrix as head and body of the rule.



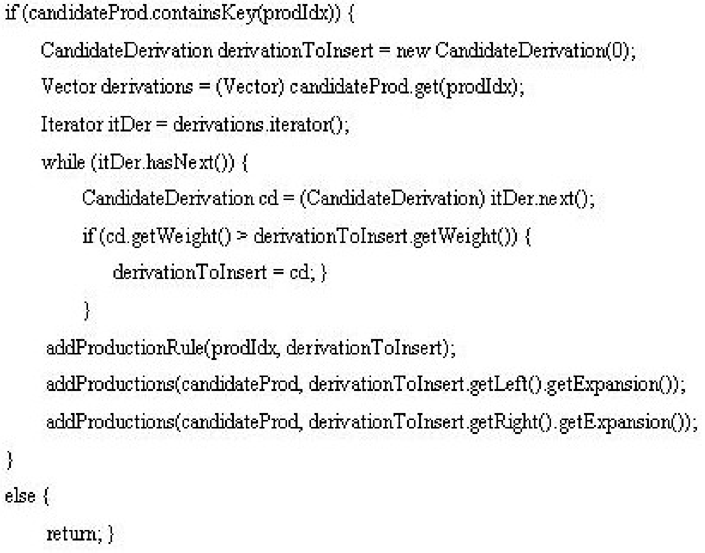
```
Vector derivations = new Vector();
MatrixCYKElement iElement = matrix[k] [i];
MatrixCYKElement jElement = matrix[j-k-1] [i+k+1];
for (int w=0; w<iElement.length(); w++) {
    for (int y=0; y<jElement.length(); y++) {
        MatrixCYKElementComp iComp = (MatrixCYKElementComp) iElement.get(w);
        MatrixCYKElementComp jComp = (MatrixCYKElementComp) jElement.get(y);
        double weight = iComp.getWeight() + jComp.getWeight();
        CandidateDerivation d = new CandidateDerivation(weight);
        d.setLeft(new NonTerminal(iComp.getNonTerminal()));
        d.setRight(new NonTerminal(jComp.getNonTerminal()));
        derivations.add(d); }
    }
```

Figure 7.10: A code excerpt from the method *getCandidateDerivation(j,i,k)*

7.3.3 Revised CYK Algorithm

When the CYK matrix is built, the revised CYK algorithm proceeds with the analysis of both the matrix and the set of candidate production rules for choosing which rule has to be added to the current grammar. This task is delegated to the class *GrammarInference*, which uses the method *upgradeGrammar(Grammar g, Sentence s)*, introduced in Section

7.3.2 (see Figure 7.8). In particular, this method takes the map of the non-terminal symbols associated with the candidate production rules, which is outputted by the invocation of the method *getCandidateDerivation(j,i,k)*, described in Section 7.3.2. If this map is null, then the current grammar is able to generate the multimodal linearized sentence and consequently the algorithm ends. Otherwise, the method *addProductions(HashMap candidateProd, String prodIdx)* of the class *MAGrammar* is invoked. A fragment of code of this method is shown in Figure 7.11.



```
if (candidateProd.containsKey(prodIdx)) {
    CandidateDerivation derivationToInsert = new CandidateDerivation(0);
    Vector derivations = (Vector) candidateProd.get(prodIdx);
    Iterator itDer = derivations.iterator();
    while (itDer.hasNext()) {
        CandidateDerivation cd = (CandidateDerivation) itDer.next();
        if (cd.getWeight() > derivationToInsert.getWeight()) {
            derivationToInsert = cd; }
        }
    addProductionRule(prodIdx, derivationToInsert);
    addProductions(candidateProd, derivationToInsert.getLeft().getExpansion());
    addProductions(candidateProd, derivationToInsert.getRight().getExpansion());
}
else {
    return; }
```

Figure 7.11: A code excerpt from the method *addProductions(HashMap candidateProd, String prodIdx)*

This method takes as parameter the map of candidate production rules and the non-terminal to be inserted in the grammar (the first time the method is called, the axiom is passed). First of all, it checks if the non-terminal is present as key of the map *candidateProd*. If it is so, the vector containing the candidate production rules corresponding to that key is extracted from the map. Afterwards, a while loop is implemented for identifying the derivation in the vector with the best weight. The object *derivationToInsert* contains the rule that will be added to the grammar. Once the best candidate derivation is chosen, the current grammar is updated by using the method *addProductionRule*, which takes as parameter the non-terminal symbol and the derivation. When the non-terminal symbol, contained in the left or right side of the new derivation, is present yet in the set of non-terminal symbols of the grammar, the method *addProductions* ends.

7.4 Usage Example of the Editor

The definition of a new language or its updating is performed by defining new multimodal sentences through the graphical editor shown in Figure 7.12.

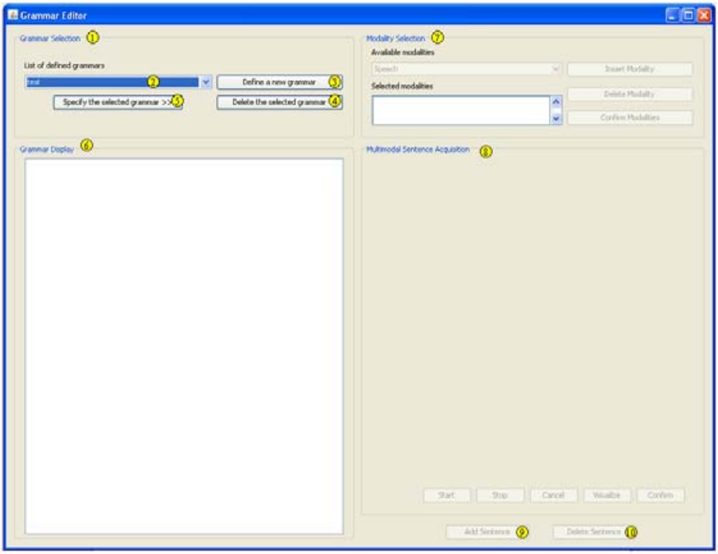


Figure 7.12: The graphical user interface of the grammar editor

This interface is composed of four main panels (identified by number 1,6,7, and 8 in Figure 7.12), corresponding to the macrotasks that the language designer has to accomplish in order to define the intended multimodal grammar.

In particular, the panel for the Grammar selection (number 1 in Figure 7.12) contains the following elements:

- a combo box for selecting an existing grammar in the domain of interest for the application (number 2 in Figure 7.12). When the grammar is selected, it is shown in the panel "Display grammar" (number 6 in Figure 7.12).
- a button for defining a new grammar (number 3 in Figure 7.12), if the intended grammar is not present in the list of existing grammars. When this button is pushed the dialog frame, shown in Figure 7.13, appears and the user can insert the name of the new grammar.
- a button for deleting an existing grammar (number 4 in Figure 7.12). If this button is pushed two warning messages are consecutively visualized for asking the confirmation of deleting the selected grammar in the combo box (number 2 in Figure 7.12).
- a button for specifying the selected grammar (number 5 in Figure 7.12), that enables the panel for the Modality Selection (number 7 in Figure 7.12). When the user pushes this button, the Grammar selection phase ends and the Modality selection phase starts.

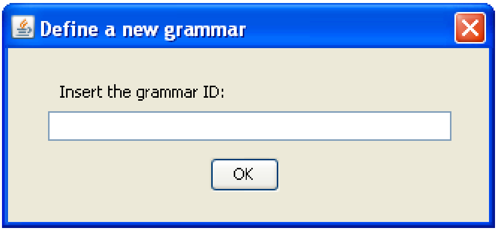
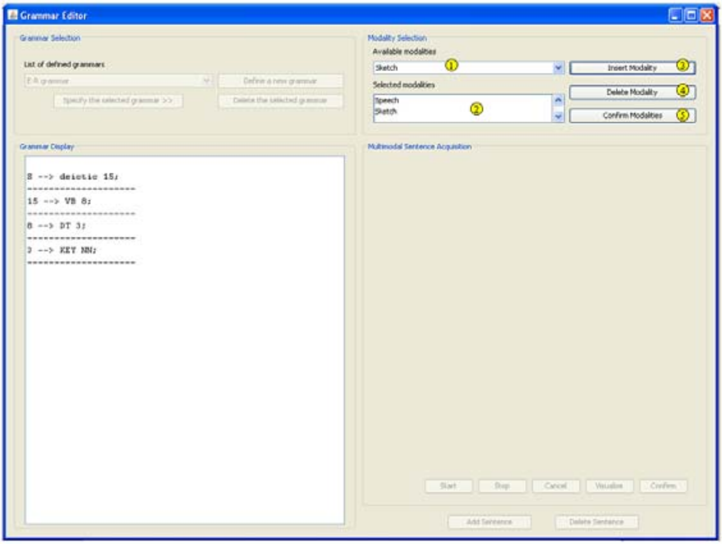


Figure 7.13: The dialog box for inserting the new grammar name



Available modalities: Sketch

Selected modalities:
Speech
Sketch

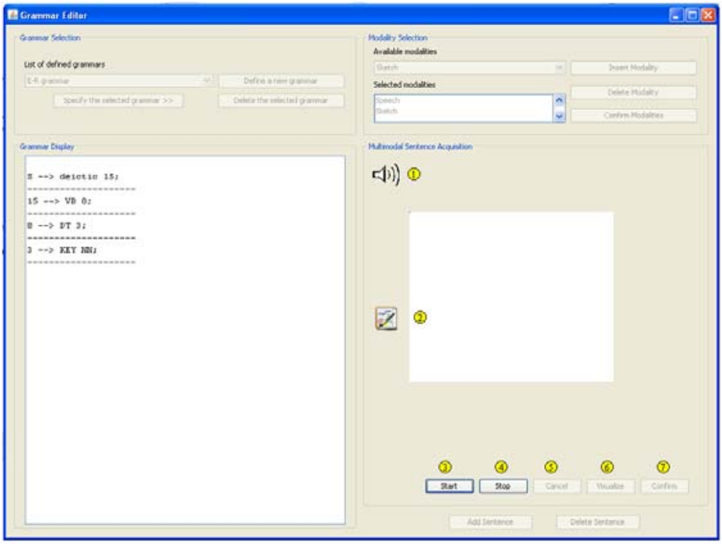
```
S --> deictic 15;
--------------------
15 --> VB 8;
--------------------
8 --> DT 3;
--------------------
3 --> KEY NN;
--------------------
```

Figure 7.14: The panel for modality selection in the graphical user interface of the grammar editor

The panel for the Modality selection (number 7 in Figure 7.12) contains the following elements:

- a combo box for selecting the modalities that are supported by the system and involved in the multimodal sentence (number 1 in Figure 6.14).
- a list (number 2 in Figure 7.14) where the selected modalities are shown.
- a button (number 3 in Figure 7.14) for inserting the selected modality from the combo box into the list.
- a button (number 4 in Figure 7.14) for deleting the selected modality from the list.
- a button for the confirmation of the modality selection (number 5 in Figure 7.14). When this button is pushed, the components of the panel “Multimodal Sentence Acquisition” (number 8 in Figure 7.12) are dynamically created,

according to the modalities that the user has selected (contained in the list).



```
S --> deictic 15;
--------------------
15 --> VB 8;
--------------------
8 --> DT 3;
--------------------
3 --> KEY NN;
--------------------
```

Figure 7.15: The panel for multimodal sentence acquisition in the graphical user interface of the grammar editor

The panel for the Multimodal sentence acquisition (number 8 in Figure 7.12) is dynamically created according to the selected modalities. If the user selected the speech modality, an icon of a microphone (number 1 in Figure 7.15) is shown in the panel and the speech recorder is connected to the interface. If the user selected the sketch (or handwriting) modality, an icon of a sketch (or handwriting) and an editable area for the sketch input acquisition are arranged in the panel (number 2 in Figure 7.15).

In addition to this dynamic components, the panel for the Multimodal sentence acquisition contains the following static elements:

- a button for starting the multimodal sentence acquisition (number 3 in Figure 7.15). When this button is pushed, the system enables the modality recorders to acquire the input of the user.

- a button for concluding the multimodal sentence acquisition (number 4 in Figure 7.15). When this button is pushed, the system disables the modality recorders to acquire the input of the user.
- a button for deleting the acquired multimodal sentence (number 5 in Figure 7.15), if the user is not satisfied by this sentence.
- a button for visualizing the inserted input (number 6 in Figure 7.15). When this button is pushed, the system opens a new window where the unimodal input recognized by the specific recognizers are displayed. This window is described in detail below.
- a button for confirming the acquired sentence (number 7 in Figure 7.15), if the user is satisfied by this sentence. When this button is pushed, the system starts the linearization process, which provides the linear sequence of input elements. Afterwards, the user can confirm the acquisition of the sentence by pushing the button "Add Sentence" in the main interface (number 9 in Figure 7.12), otherwise s/he can cancel the acquisition by pushing the button "Delete Sentence" (number 10 in Figure 7.12). If the "Add Sentence" button is pushed, the system starts the grammar inference algorithm, whose results are displayed in the panel "Display grammar" (number 6 in Figure 7.12).

