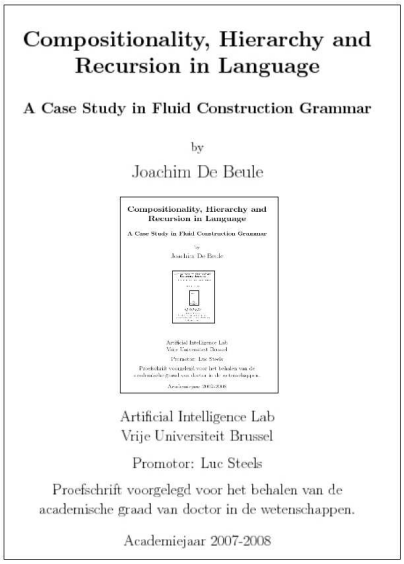
Compositionality, Hierarchy and Recursion in Language

A Case Study in Fluid Construction Grammar

by

Joachim De Beule



Artificial Intelligence Lab
Vrije Universiteit Brussel

Promotor: Luc Steels