The window for visualizing the unimodal input recognized by the specific recognizers is shown in Figure 7.16. This window contains the following elements:

- a button for defining the syntactic roles of the inserted unimodal input;
- a button for defining the kind of cooperation among the inserted unimodal input;
- a button for concluding the definition of the syntactic roles and the kinds of cooperation of the recognized input elements.

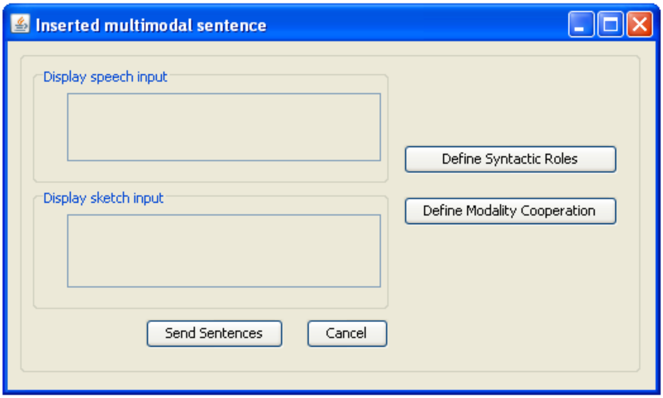


Figure 7.16: The window for visualizing the unimodal input recognized by the specific recognizers

In order to explain the grammar editor functionalities, a simple example is shown in the following. Suppose the user wants to define a multimodal language for editing E-R diagrams. First of all, s/he selects the appropriate grammar from the combo box (number 2 in Figure 7.12). Whether an E-R grammar does not exist yet, the user can define it by pushing the button "Define a new grammar" and inserting the name of the new grammar in the textual field. Let us suppose that the concepts dictionary containing the *entity* and the *relationship* concepts belonging to the E-R domain have been already defined. Moreover, dictionaries containing the different modalities symbols representing these concepts have been already defined too.

Suppose again that the first multimodal sentence that the user inserts is composed by the speech input "This is the entity Professor" and the sketch input of a rectangle representing an entity. Therefore, the user selects the speech and sketch modalities from the combo box (number 1 in Figure 7.14), and confirms the selection by pressing the button "Confirm Modalities" (number 5 in Figure 7.14). Afterwards, a dynamic editable area appears (number 8 in Figure 7.12) where the user can input the multimodal sentence by using a microphone for the speech and the area for the sketch.

The acquisition of the sentence begins with the pressure of the button “Start”, as shown in Figure 7.17.

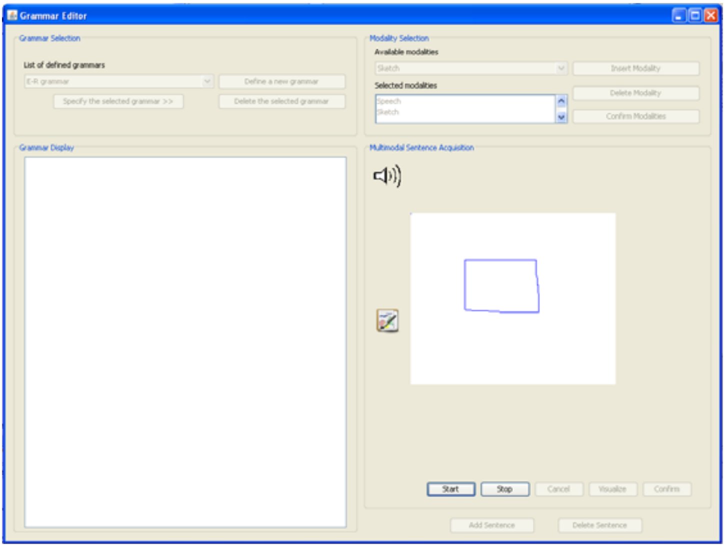
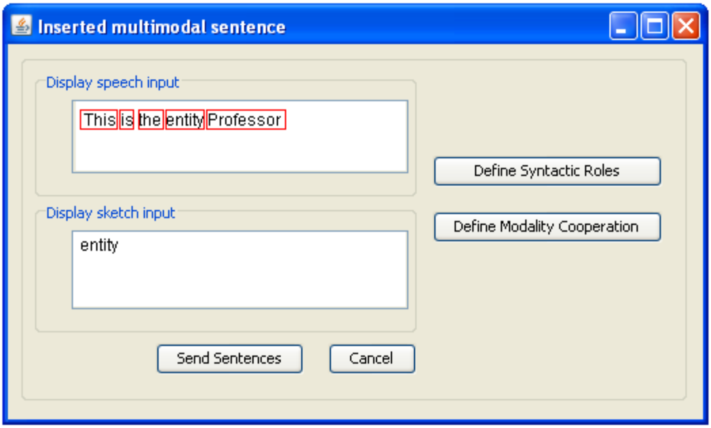


Figure 7.17: Multimodal sentence acquisition

When the user finishes to insert the sentence, s/he presses the button “Stop”. If the user is not satisfied by the inserted sentence s/he can cancel it and start again the acquisition.

Pressing the button “Visualize”, the system acquires the input elements of the multimodal sentence, sends them to the specific recognizers and displays the recognized inputs in the window “Inserted Multimodal Sentence”, as shown in Figure 7.18.



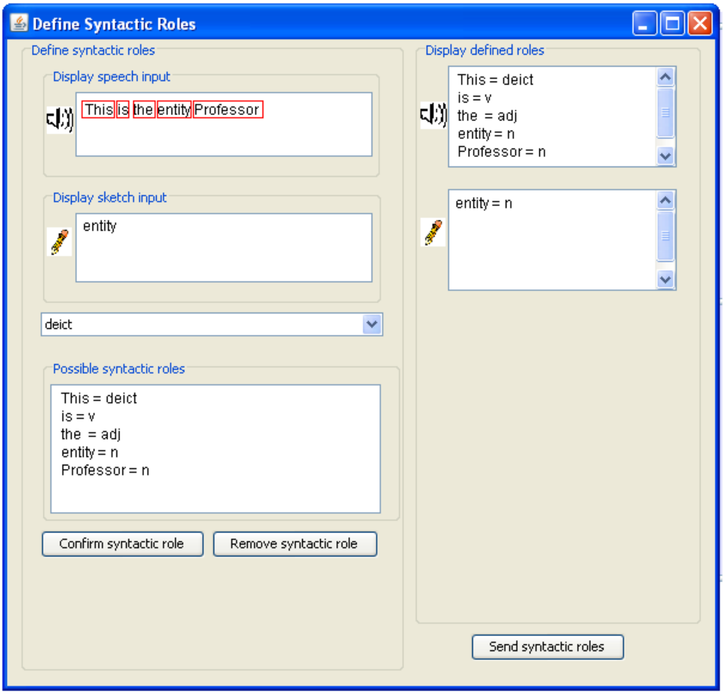
Inserted multimodal sentence

Display speech input: This is the entity Professor

Display sketch input: entity

Figure 7.18: Recognized unimodal inputs

At this point the user presses the button “Define Syntactic Roles” for identifying the syntactic role that each element has within the sentence. Pressing this button, the interface in Figure 7.19 is visualized. As the input in our assumption can be represented through a Natural Language (NL) expression, the system provides the possible syntactic roles of the input elements; these roles comes from the application of a NL parser to the NL expression. For instance, the application of a NL parser to the speech input in the example produces the set of syntactic roles shown in the area “Possible syntactic roles” of Figure 7.19. Whether these roles correspond to the intention of the user, s/he can confirm them by pressing the button “Confirm syntactic role”, otherwise s/he can define the syntactic roles manually by selecting the input element and choosing the appropriate role in the drop-down list. When syntactic roles are defined for all input elements, the user visualizes the defined roles in the text area on the right of the interface, and either confirms them by pressing the button “Send syntactic roles” or delete some syntactic roles by pressing the button “Remove syntactic role”.



This is the entity Professor

entity

deict

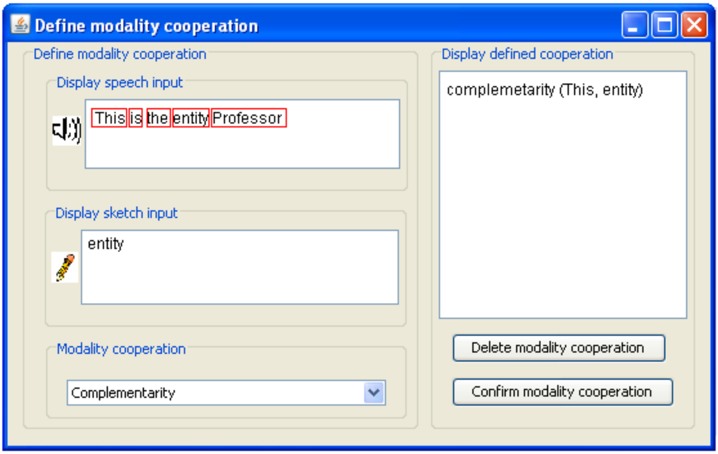
This = deict
is = v
the = adj
entity = n
Professor = n

This = deict
is = v
the = adj
entity = n
Professor = n

entity = n

Figure 7.19: Interface for the definition of syntactic roles of inserted input

At this point the user presses the button “Define Modality Cooperation” for identifying the kind of cooperation among input elements. Pressing this button, the interface in Figure 7.20 is visualized. The user selects the input elements that have to be linked by a cooperation mode (complementarity, redundancy..) and chooses the appropriate mode in the drop-down list. When all necessary rules of modality cooperation are defined, the user visualizes them in the text area on the right of the interface, and s/he either confirms them by pressing the button “Confirm modality cooperation” or deletes some cooperation rules by pressing the button “Delete modality cooperation” and the system shows again the main interface configuration of Figure 7.17.


Define modality cooperation
Define modality cooperation
Display defined cooperation
Delete modality cooperation
Confirm modality cooperation

Display speech input: This is the entity Professor

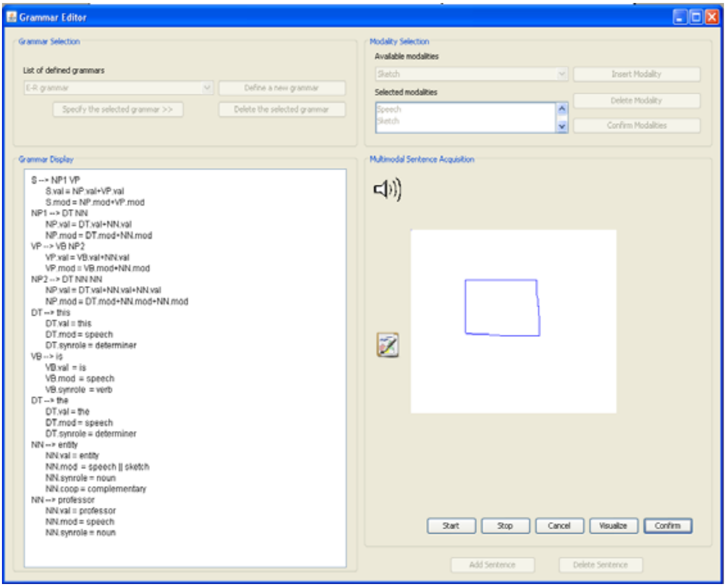
Display sketch input: entity

Modality cooperation: Complementarity

complemetarity (This, entity)

Figure 7.20: Interface for the definition of modality cooperation

At this point the user presses the button “Add Sentence” of Figure 7.17 for concluding the multimodal sentence input. The system automatically applies the algorithm of grammar inference and generates the production rules necessary for parsing the inserted sentence. The user can visualize these rules in the text area on the left of the interface (number 3 in Figure 7.12). In Figure 7.21 the production rules generated for the example are shown.



```
S --> NP1 VP
    S.val = NP.val+VP.val
    S.mod = NP.mod+VP.mod
NP1 --> DT NN
    NP.val = DT.val+NN.val
    NP.mod = DT.mod+NN.mod
VP --> VB NP2
    VP.val = VB.val+NN.val
    VP.mod = VB.mod+NN.mod
NP2 --> DT NN NN
    NP.val = DT.val+NN.val+NN.val
    NP.mod = DT.mod+NN.mod+NN.mod
DT --> this
    DT.val = this
    DT.mod = speech
    DT.synrole = determiner
VB --> is
    VB.val  = is
    VB.mod  = speech
    VB.synrole  = verb
DT --> the
    DT.val = the
    DT.mod = speech
    DT.synrole = determiner
NN --> entity
    NN.val = entity
    NN.mod  = speech || sketch
    NN.synrole = noun
    NN.coop = complementary
NN --> professor
    NN.val = professor
    NN.mod = speech
    NN.synrole = noun
```

Figure 7.21: Visualization of the generated production rules for the example

7.5 Summary

This chapter has presented the implementation process of the multimodal grammar editor. The editor relies on two main components: the first, named *Multimodal Grammar Definition*, is devoted to acquire all data necessary for defining/updating a multimodal attribute grammar through a multimodal user interface; and the other one, named *MAG Inference*, is responsible of the implementation of the grammar inference algorithm, described in Section 5.5.

Therefore, the final result of the implementation process is an editor that, following a “by example” approach, allows to define production rules of the multimodal attribute grammar from concrete examples of multimodal sentences. The application of a grammar inference method, which automatically generates the production rules, relieves the user from the task of learning the grammar formalism, making grammars easier to use also by non-expert users.

In the next chapter, two experiments are presented that were conducted to investigate the usability of the multimodal grammar editor and the performance of the grammar inference algorithm.

Chapter 8

Evaluation and Results

This chapter presents some validation of the Multimodal Grammar Editor (MGE), whose theoretical foundations, design and implementation are described in previous chapters. The goals of the validation are mainly twofold. First of all, the usability of the MGE has been assessed for understanding how well it works in practice. Secondly, the evaluation of the grammar inference algorithm has been performed for measuring the correctness of the induced grammar.

8.1 Introduction

The evaluation of the Multimodal Grammar Editor (MGE) has been carried out from a twofold point of view: firstly, by considering the editor as a software system, the assessment of the usability of this system can be performed; secondly, by considering the editor as a grammar inference system, the correctness of the inferred grammar can be evaluated.

In order to evaluate the usability of the editor, some experiments were conducted, which involved six subjects. These experiments aimed at observing the subjects while interacting with the editor in order to provide some real data about the usability of the editor. In particular, the ability of real users to accomplish the grammar definition using the multimodal interface of the grammar editor has been evaluated. This evaluation and its consequent results are illustrated in Section 8.2.

Concerning the second point of view, evaluating grammar inference systems within NLP is a critical task for several reasons. First of all, in order to evaluate an inferred grammar it is necessary to compare it against a "correct" grammar, which is difficult to identify. Secondly, the ambiguity represent an obstacle as there is no an obvious single correct grammar that represents a given set of training examples. These issues have been largely addressed in the literature and several evaluation metrics have been defined for measuring the correctness of the induced grammar. A brief description of the three main evaluation methods used in NL grammar inference is provided in Section 8.3.1.

The analysis of the advantages and drawbacks of these methods has lead to choice the *rebuilding known grammars* method for evaluating the proposed grammar inference algorithm due to its simplicity and objectivity of the evaluation. The application of this method and its results are illustrated in Section 8.3.2 and 8.3.3.

8.2 Usability Evaluation of the MGE

Usability testing has been defined by Barnum [Bar02] as the "process of learning from users about a product's usability by observing them using the product". Originally, these tests were conducted with a large number of users (30-50). Nowadays, the advent of modern usability testing methods has allowed to decrease the number of participants, requiring 5-7 representative users for finding most of the problems, in particular when it is a qualitative test.

The usability testing performed for evaluating the MGE consisted of a series of user trials designed to assess how the editor functionalities are perceived by the users. A total of six subjects took part in the testing. This small number of people is justified also by the qualitative nature of the analysis that has been performed.

A detailed description of the testing phases and an overview of obtained results are given in the following sub-sections.

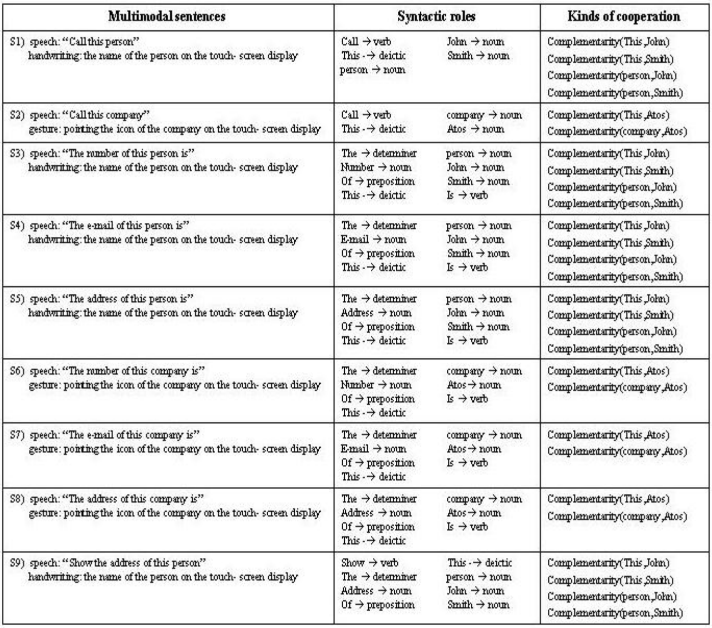
8.2.1 Experimental Setting

In order to assess the usability of the implemented editor, a series of experiments among the research staff of the Institute of Research on Population and Social Policies were conducted. The objective of these experiments was to compare the ease of writing grammars by examples instead of writing grammars by text.

The total number of participants in the experiments is six. These people have been divided in two groups: group 1 (1 male and 2 females) is composed of members of the MultiModal Laboratory, with high skill with multimodal languages, and group 2 (2 males and 1 female) is composed of members of the social science research staff, without any skill with multimodal languages. This partition has been introduced because people of group 1 already know the multimodal grammar formalism and, therefore, they have less difficulties to write grammars by text. For this reason people belonging to group 1 should provide a preference in using MGE or the text-based editor, independently from their specific skill. On the contrary, people of group 2 need to be trained in the multimodal attribute grammar formalism and, consequently, have more difficulties to write grammars by text.

The trial consisted of defining a multimodal attribute grammar, starting from a set of positive examples of multimodal sentences, which are shown in the first column of Table 8.1. For defining this grammar, the participants have to use alternatively a textual editor, which requires the specification of grammar rules in BNF text form, and the multimodal grammar editor proposed in this thesis, that requires the specification of a set of multimodal sentences. These different versions were identified to the participants as “yellow editor” and “red editor”, respectively, in order to avoid influencing their opinions on the editors.

Table 8.1: The multimodal sentences for the experiments



Multimodal sentences	Syntactic roles		Kinds of cooperation

S1) speech: "Call this person"
handwriting: the name of the person on the touch- screen display
	Call → verb
	This -→ deictic
	person → noun
		John → noun
		Smith → noun
			Complementarity(This ,John)
			Complementarity(This ,Smith)
			Complementarity(person,John)
			Complementarity(person,Smith)

S2) speech: "Call this company"
gesture: pointing the icon of the company on the touch- screen display
	Call → verb
	This -→ deictic
		company → noun
		Atos → noun
			Complementarity(This ,Atos)
			Complementarity(company ,Atos)

S3) speech: "The number of this person is"
handwriting: the name of the person on the touch- screen display
	The → determiner
	Number → noun
	Of → preposition
	This -→ deictic
		person → noun
		John → noun
		Smith → noun
		Is → verb
			Complementarity(This ,John)
			Complementarity(This ,Smith)
			Complementarity(person,John)
			Complementarity(person,Smith)

S4) speech: "The e-mail of this person is"
handwriting: the name of the person on the touch- screen display
	The → determiner
	E-mail → noun
	Of → preposition
	This -→ deictic
		person → noun
		John → noun
		Smith → noun
		Is → verb
			Complementarity(This ,John)
			Complementarity(This ,Smith)
			Complementarity(person,John)
			Complementarity(person,Smith)

S5) speech: "The address of this person is"
handwriting: the name of the person on the touch- screen display
	The → determiner
	Address → noun
	Of → preposition
	This -→ deictic
		person → noun
		John → noun
		Smith → noun
		Is → verb
			Complementarity(This ,John)
			Complementarity(This ,Smith)
			Complementarity(person,John)
			Complementarity(person,Smith)

S6) speech: "The number of this company is"
gesture: pointing the icon of the company on the touch- screen display
	The → determiner
	Number → noun
	Of → preposition
	This -→ deictic
		company → noun
		Atos → noun
		Is → verb
			Complementarity(This ,Atos)
			Complementarity(company ,Atos)

S7) speech: "The e-mail of this company is"
gesture: pointing the icon of the company on the touch- screen display
	The → determiner
	E-mail → noun
	Of → preposition
	This -→ deictic
		company → noun
		Atos → noun
		Is → verb
			Complementarity(This ,Atos)
			Complementarity(company ,Atos)

S8) speech: "The address of this company is"
gesture: pointing the icon of the company on the touch- screen display
	The → determiner
	Address → noun
	Of → preposition
	This -→ deictic
		company → noun
		Atos → noun
		Is → verb
			Complementarity(This ,Atos)
			Complementarity(company ,Atos)

S9) speech: "Show the address of this person"
handwriting: the name of the person on the touch- screen display
	Show → verb
	The → determiner
	Address → noun
	Of → preposition
		This -→ deictic
		person → noun
		John → noun
		Smith → noun
			Complementarity(This ,John)
			Complementarity(This ,Smith)
			Complementarity(person,John)
			Complementarity(person,Smith)

The yellow editor has an interface (see Figure 8.1) similar to the MGE interface, described in Section 7.4. The fundamental difference is in the panel for the acquisition of multimodal sentences, that in the yellow editor is replaced by an editable area for manually inserting the textual rules of the multimodal attribute grammar. Therefore, in this editor there is no need of the grammar inference method, as the grammar description is manually defined by the user. The trial with the yellow editor consists in writing the production rules and semantic functions, which generate the given multimodal sentences (in the first column of Table 8.1), within the textual area of the editor. The correct description of the grammar that generates these sentences is shown in Table 8.2. As group 2 is composed of people not skilled in multimodal grammars, they need to be trained in the multimodal attribute grammar formalism.

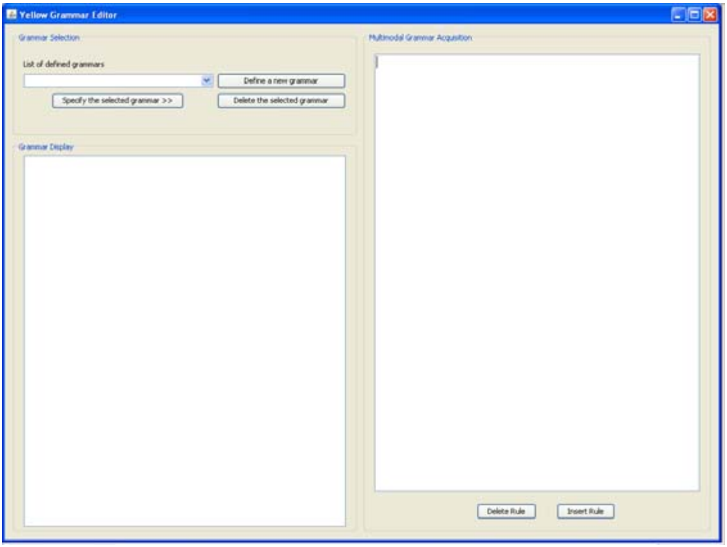
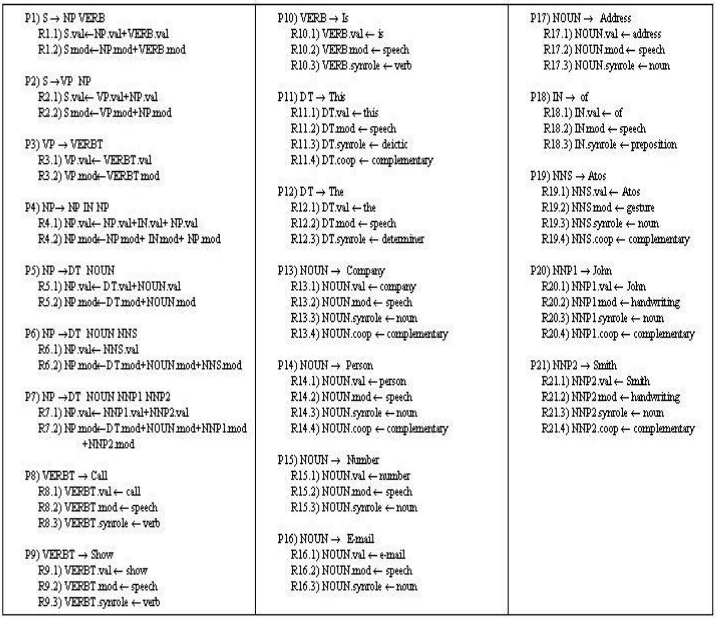


Figure 8.1: Interface of the yellow editor

The red editor is the MGE and, therefore, it works as described in Section 7.4. The trial with this editor consists in formulating the set of multimodal sentences, which are shown in the first column of Table 8.1, with syntactic roles and kinds of cooperation shown in second and third columns of Table 8.1. If the grammar is correctly defined, the red editor has to output the multimodal attribute grammar shown in Table 8.2.

To perform the experiments a PC workstation has been configured, with the two editors available. All trials were conducted on this workstation, in order to have the same hardware configuration.

Table 8.2: The multimodal attribute grammar for the experiments



```
P1) S → NP VERB
   R1.1) S.val←NP.val+VERB.val
   R1.2) S mod←NP mod+VERB mod

P2) S →VP  NP
   R2.1) S.val← VP.val+NP.val
   R2.2) S mod←VP mod+NP mod

P3) VP → VERBT
   R3.1) VP.val← VERBT.val
   R3.2) VP mod←VERBT mod

P4) NP→ NP IN NP
   R4.1) NP.val← NP.val+IN.val+ NP.val
   R4.2) NP mod←NP mod+ IN mod+ NP mod

P5) NP →DT  NOUN
   R5.1) NP.val← DT.val+NOUN.val
   R5.2) NP mod←DT mod+NOUN mod

P6) NP →DT  NOUN NNS
   R6.1) NP.val← NNS.val
   R6.2) NP mod←DT mod+NOUN mod+NNS mod

P7) NP →DT  NOUN NNP1 NNP2
   R7.1) NP.val← NNP1.val+NNP2.val
   R7.2) NP mod←DT mod+NOUN mod+NNP1 mod
            +NNP2 mod

P8) VERBT → Call
   R8.1) VERBT.val ← call
   R8.2) VERBT mod ← speech
   R8.3) VERBT.synrole ← verb

P9) VERBT → Show
   R9.1) VERBT.val ← show
   R9.2) VERBT mod ← speech
   R9.3) VERBT.synrole ← verb

P10) VERB → Is
   R10.1) VERB.val ← is
   R10.2) VERB mod ← speech
   R10.3) VERB.synrole ← verb

P11) DT → This
   R11.1) DT.val ← this
   R11.2) DT mod ← speech
   R11.3) DT.synrole ← deictic
   R11.4) DT.coop ← complementary

P12) DT → The
   R12.1) DT.val ← the
   R12.2) DT mod ← speech
   R12.3) DT.synrole ← determiner

P13) NOUN →  Company
   R13.1) NOUN.val ← company
   R13.2) NOUN mod ← speech
   R13.3) NOUN.synrole ← noun
   R13.4) NOUN.coop ← complementary

P14) NOUN →  Person
   R14.1) NOUN.val ← person
   R14.2) NOUN mod ← speech
   R14.3) NOUN.synrole ← noun
   R14.4) NOUN.coop ← complementary

P15) NOUN →  Number
   R15.1) NOUN.val ← number
   R15.2) NOUN mod ← speech
   R15.3) NOUN.synrole ← noun

P16) NOUN →  E-mail
   R16.1) NOUN.val ← e-mail
   R16.2) NOUN mod ← speech
   R16.3) NOUN.synrole ← noun

P17) NOUN →  Address
   R17.1) NOUN.val ← address
   R17.2) NOUN mod ← speech
   R17.3) NOUN.synrole ← noun

P18) IN →  of
   R18.1) IN.val ← of
   R18.2) IN mod ← speech
   R18.3) IN.synrole ← preposition

P19) NNS → Atos
   R19.1) NNS.val ← Atos
   R19.2) NNS mod ← gesture
   R19.3) NNS synrole ← noun
   R19.4) NNS.coop ← complementary

P20) NNP1 → John
   R20.1) NNP1.val ← John
   R20.2) NNP1 mod ← handwriting
   R20.3) NNP1 synrole ← noun
   R20.4) NNP1.coop ← complementary

P21) NNP2 → Smith
   R21.1) NNP2.val ← Smith
   R21.2) NNP2 mod ← handwriting
   R21.3) NNP2 synrole ← noun
   R21.4) NNP2.coop ← complementary
```

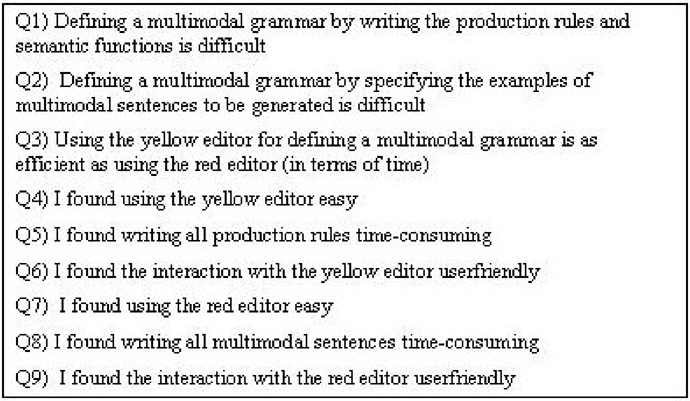
The trial included the following phases (in temporal order):

- a training phase, in which an explanation of the task and a short tutorial on how to operate each editor prototype are given. For people of group 2, a tutorial on the multimodal attribute grammar formalism is provided.
- a familiarization phase, in which the participants were asked to use both editors in order to become familiar with their functionalities. For this familiarization, a set of sample sentences are used, different from the sentences of the trial.
- an experimental phase, that is the core of the trial. Once the participants felt comfortable with the editors, they were asked to use alternatively the yellow editor, that requires the definition of grammar rules in BNF text form (as shown in Table 8.2) starting from the multimodal sentences, shown in the first column of Table 8.1, and the red editor, that requires

the formulation of the set of multimodal sentences, shown in Table 8.1.

- an evaluation phase, in which each participant was asked to fill out a simple evaluation questionnaire aimed to find out whether it liked the editor or not. The questionnaire (detailed described in Appendix A) is composed of nine questions that asked the users to rate (on a 5-point Likert scale ranging from “strongly disagree” to “strongly agree”) certain features (e.g., helpfulness, usefulness, etc.) of the two editors. These questions are briefly summarized in Table 8.3. In addition, a final question asked the users which of the editors they preferred to use, and their comments, if any.

Table 8.3: The questionnaire for the usability evaluation



Q1) Defining a multimodal grammar by writing the production rules and semantic functions is difficult

Q2) Defining a multimodal grammar by specifying the examples of multimodal sentences to be generated is difficult

Q3) Using the yellow editor for defining a multimodal grammar is as efficient as using the red editor (in terms of time)

Q4) I found using the yellow editor easy

Q5) I found writing all production rules time-consuming

Q6) I found the interaction with the yellow editor userfriendly

Q7) I found using the red editor easy

Q8) I found writing all multimodal sentences time-consuming

Q9) I found the interaction with the red editor userfriendly

A detailed description of the instructions provided to the test participants and the evaluation questionnaire can be found in Appendix A.

8.2.2 Results

The test results described below demonstrate that the multimodal grammar editor, designed and implemented in this dissertation, is a valid support for the definition of multimodal grammars.

In fact, the majority of participants (5 users, i.e. 83%) answered to the final question that they prefer using the red editor, and only one participant (17%), belonging to group 1, prefers the yellow editor. The answers to the evaluation questionnaire are summarized in the pie charts of Figure 8.2.

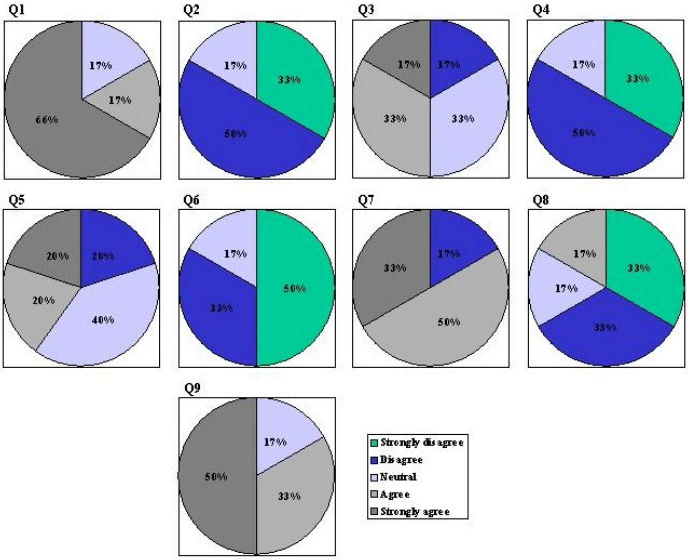


Figure 8.2: Responses to the evaluation questionnaire

The first three questions aim to find out whether the participants have difficulty in writing grammars by examples using the red editor or by text using the yellow editor. All people of group 2 and two people of group 1 found easier to define a multimodal grammar by specifying the examples of multimodal sentences instead of writing the production rules (question 1 and 2). The remaining one person of group 1 was neutral.

Considering differences between the two users' groups it is possible to observe that the red editor is usually preferred also by people skilled in multimodal grammars. In the comments, a user of group 1 said that s/he "prefers the red editor as it is easier to use according to its natural interaction". Moreover, a user of group 2 said that s/he “prefers the red editor because s/he is not obliged to remember how to formalize the grammar”.

The further three questions aim to assess the usability of the yellow editor, in terms of difficulty, time and user-friendliness. 5 out of 6 participants evaluated negatively the ease and user-friendliness of the yellow editor (question 4 and 6), while only one person (of group 1) was neutral. Participants were divided as to whether writing all production rules using the yellow editor is time-consuming (question 5): people of group 2 tended towards the agreement, while people of group 1 towards the disagreement. This is due to the time required to learn the grammar by people of group 2. Comments made by participants of group 2 regarding the interaction with the yellow editor are the following: “I found the process of the grammar definition an onerous work for me”, “I’m not skill in grammars and the preliminary tutorial is not sufficient for enabling me to use the yellow editor”. A user of group 1 said that “I’m not sure of the correctness of the grammar I have defined and the yellow editor does not support me in resolving my problems”.

The final three questions aim to assess the usability of the red editor, in terms of difficulty, time and user-friendliness. 5 out of 6 participants evaluated positively the ease and user-friendliness of the red editor (question 7 and 9), while only one person (of group 1) was neutral. Most participants (4 users, i.e. 66%) did not find writing the multimodal sentences through the red editor time-consuming (question 8). Comments made by participants of group 2 regarding the interaction with the red editor are the following: “I felt the red editor more suitable for my skill because I don’t need to learn the grammar formalism”.

8.3 Evaluation of the Grammar Inference Algorithm

A brief review of existing evaluation methods for grammar inference, along with a detailed description of the experimental phases and an overview of obtained results, are given in the following sub-sections.

8.3.1 Evaluation metrics

The evaluation of grammar inference algorithms is not a trivial task, and many different approaches have been proposed in the literature.

The *looks good to me* approach has prevailed for many years due to its apparent simplicity. When a grammar inference algorithm is evaluated using this approach, the algorithm is applied to a piece of unstructured text and the resulting grammar is qualitatively evaluated on the base of the linguistic intuitions of the evaluator, that highlights the grammatical structures which look "good". As this approach needs only unstructured data to be applied, it can be evaluated on different languages without the need of structured corpora [Zaa01]. However, the method has many disadvantages. First of all, this kind of evaluation is mainly conducted by an expert who has specific knowledge of the syntax of the language, that is generally the developer of the system. This leads to a high chance of a biased evaluation, making it almost impossible to gain an accurate picture of system performance.

Another approach for evaluating grammar inference algorithms is the *compare against treebank*. This evaluation method consists in applying the grammar inference algorithm to a set of plain natural language sentences which are extracted from an annotated treebank, which is selected as a "gold standard". The structured sentences generated by the algorithm are then compared against the original structured sentences from the treebank. There are several metrics that can be used to compare the learned tree against the original tree structure. Most often, the recall, which gives a measure of the completeness of the learned grammar, and the precision, which shows how correct the learned structure is, are used. The *compare against treebank* method does not need an expert to indicate if some construction is correct or incorrect, allowing for a relatively objective comparison of different algorithms. The main problem with this approach is that structured corpora are needed. This may be a problem in the case of multimodal languages because structured treebanks are not available and need to be built by hand (or semi-automatically).

The *rebuilding known grammars* approach is another evaluation method, which will be followed in this thesis. This method, starting from a pre-defined (simple) grammar, generates a set of example sentences, which are given as input to the grammar inference algorithm and the resulting grammar is compared manually to the original grammar. If the inferred grammar is similar or equal to the original grammar then the learning system is considered good. The advantages of this evaluation method are quite similar to the *looks good to me* approach. An additional advantage, similarly to the *compare against treebank* method, is that the evaluation can be done automatically, without the need for a language expert, and, therefore, it yields a more objective way of comparing different algorithms. One of the disadvantages of this approach is that the evaluation of the system depends heavily on the chosen grammar.

From the analysis of these existing evaluation methods, the *rebuilding known grammars* and the *compare against treebank* approaches have most potential, mainly for the objectivity of the evaluation they perform. In particular, the former works well with relatively small artificial grammars, while the latter requires a large corpus that contains multimodal language data and syntactic tree structures generated by the multimodal grammar. As such a kind of corpus does not exist in the literature yet, it should be built opportunely, requiring lots of resources (both in time and money). This severely restricts the application of this evaluation method to grammar inference methods for multimodal languages.

Therefore, the *rebuilding known grammars* method is applied in this thesis. In particular, for evaluating whether the inferred grammar is similar or equal to the original grammar, the following two aspects of the inferred grammar are measured during the evaluation, according to the study of [LaS00]:

- errors of omission (failures to parse sentences generated by the “correct” grammar), which indicate that an overly specific grammar has been learned,
- errors of commission (failures of the “correct” grammar to parse sentences generated by the inferred grammar), which indicate that an overly general grammar has been learned.

More formally, given the artificial “correct” grammar G_C and the inferred grammar G_I, errors of omission can be estimated as the fraction of the number of sentences generated by G_C that are not parsed by G_I to the total number of sentences generated by G_C.

Errors of commission can be estimated as the fraction of the number of sentences generated by G_I that are not parsed by G_C to the total number of sentences generated by G_I.

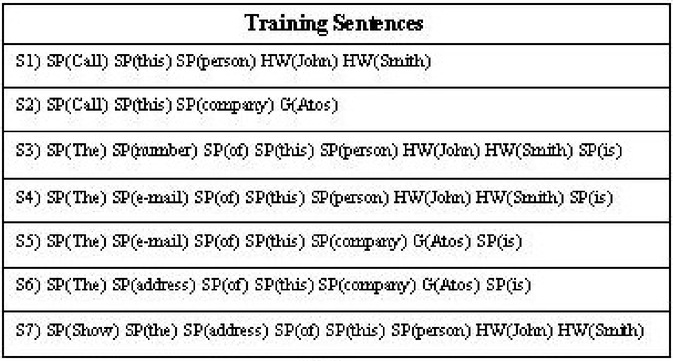
The application of this method to the proposed grammar inference algorithm is described in the following sections.

8.3.2 Experimental Setting

In order to evaluate the grammar inference algorithm proposed in this thesis, several experiments were conducted, following the *rebuilding known grammars* evaluation method.

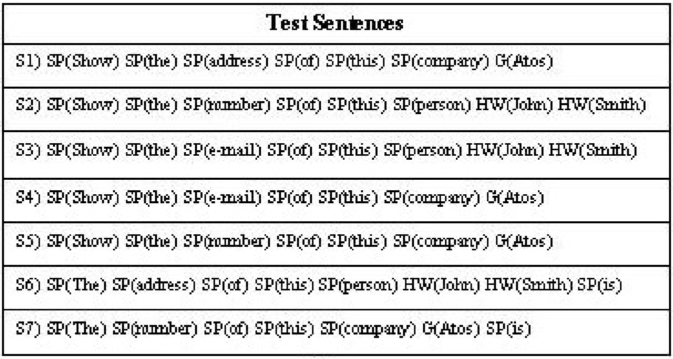
The main objective of the experiments is to examine the ability of the grammar inference algorithm to infer a "correct" multimodal attribute grammar. To achieve that, the artificial grammar shown in Table 8.2 has been used, which is the same applied for the usability evaluation, described in the previous section. Two sets of positive training sentences S_{TR} and test sentences S_{TE} were generated top-down from the artificial grammar. The training sentences are shown in Table 8.4.a (they are the linearized sentences corresponding to the sentences used for the usability evaluation), while the test sentences are depicted in Table 8.4.b. It is necessary that the same sentence does not appear both in the training and test sets. The training set S_{TR} was used to train the algorithm for generating the inferred grammar, while the test set S_{TE} was used for evaluating the performance of the inferred grammar in terms of errors of omission. Furthermore, for evaluating the error of commission, another set of test sentences S_{TE2} was generated top-down from the inferred grammar. These sentences were used for evaluating the performance of the inferred grammar in terms of errors of commission.

Table 8.4: Training and test sentences for the experiments



Training Sentences
S1) SP(Call) SP(this) SP(person) HW(John) HW(Smith)
S2) SP(Call) SP(this) SP(company) G(Atos)
S3) SP(The) SP(number) SP(of) SP(this) SP(person) HW(John) HW(Smith) SP(is)
S4) SP(The) SP(e-mail) SP(of) SP(this) SP(person) HW(John) HW(Smith) SP(is)
S5) SP(The) SP(e-mail) SP(of) SP(this) SP(company) G(Atos) SP(is)
S6) SP(The) SP(address) SP(of) SP(this) SP(company) G(Atos) SP(is)
S7) SP(Show) SP(the) SP(address) SP(of) SP(this) SP(person) HW(John) HW(Smith)

(a)



Test Sentences
S1) SP(Show) SP(the) SP(address) SP(of) SP(this) SP(company) G(Atos)
S2) SP(Show) SP(the) SP(number) SP(of) SP(this) SP(person) HW(John) HW(Smith)
S3) SP(Show) SP(the) SP(e-mail) SP(of) SP(this) SP(person) HW(John) HW(Smith)
S4) SP(Show) SP(the) SP(e-mail) SP(of) SP(this) SP(company) G(Atos)
S5) SP(Show) SP(the) SP(number) SP(of) SP(this) SP(company) G(Atos)
S6) SP(The) SP(address) SP(of) SP(this) SP(person) HW(John) HW(Smith) SP(is)
S7) SP(The) SP(number) SP(of) SP(this) SP(company) G(Atos) SP(is)

(b)

Therefore, the evaluation requires the following phases:

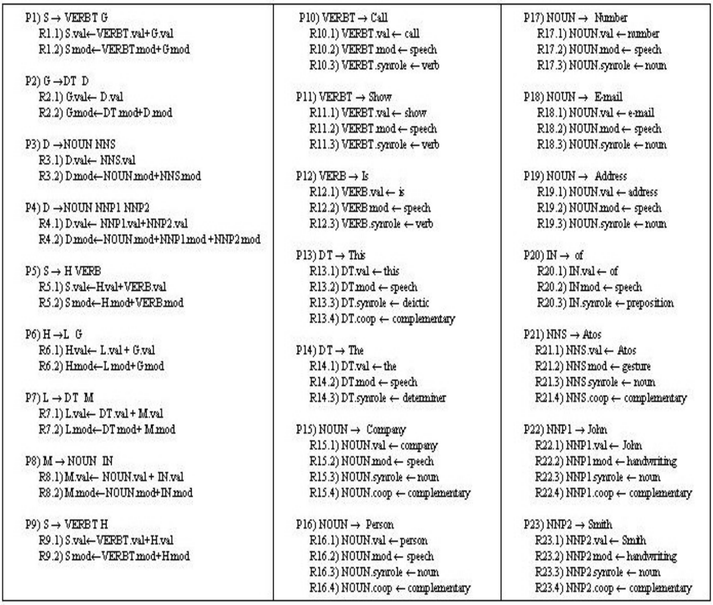
- starting from the artificial “correct” grammar G_C, the two sets of training and test sentences are generated;
- the set S_{TR} of training sentences is given as input to the grammar inference algorithm that generates the inferred grammar G_I;

- the inferred grammar is evaluated on the test set S_{TE}, i.e. the errors of omission in parsing the test sentences are measured;
- a further set S_{TE2} of test sentences is generated from the inferred grammar G_I;
- the artificial "correct" grammar is evaluated on the test set S_{TE2}, i.e. the errors of commission in parsing the test sentences are measured.

8.3.3 Evaluation Results

When the grammar inference algorithm received as input the set S_{TE} of training sentences shown in Table 8.4.a, it generated the inferred grammar that is shown in Table 8.5.

Table 8.5: The multimodal attribute grammar inferred by the algorithm

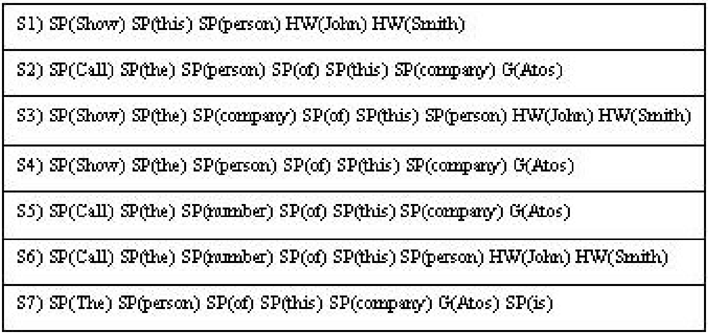


```
P1) S → VERBT G
    R1.1) S.val←VERBT.val+G.val
    R1.2) S.mod←VERBT.mod+G.mod

P2) G →DT D
    R2.1) G.val← D.val
    R2.2) G.mod←DT.mod+D.mod

P3) D →NOUN NNS
    R3.1) D.val← NNS.val
    R3.2) D.mod←NOUN.mod+NNS.mod

P4) D →NOUN NNP1 NNP2
    R4.1) D.val← NNP1.val+NNP2.val
    R4.2) D.mod←NOUN.mod+NNP1.mod +NNP2.mod

P5) S → H VERB
    R5.1) S.val←H.val+VERB.val
    R5.2) S.mod←H.mod+VERB.mod

P6) H →L G
    R6.1) H.val← L.val + G.val
    R6.2) H.mod←L.mod+G.mod

P7) L → DT M
    R7.1) L.val← DT.val + M.val
    R7.2) L.mod←DT.mod+ M.mod

P8) M → NOUN IN
    R8.1) M.val← NOUN.val + IN.val
    R8.2) M.mod←NOUN.mod+IN.mod

P9) S → VERBT H
    R9.1) S.val←VERBT.val+H.val
    R9.2) S.mod←VERBT.mod+H.mod

P10) VERBT → Call
    R10.1) VERBT.val ← call
    R10.2) VERBT.mod ← speech
    R10.3) VERBT.synrole ← verb

P11) VERBT → Show
    R11.1) VERBT.val ← show
    R11.2) VERBT.mod ← speech
    R11.3) VERBT.synrole ← verb

P12) VERB → Is
    R12.1) VERB.val ← is
    R12.2) VERB.mod ← speech
    R12.3) VERB.synrole ← verb

P13) DT → This
    R13.1) DT.val ← this
    R13.2) DT.mod ← speech
    R13.3) DT.synrole ← deictic
    R13.4) DT.coop ← complementary

P14) DT → The
    R14.1) DT.val ← the
    R14.2) DT.mod ← speech
    R14.3) DT.synrole ← determiner

P15) NOUN → Company
    R15.1) NOUN.val ← company
    R15.2) NOUN.mod ← speech
    R15.3) NOUN.synrole ← noun
    R15.4) NOUN.coop ← complementary

P16) NOUN → Person
    R16.1) NOUN.val ← person
    R16.2) NOUN.mod ← speech
    R16.3) NOUN.synrole ← noun
    R16.4) NOUN.coop ← complementary

P17) NOUN → Number
    R17.1) NOUN.val ← number
    R17.2) NOUN.mod ← speech
    R17.3) NOUN.synrole ← noun

P18) NOUN → E-mail
    R18.1) NOUN.val ← e-mail
    R18.2) NOUN.mod ← speech
    R18.3) NOUN.synrole ← noun

P19) NOUN → Address
    R19.1) NOUN.val ← address
    R19.2) NOUN.mod ← speech
    R19.3) NOUN.synrole ← noun

P20) IN → of
    R20.1) IN.val ← of
    R20.2) IN.mod ← speech
    R20.3) IN.synrole ← preposition

P21) NNS → Atos
    R21.1) NNS.val ← Atos
    R21.2) NNS.mod ← gesture
    R21.3) NNS.synrole ← noun
    R21.4) NNS.coop ← complementary

P22) NNP1 → John
    R22.1) NNP1.val ← John
    R22.2) NNP1.mod ← handwriting
    R22.3) NNP1.synrole ← noun
    R22.4) NNP1.coop ← complementary

P23) NNP2 → Smith
    R23.1) NNP2.val ← Smith
    R23.2) NNP2.mod ← handwriting
    R23.3) NNP2.synrole ← noun
    R23.4) NNP2.coop ← complementary
```

Afterwards, the performance of the inferred grammar was evaluated by testing if the sentences in S_{TE}, shown in Table 8.4.b, can be parsed by the inferred grammar. The results of this evaluation showed that the algorithm infers a grammar that is able to recognize all the sentences in the test set (error of omission = 0). This means that the inferred grammar is general enough, as it is able to recognize all the unseen sentences.

Moreover, generating top-down a set S_{TE2} of test sentences (with the same size of the training and test sets in Table 8.4.a and b) from this inferred grammar, the errors of commission can be measured. For instance, considering the set of sentences in Table 8.6, the results showed that the inferred grammar does not generate ungrammatical sentences (error of omission = 0).

Table 8.6: test sentences generated from the inferred grammar for the experiment



S1) SP(Show) SP(this) SP(person) HW(John) HW(Smith)
S2) SP(Call) SP(the) SP(person) SP(of) SP(this) SP(company) G(Atos)
S3) SP(Show) SP(the) SP(company) SP(of) SP(this) SP(person) HW(John) HW(Smith)
S4) SP(Show) SP(the) SP(person) SP(of) SP(this) SP(company) G(Atos)
S5) SP(Call) SP(the) SP(number) SP(of) SP(this) SP(company) G(Atos)
S6) SP(Call) SP(the) SP(number) SP(of) SP(this) SP(person) HW(John) HW(Smith)
S7) SP(The) SP(person) SP(of) SP(this) SP(company) G(Atos) SP(is)

In order to further validate the performance of the algorithm, fifteen other experiments were conducted, by varying the sets of training and test sentences. In particular, the fourteen generated sentences of Table 8.4 were shuffled and two new sets (with size equal to 7) were randomly defined. The results showed that the average error of omission is equal to 0.018 (only two sentences were not parsed in all the sixteen conducted experiments), while the average error of commission is equal to 0.009 (only one sentence was not parsed in all the sixteen conducted experiments).

From these simple experiments it is possible to gather that the proposed grammar inference method has an acceptable performance, since the inferred grammar has a very high probability (i.e. >0.97) of parsing valid sentences.

However, more complex experiments should be conducted in order to have a more accurate evaluation of the algorithm. In particular, the number of training and test sentences can be increased, so that the experiments can be conducted over various training and test set size. Moreover, several artificial grammars can be considered in order to have a more objective evaluation, dependent as little as possible on the chosen grammar. Experiments along this line has not been performed yet because they require a high effort for writing manually "correct" grammars following the MAG notation, but they represent a future work to give more accuracy to the performance evaluation of the algorithm.

Chapter 9 Conclusion and Future Work

Conclusion and Future Work

This chapter concludes the dissertation by summarizing the contributions which this thesis offers to the research community, and point out directions for future research.

9.1 Summary of the Thesis

Multimodal interaction has emerged in the last few years as the future paradigm of human-computer interaction. This fact is gathered also by the increasingly application of the multimodal paradigm to computer interfaces making computer behaviour closer to human communication. Multimodal interaction requires that several simultaneous inputs, coming from various input modalities, are opportunely integrated and combined into a complete sentence, i.e. a *multimodal fusion* process has to occur.

In the literature, three main different approaches to the fusion process have been proposed: the recognition-based, decision-based, and hybrid multi-level fusion. The last one contains the grammar-based fusion strategy. A comparison of these approaches [MPA06] showed that the grammar-based paradigm is the most natural one as it is more coherent with the human-human communication paradigm in which the dialogue is seen as a unique and multimodal communication act. Moreover, this approach allows an easier inter-modality disambiguation. However, the use of a grammar implies a higher computational complexity for generating the rule sets of the grammar as well as a highly expert user that is skilled in computational linguistics for writing the grammar.

In order to overcome the deficiencies of the grammar-based paradigm, this thesis proposed an approach of grammar definition that follows the “by example” paradigm, that is, the language designer provides concrete examples of multimodal sentences that have to be recognized, and a grammar inference algorithm automatically generates the grammar rules to parse those examples. In such a way no skilled grammar writers are needed, but even non-expert users can define multimodal grammars.

For verifying the workability of this theoretical approach, an innovative multimodal grammar editor has been implemented, which, unlike task-specific multimodal grammars, allows to define complex multimodal expressions, integrating whatever input modalities. A test on the usability of this editor showed that it facilitates the grammar definition and is more suitable also for non-expert people as it does not require the learning of the grammar notation. In fact, most of people involved in the test considered the MGE user-friendly and ease to use compared to a text-based editor.

9.2 Contributions

This thesis offers some contributions to the area of multimodal human-computer interaction research, which are summarized in the following of this section.

First of all, the thesis introduces a new multimodal grammar, named *Multimodal Attribute Grammar* (MAG), which is an extension of attribute grammars for multimodal input processing. This grammar has the capability to manage whatever modalities and to represent temporal constraints into the grammar rules. Moreover, it provides a good compromise between the context-free paradigm and the necessity to represent semantic and temporal aspects of multimodal input.

Secondly, a computationally efficient algorithm for grammatical inference has been defined, which join together the strengths of the inductive CYK and e-GRIDS algorithms, adapting them to multimodal sentences. The strength of this algorithm relies on its efficiency, simplicity and capability of avoiding the over-generalization problem through the introduction of a heuristics based on the simplicity of the grammar description.

Thirdly, an implementation of the underlying theory of multimodal grammars and grammar inference into the *Multimodal Grammar Editor* (MGE) has been provided. Using this editor the language designer can define grammars interactively, by expressing concrete examples of multimodal sentences, which s/he wants the system recognizes, and to define all the opportune constraints on syntactic roles and types of cooperation among modalities. Afterwards, the editor applies the grammar inference method for generating the set of production rules and the associated semantic functions, which are expressed following the MAG notation.

Finally, a validation of the proposed grammar editor has been provided by the means of a usability evaluation experiment, which compare user acceptability in using the proposed editor against a text-based editor. This experiment confirmed that the MGE, designed and implemented in this dissertation, is a valid support for the definition of multimodal grammars, mainly for people not skilled in this field. Moreover, the performance of the grammar inference algorithm has been validated through several experiments which aim to examine its ability to infer a "correct" multimodal attribute grammar. The main outcome of these early experiments is that the algorithm has an acceptable performance, since the inferred grammar has a very high probability (i.e. >0.97) of parsing valid sentences.

To summarize, the main contributions of this thesis are twofold:

- a grammatical editor for multimodal language definition that is general enough to be applicable for whatever modalities and in whichever domains,
- an efficient incremental learning algorithm that, following an approach "by example", allows to generate the production rules of the defined grammar starting from the acceptable multimodal sentences.

9.3 Future Work

This thesis is a first step into the domain of multimodal languages and grammars. In this section, some of the directions for future work in this area are presented.

As discussed in Section 5.5.2.1, a heuristic, based on the minimum description length of the grammar, was developed in order to avoid the over-generalization problem without the use of negative examples. However, the definition of such a kind of heuristics constitutes an argument not deeply investigated yet. Therefore, a promising research direction for future work can be towards finding new heuristics or towards improving the developed one. Moreover, the introduction of new learning operators, in addition to the merge and create operators, can be examined in order to enhance the way the algorithm improves the grammar description.

Another interesting task for future work is to further evaluate the proposed grammar inference algorithm. The improvements on the experimental phase are mainly twofold. First of all, experiments over larger example sets could be performed. In this dissertation, a basic evaluation over small artificial grammars was conducted. However, a more accurate evaluation of the algorithm can be achieved by increasing the number of training and test sentences, so that the experiments can be conducted over various training and test set size. Moreover, several artificial grammars can be considered in order to have a more objective evaluation, dependent as little as possible on the chosen grammar. Secondly, a comparison of the computational efficiency of the proposed grammar inference method with other existing algorithms could be performed. Such a comparison will also provide a deeper evaluation of the accuracy of grammars inferred by the proposed algorithm rather than the existing algorithms.

A more implementative aspect that can be examined as future work is the integration of further modalities into the grammar editor. Although the theoretical framework presented in this thesis (i.e. the multimodal grammar and the inference algorithm) can accommodate many input modalities, the current implementation of the editor strongly supports only speech, handwriting, sketch, and pointing gesture modalities. Future versions of the editor should support also other modalities, including lip-reading, 2D and 3D gestures, gaze tracking, and face tracking.

Appendices

Usability Evaluation

This appendix describes the procedure followed for the usability evaluation reported in Section 8.2. The instructions, which were given to participants for using the yellow and red editors, and the evaluation questionnaire are reported. The questionnaire included questions that asked the users to rate (on a 5-point Likert scale ranging from “strongly disagree” to “strongly agree”) certain features of the two editors.

Instructions for using yellow editor

Starting from the set of multimodal sentences, illustrated in the section about “The training set of multimodal sentences”, you will define the multimodal grammar by using the interface of the editor you visualize on your computer screen.

First of all, you have to set the name of the grammar, that is “PHONE-BOOK”, by pushing the button “Define a new grammar” and inserting the name “PHONE-BOOK”. Afterwards, you have to write the production rules and the associated semantic functions, which allows to parse the first multimodal sentence of the set, in the text area “Multimodal grammar acquisition” at the right side of the interface. After all rules are written, you have to push the button “Insert Rule” and the current grammar will be displayed in the “Grammar Display” area.

Afterwards, you have to select the name of the grammar you are defining (i.e. PHONE-BOOK) from the list of defined grammars and push the button “Specify the selected grammar”.

At this point, you have to write (in the text area “Multimodal grammar acquisition”) the production rules and the associated semantic functions for parsing the second multimodal sentence, taking into account the rules previously inserted and displayed on the area “Grammar Display”.

You have to repeat these steps for each of the seven sentences in the training set. When you finish writing the production rules for parsing all the sentences, the defined grammar will be shown in the area “Grammar Display”.

Instructions for using red editor

Starting from the set of multimodal sentences, illustrated in the following section about “The training set of multimodal sentences”, you will define the multimodal grammar by using the interface of the editor you visualize on your computer screen. You will wear a headset that allows you to speak to the computer. You will be equipped also with a digital pen that allows you to draw/handwrite directly on the editable area of the editor’s interface.

First of all, you have to set the name of the grammar, that is “PHONE-BOOK”, by pushing the button “Define a new grammar” and inserting the name “PHONE-BOOK”. Afterwards, you have to select the modalities you will use for expressing the first multimodal sentence (i.e. speech and handwriting modalities) from the combo box of the “Modality selection” panel, and you can confirm the selection by pressing the button “Confirm Modalities”.

At this point, you have to insert the first multimodal sentence by using the headset for the speech and the digital pen for the handwriting. The acquisition of the sentence begins with the pressure of the button “Start”. When you finish to insert the sentence, you can press the button “Stop”. If you are not satisfied by the inserted sentence you can cancel it and start again the acquisition.

Afterwards, you have to press the button “Visualize”, for visualizing the inputs recognized by the specific unimodal recognizers.

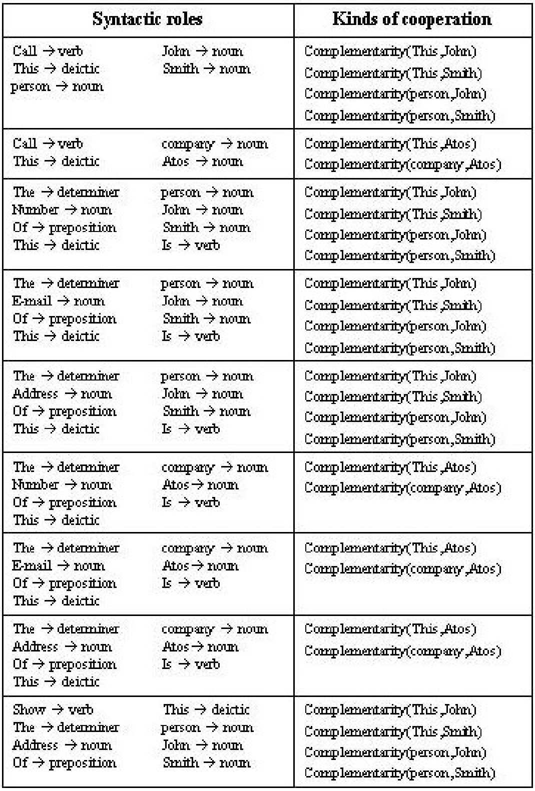
At this point you have to define the syntactic role that each element has within the sentence, by pushing the button “Define Syntactic Roles”. The system automatically shows the possible syntactic roles of the input elements. Whether these roles correspond to the roles described in column a of Table A.1, you can confirm them by pressing the button “Confirm syntactic role”, otherwise you can define the syntactic roles manually by selecting the input element and choosing the appropriate role in the drop-down list. When syntactic roles are defined for all input elements, you can visualize the defined roles in the text area on the right of the interface, and you can either confirm them by pressing the button “Send syntactic roles” or delete some syntactic roles by pressing the button “Remove syntactic role”.

Afterwards, you have to identify the kind of cooperation among input elements by pressing the button “Define Modality Cooperation”. According to the cooperation mode described in column b of Table A.1, you have to select the input elements that have to be linked by a cooperation mode (complementarity, redundancy..) and choose the appropriate mode in the drop-down list. When all necessary rules of modality cooperation are defined, you can visualize them in the text area on the right of the interface, and you can either confirm them by pressing the button “Confirm modality cooperation” or delete some cooperation rules by pressing the button “Delete modality cooperation”.

At this point you can press the button “Add Sentence” for concluding the multimodal sentence input. The system automatically applies the algorithm of grammar inference and generates the production rules that can be visualized in the text area on the left of the interface.

You have to repeat these steps for each of the nine sentences in the training set. When you finish inserting the multimodal sentences, the defined grammar will be shown in the area “Grammar Display”.

Table A.1: The syntactic roles and kind of cooperation of multimodal sentences for the usability test



Syntactic roles	Kinds of cooperation

Call → verb
This → deictic
person → noun
John → noun
Smith → noun
	Complementarity(This ,John)
	Complementarity(This ,Smith)
	Complementarity(person,John)
	Complementarity(person,Smith)

Call → verb
This → deictic
company → noun
Atos → noun
	Complementarity(This ,Atos)
	Complementarity(company ,Atos)

The → determiner
Number → noun
Of → preposition
This → deictic
person → noun
John → noun
Smith → noun
Is → verb
	Complementarity(This ,John)
	Complementarity(This ,Smith)
	Complementarity(person,John)
	Complementarity(person,Smith)

The → determiner
E-mail → noun
Of → preposition
This → deictic
person → noun
John → noun
Smith → noun
Is → verb
	Complementarity(This ,John)
	Complementarity(This ,Smith)
	Complementarity(person,John)
	Complementarity(person,Smith)

The → determiner
Address → noun
Of → preposition
This → deictic
person → noun
John → noun
Smith → noun
Is → verb
	Complementarity(This ,John)
	Complementarity(This ,Smith)
	Complementarity(person,John)
	Complementarity(person,Smith)

The → determiner
Number → noun
Of → preposition
This → deictic
company → noun
Atos → noun
Is → verb
	Complementarity(This ,Atos)
	Complementarity(company ,Atos)

The → determiner
E-mail → noun
Of → preposition
This → deictic
company → noun
Atos → noun
Is → verb
	Complementarity(This ,Atos)
	Complementarity(company ,Atos)

The → determiner
Address → noun
Of → preposition
This → deictic
company → noun
Atos → noun
Is → verb
	Complementarity(This ,Atos)
	Complementarity(company ,Atos)

Show → verb
The → determiner
Address → noun
Of → preposition
This → deictic
person → noun
John → noun
Smith → noun
	Complementarity(This ,John)
	Complementarity(This ,Smith)
	Complementarity(person,John)
	Complementarity(person,Smith)

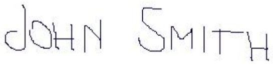
The training set of multimodal sentences

The nine training sentences that you have to insert are the following:

S1) speech: “Call this person”

handwriting: the name of the person, that is “JOHN SMITH”, on the handwriting area

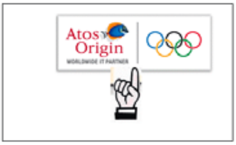
Result: After you have inserted the sentence, the area for handwriting should look like this:



while the recognized speech input (that you can visualize when you push the button “visualize”) should be “Call this person”.

S2) speech: “Call this company”
gesture: pointing the icon of the company, that is “ATOS”, on the touch- screen display

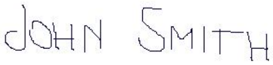
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “Call this company”.

S3) speech: “The number of this person is”
handwriting: the name of the person, that is “JOHN SMITH”, on the handwriting area

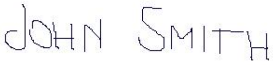
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “The number of this person is”.

S4) speech: “The e-mail of this person is”
handwriting: the name of the person, that is “JOHN SMITH”, on the handwriting area

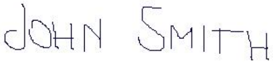
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “The e-mail of this person is”.

S5) speech: “The address of this person is”
handwriting: the name of the person, that is “JOHN SMITH”, on the handwriting area

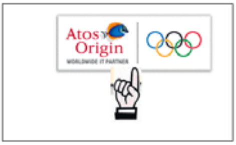
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “The address of this person is”.

S6) speech: “The number of this company is”
gesture: pointing the icon of the company, that is “ATOS”, on the touch- screen display

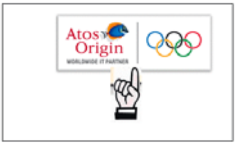
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “The number of this company is”.

S7) speech: “The e-mail of this company is”
gesture: pointing the icon of the company, that is “ATOS”, on the touch- screen display

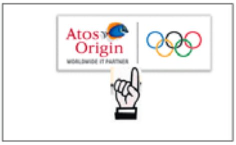
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “The e-mail of this company is”.

S8) speech: “The address of this company is”
gesture: pointing the icon of the company, that is “ATOS”, on the touch- screen display

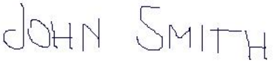
Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “The address of this company is”.

S9) speech: “Show the address of this person”
handwriting: the name of the person, that is “JOHN SMITH”, on the handwriting area

Result: After you have inserted the sentence, the pointing area should look like this:



while the recognized speech input should be “Show the address of this person”.

Evaluation questionnaire

EDITORS’ COMPARISON

1) Defining a multimodal grammar by writing the production rules and semantic functions is difficult

______ strongly disagree

______ disagree

______ neither agree or disagree

______ agree

______ strongly agree

2) Defining a multimodal grammar by specifying the examples of multimodal sentences to be generated is difficult

______ strongly disagree

______ disagree

______ neither agree or disagree

______ agree

_____ strongly agree

3) Using the yellow editor for defining a multimodal grammar is as efficient as using the red editor (in terms of time)

_____ strongly disagree
_____ disagree
_____ neither agree or disagree
_____ agree
_____ strongly agree

YELLOW EDITOR

4) I found using the yellow editor easy

_____ strongly disagree
_____ disagree
_____ neither agree or disagree
_____ agree
_____ strongly agree

5) I found writing all production rules time-consuming

_____ strongly disagree
_____ disagree
_____ neither agree or disagree
_____ agree
_____ strongly agree

6) I found the interaction with the yellow editor userfriendly

_____ strongly disagree
_____ disagree
_____ neither agree or disagree
_____ agree
_____ strongly agree

RED EDITOR

7) I found using the red editor easy

_____ strongly disagree
_____ disagree

_____ neither agree or disagree
_____ agree
_____ strongly agree

8) I found writing all multimodal sentences time-consuming

_____ strongly disagree
_____ disagree
_____ neither agree or disagree
_____ agree
_____ strongly agree

9) I found the interaction with the red editor userfriendly

_____ strongly disagree
_____ disagree
_____ neither agree or disagree
_____ agree
_____ strongly agree

Bibliography

[Adr92] Adriaans, P. *Language Learning from a Categorical Perspective*. Ph.D. thesis, Universiteit van Amsterdam. 1992.

[AFG08] Avola, D., Ferri, F., Grifoni, P., Paolozzi, S. *A Framework for Designing and Recognizing Sketch-Based Libraries for Pervasive Systems*. UNISCON 2008, LNBIP, pp. 405-416. 2008.

[Ang81] Angluin, D. *A Note on the Number of Queries Needed to Identify Regular Languages*. Information and Control (51), pp.76–87. 1981.

[Ang82] Angluin, D. *Inference of reversible languages*. Journal of ACM, vol. 29, pp. 741-765. 1982.

[Ang88] Angluin, D. *Queries and concept learning*. Machine Learning. Vol. 2, pp. 319-342. 1988.

[Ang90] Angluin, D. *Negative results for equivalence queries*. Machine Learning Journal (5), 121–150. 1990.

[APS98] Andre, M., Popescu, V.G., Shaikh, A., Medl, A., Marsic, I., Kulikowski, C., Flanagan J.L. *Integration of Speech and Gesture for Multimodal Human-Computer Interaction*. In Second International Conference on Cooperative Multimodal Communication. 28-30 January, Tilburg, The Netherlands. 1998.

[Bak79] Baker, J. K. *Trainable grammars for speech recognition*. In: D. H. Klatt and J. J. Wolf (eds.): Speech

Communication Papers for the 97th Meeting of the Acoustical Society of America. pp. 547–550. 1979.

[Bar02] Barnum, C. M. *Usability Testing and Research.* New York: Longman Publishers, pp 9, 10, 147. 2002.

[BlM95] Blattner, M. M., Milota, A. D. *Multimodal interfaces with voice and gesture input*, Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (ICSMC '95) (Vancouver, Canada), vol. 3, October 1995. 1995.

[BNB04] Bouchet, J., Nigay, L., Balzagette, D. *ICARE: Approche à composants pour l'interaction multimodale*. Actes des Premières Journées Francophones: Mobilité et Ubiquité 2004, Nice Sophia-Antipolis, France, June 2004, pp 36-43. 2004.

[BNG04] Bouchet, J., Nigay, L., Ganille, T. *Icare software components for rapidly developing multimodal interfaces*, ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces (New York, NY, USA), ACM, pp. 251-258. 2004.

[Bol80] Bolt, R. *Put-that-there: Voice and gesture at the graphics interface*. Computer Graphics, 14(3), pp. 262-270. 1980.

[Bou03] Bourguet, M.L. *Designing and Prototyping Multimodal Commands.* Proceedings of Human-Computer Interaction (INTERACT'03), pp. 717-720. 2003.

[CMU] Carnegie Mellon University. CMU pronouncing dictionary. [Online]. Available: http://www.speech.cs.cmu.edu/cgi-bin/cmudict

[Car92] Carpenter, B. *The logic of typed feature structures*. Cambridge University Press, Cambridge, England. 1992.

[CBB94] Cohen, D., Berke, L., Bloom, P., Cohen, D., Tsur, D. *The role of knowledge mining in the development and evolution of new applications*, Proceedings of the 10th International Conference on Data Engineering (Houston, TX) (Ahmed K. Elmagarmid and Erich Neuhold, eds.), IEEE Computer Society Press, pp. 166-167. 1994.

[CGN90] Crimi, C., Guercio, A., Nota, G., Pacini, G., Tortora, G., Tucci, M. *Relation grammars for modelling multi-dimensional structures*. In IEEE Symposium on Visual Languages, pages 168–173. IEEE Computer Society Press. 1990.

[Cho57] Chomsky, N. *Syntactic Structures*, The Hague Mouton. 1957.

[CJM97] Cohen, P.R., Johnston, M., McGee, D., Oviatt, S.L., Pittman, J., Smith, I.A., Chen, L., Clow, J. *Quickset: Multimodal interaction for distributed applications*, ACM Multimedia, pp. 31-40. 1997.

[CMB03] Corradini, A., Mehta M., Bernsen, N.O., Martin, J.-C. *Multimodal Input Fusion in Human-Computer Interaction on the Example of the on-going NICE Project.* In Proceedings of the NATO-ASI conference on Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management, Yerevan, Armenia. 2003.

[CoC91] Coutaz, J., Caelen, J. *A Taxonomy For Multimedia and Multimodal User Interfaces*. In Proceedings of the 1st ERCIM Workshop on Multimedia HCI, November 1991, Lisbon. 1991.

[CoO91] Cohen, P. R., Oviatt, S. L. *Discourse structure and performance efficiency in interactive and noninteractive spoken modalities*, Computer Speech and Language, no. 5(4), 297-326. 1991.

[Den01] Denis, F. *Learning Regular Languages from Simple Positive Examples*, Journal of Machine Learning, vol. 44, pp. 37-66. 2001.

[Den98] Denis, F. *PAC learning from positive statistical queries*, M. M. Richter, C. H. Smith, R. Wiehagen and T. Zeugmann (eds), Proceedings of the 9th International Conference on Algorithmic Learning Theory (ALT-98), Berlin Springer Vol. 1501 of LNAI, pp 112-126. 1998.

[ElR92] Elhadad, M., Robin, J. *Controlling content realization with functional unification grammar*. In Proc. of the 6th

International Workshop on NLG. Springer, Lecture Notes in AI. 1992.

[EST96] Emerald, J. D., Subramanian, K. G., Thomas, D. G. *Learning Code regular and Code linear languages*. Proceedings of International Colloquium on Grammatical Inference (ICGI-96), Lecture Notes in Artificial Intelligence 1147, Springer-Verlag, pp. 211-221. 1996.

[FoL98] Fodor, J., Lepore, E. *The Emptiness of the Lexicon: Reflections on James Pustejovsky's The Generative Lexicon*. Linguistic Inquiry 29: 269-288. 1998.

[GaV90] Garcia, P., Vidal, E. *Inference of K-testable languages in the strict sense and applications to syntactic pattern recognition*. Journal of IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12(9), pp. 920-925. 1990.

[Gol67] Gold, E. M. *Language identification in the limit*. Information and Control 10.447-474. 1967.

[Gru96] Grunwald, P. *A minimum description length approach to grammar inference*. In S. Wemter, E. Riloff, and G. Scheler, editors, Symbolic, Connectionist and Statistical Approaches to Learning for Natural Language Processing, Lecture Note inAI. Springer Verlag, pages 203-216. 1996.

[Gup03] Gupta, A. *An adaptive approach to collecting multimodal input*, Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics. 2003.

[Hew92] Thomas T. H. *Acm sigchi curricula for human-computer interaction*. Technical report, New Jersey Institute of Technology, New York, NY, USA, 1992.

[HiO03] de la Higuera, C., Oncina, J. *Identification with Probability One of Stochastic Deterministic Linear Languages*. In: Proceedings of ALT 2003. Berlin, Heidelberg, pp. 134–148, Springer-Verlag. 2003.

[HMO91] Helm, R., Marriott, K., Odersky, M. *Building visual language parsers*. In Proceedings of Conference on

Human Factors in Computing Systems: CHI '91, ACM Press, New York, 105–112. 1991.

[JARNAL] http://www.dklevine.com/general/software/tc1000/jarnal.htm

[JCM97] Johnston, M., Cohen, P.R., McGee, D., Oviatt, S.L., Pittman, J.A. *Unification-based multimodal interaction*, in Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain, pp. 281-288. 1997.

[JoB00] Johnston, M., Bangalore, S. *Finite-state Multimodal Parsing and Understanding*, In Proceedings of the International Conference on Computational Linguistics, Saarbruecken, Germany. 2000.

[JoB05] Johnston, M., Bangalore, S. *Finite-state multimodal integration and understanding*. Nat. Lang. Eng. 11, 2 (Jun. 2005), 159-187. 2005.

[Joh98] Johnston, M. *Unification-based Multimodal Parsing*. Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL '98), August 10-14, Université de Montréal, Montreal, Quebec, Canada. pp. 624-630. 1998.

[Kas65] T. Kasami. *An efficient recognition and syntax analysis algorithm for context-free languages*. Science Report, Air Force Cambridge Research Laboratory, Bedford MA. 1965.

[Kay79] Kay M. *Functional grammar*. In Proceedings of the Fifth Meeting of the Berkeley Linguistics Society, pages 142–158, Berkeley, CA. 1979.

[KeL97] Keller, B., Lutz, R. *Evolving Stochastic Context-Free Grammars from Examples Using a Minimum Description Length Principle*. In Worksop on Automata Induction, Grammatical Inference and Language Acquisition, Nashville, Tennessee, USA, ICML097. 1997.

[KMT97] Koshiba, T., Makinen, E., Takada, Y. *Inferring pure context-free languages from positive data*, Technical report A-1997-14, Department of Computer Science, University of Tampere. 1997.

[Knu68] Donald E. Knuth. *Semantics of context-free languages,* Mathematical Systems Theory 2, 127–145. 1968.

[LaS00] Langley, P., Stromsten, S. *Learning Context-Free Grammars with a Simplicity Bias*. Proceedings of the Eleventh European Conference on Machine Learning (ECML 2000), Lecture Notes in Artificial Intelligence 1810, Springer-Verlag, pp. 220-228, Barcelona, Spain. 2000.

[Lee96] Lee, L. *Learning of context-free languages: A survey of the literature*, Tech. Report TR-12-96, Harvard University. 1996.

[LeM92] Levy, E.T., McNeill, D. *Speech, gesture, and discourse*. Discourse Processes, (15):277-301. 1992.

[MGA01] Martin, J. C., Grimard, S., Alexandri, K. *On the annotation of the multimodal behavior and computation of cooperation between modalities*. Proceedings of the Workshop on Representing, Annotating, and Evaluating Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents, Montreal, Canada. pp.1-7. 2001.

[MPA06] Manchón, P., Pérez, G., Amores, G. *Multimodal Fusion: A New Hybrid Strategy for Dialogue Systems.* In Proceedings of Eighth International Conference on Multimodal Interfaces (ICMI 2006), Banff, Alberta, Canada. ACM: New York, , pp. 357-363. 2006.

[MSM94] Mitchell P. M., Santorini, B., Marcinkiewicz, M.A. *Building a large annotated corpus of english: The penn treebank.* Computational Linguistics, 19(2), pp 313-330. 1994.

[NaI00] Nakamura, K., Ishiwata, T. *Synthesizing context free grammars from sample strings based on inductive cyk* *algorithm*, ICGI '00: Proceedings of the 5th International Colloquium on Grammatical Inference (London, UK), Springer-Verlag. pp. 186-195. 2000.

[Nak03] Nakamura K. *Incremental learning of context free grammars by extended inductive cyk algorithm*, ECML Workshop on Learning Contex-Free Grammars (Colin de la Higuera, Pieter W. Adriaans, Menno van Zaanen, and José Oncina, eds.), Ruder Boskovic Institute, Zagreb, Croatia. pp. 53-64. 2003.

[NaM02] Nakamura, K., Matsumoto, M. *Incremental learning of context free grammars*, ICGI '02: Proceedings of the 6th International Colloquium on Grammatical Inference (London, UK), Springer-Verlag. pp. 174-184. 2002.

[NeS91] Neal, J. G., Shapiro, S. C. *Intelligent multimedia interface technology*. In J. Sullivan & S. Tyler (Eds.), Intelligent User Interfaces, New York: ACM Press., pp.11-43. 1991.

[NiC95] Nigay, L. Coutaz, J. *A generic platform for addressing the multimodal challenge*, in the Proceedings of the Conference on Human Factors in Computing Systems, ACM Press. 1995.

[OC00] Oviatt, S.L., Cohen, P.R. *Multimodal interfaces that process what comes naturally*. Communications of the ACM, 43, no. 3, 45-53. 2000.

[OCW00] Oviatt, S.L., Cohen, P.R., Wu, L., Vergo, J., Duncan, L. et al. *Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions*. Human-Computer Interaction, vol. 15, pp. 263-322. 2000.

[ODK97] Oviatt, S. L., DeAngeli, A., Kuhn, K. *Integration and synchronization of inputmodes during multimodal human-computer interaction*. In Proceedings of Conference on Human Factors in Computing Systems, 415–422. 1997.

[OMG01] OMG. *UML Unified modeling language specification*. http://www.omg.org/uml, 2001.

[Ovi02] Oviatt, S. L. *Multimodal interfaces.* in Handbook of Human-Computer Interaction, (ed. by J. Jacko & A. Sears), Lawrence Erlbaum: New Jersey. 2002.

[Ovi04] Oviatt S.L., Coulston R., Lunsford R. *When Do We Interact Multi-modally?* Cognitive Load and Multi-modal Communication Patterns, Proceedings of ICMI. pp. 129-136. 2004.

[Ovi96] Oviatt, S. L. *Multimodal interfaces for dynamic interactive maps.* In Proceedings of Conference on Human Factors in Computing Systems, 95–102. 1996.

[PAM05] Pérez, G., Amores, G., Manchón, P. *Two strategies for multimodal fusion*. In Proceedings of Multimodal Interaction for the Visualization and Exploration of Scientific Data, Trento, Italy, 26–32. 2005.

[PBH97] Pavlovic, V.I., Berry, G.A., Huang, T.S. *Integration of audio/visual information for use in human-computer intelligent interaction*. Proceedings of the 1997 International Conference on Image Processing (ICIP '97), Volume 1, pp. 121-124. 1997.

[PeW80] Pereira, F., and Warren, *D.H.D. Definite Clause Grammars for Language Analysis - A survey of the Formalism and a Comparison with Augmented Transition Networks*, Artificial Intelligence, vol. 13, no. 3. 1980.

[PPK04] Petasis, G., Paliouras, G., Karkaletsis, V., Halatsis, C., Spyropoulos, C.D. *e-GRIDS: Computationally Efficient Grammatical Inference from Positive Examples*. GRAMMARS, (7), pp. 69 – 110. 2004.

[Pul03] Pullum, G. K. *Learnability*. In the second edition of The Oxford International Encyclopedia of Linguistics, 431-434. Oxford: Oxford University Press. 2003.

[QTG00] Quesada, J. F., Torre, D., Amores, G. *Design of a Natural Command Language Dialogue System*. Deliverable 3.2, Siridus Project. 2000.

[RPC04] Reitter, D., Panttaja, E. M., Cummins, F. *UI on the fly: Generating a multimodal user interface*. In Proceedings of HLT-NAACL-2004, Boston, Massachusetts, USA. 2004.

[Ris78] Rissanen, J. *Modeling by shortest data description*. Automatica, 14:465–471. 1978.

[RSH05] Russ, G., Sallans, B., Hareter, H. *Semantic Based Information Fusion in a Multimodal Interface*. International Conference on Human-Computer Interaction (HCI'05), Las Vegas, Nevada, USA, 20-23 June, pp 94-100. 2005.

[SaB02] Salvador, I., Benedı, J.-M. *RNA Modeling by Combining Stochastic Context-Free Grammars and n-Gram Models*. International Journal of Pattern Recognition and Artificial Intelligence 16(3), 309–316. 2002.

[Sak97] Sakakibara, Y. *Recent Advances of Grammatical Inference.* Theoretical Computer Science 185, 15–45. 1997.

[SBH94] Sakakibara, Y., Brown, M., Hughley, R., Mian, I., Sjolander, K., Underwood, R., Haussler D. *Stochastic context-free grammars for tRNA modeling.* Nuclear Acids Res. 22, 5112–5120. 1994.

[SCS06] Sun, Y., Chen, F., Shi, Y.D., Chung, V. *A novel method for multi-sensory data fusion in multimodal human computer interaction.* In Proceedings of the 20th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artefacts and environments, Sydney, Australia, 401-404, 2006.

[Shi86] S.M. Shieber. *An Introduction to Unification-Based Approaches to Grammar*. CSLI Publications. 1986.

[ShT95] Shimazu, H., Takashima, Y. *Multimodal Definite Clause Grammar.* Systems and Computers in Japan 26(3):93-102. 1995.

[SNC95] Schomaker, L., Nijtmans, J., Camurri, A., Lavagetto, F., Morasso, P., Benoit, C., Guiard-Marigny, T., Le Goff, B., Robert-Ribes, J., Adjoudani, A., Defee, I., Munch, S., Hartung, K., Blauert, J. *A Taxonomy of Multimodal Interaction in the Human Information Processing System.* Multimodal Integration for Advanced Multimedia Interfaces (MIAMI). ESPRIT III, Basic Research Project 8579. 1995.

[SPH98] Sharma, R., Pavlovic, V. I., Huang, T. S. *Toward Multimodal Human-Computer Interface*. Proceedings of the IEEE, special issue on Multimedia Signal Processing, 86(5), pp 853-869. 1998.

[SSC07] Sun, Y., Shi, Y., Chen, F., and Chung, V. *An Efficient Multimodal Language Processor for Parallel Input Strings in Multimodal Input Fusion.* In Proceedings of the international Conference on Semantic Computing (September 17 - 19, 2007). ICSC. IEEE Computer Society, Washington, DC, pp. 389-396. 2007.

[StB03] Steedman, M. Baldridge, J. *Combinatory categorial grammar*. Unpublished tutorial, School of Informatics, Edinburgh University. ftp://ftp.cogsci.ed.ac.uk/pub/steedman/ccg/manifesto.pdf. 2003.

[Ste00] Steedman, M. *The syntactic process*, Cambridge Massachusset: the MIT press. 2000.

[StS05] Stivers, T., Sidnell, J. *Introduction: Multimodal interaction*. *Semiotica,* 156(1/4), pp. 1-20. 2005.

[TKM03] Toutanova, K., Klein, D., Manning, C., Singer, Y. *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*. In Proceedings of HLT-NAACL 2003, pp. 252-259. 2003.

[TSW90] Tappert, C. C., Suen, C. Y., Wakahara, T. *The state of the art in online handwriting recognition*, IEEE Trans. Pattern Anal. Mach. Intell.12, no. 8, pp.787-808. 1990.

[VaB87] Vanlehn, K., Ball, W. *A version space approach to learning context-free grammars*. Machine Learning, 2(1), pp. 39-74. 1987.

[Val84] Valiant, L. *A theory of the learnable*. Communications of the ACM, 27 (11), pp. 1134-1142. 1984.

[Zaa01] van Zaanen, M. *Bootstrapping structure into language: alignment-based learning*. PhD thesis, School of Computing, University of Leeds. 2001.

[Vo98] Vo, M.T. *A framework and Toolkit for the Construction of Multimodal Learning Interfaces*, PhD. Thesis, Carnegie Mellon University, Pittsburgh, USA. 1998.

[VoW96] Vo, M.T., Wood, C. *Building an application framework for speech and pen input integration in multimodal learning interfaces.* In Proceedings of the Acoustics, Speech, and Signal Processing (ICASSP'96), May 7-10, IEEE Computer Society, Volume 06, pp. 3545-3548. 1996.

[WLK04] Walker, W., Lamere, P.., Kwok, P., Raj, B., Singh, R., Gouvea, E., Wolf, P., Woelfel, J. *Sphinx-4: A flexible open source framework for speech recognition*. Technical Report TR2004-0811, SMLI, Carnegie Mellon University, SUN MICROSYSTEMS INC. 2004.

[WRB01] Wahlster, W., Reithinger, N., Blocher, A. *SmartKom: Multimodal Communication with a Life-Like Character*, Proceedings of Eurospeech, Aalborg, Denmark, 2001.

[WWT91] Wittenburg, K., Weitzman, L., Talley, J. *Unification-Based grammars and tabular parsing for graphical languages*. Journal of Visual Languages and Computing 2, pp. 347–370. 1991.

[Wol82] Wolff, G. *Language Acquisition, Data Compression and Generalisation,* Language and Communication, 2, pp. 57-89. 1982.

[Yok95] Yokomori, T. *On Polynomial-Time Learnability in the Limit of Strictly Deterministic Automata*. Journal of Machine Learning, vol. 19, pp. 153-179. 1995.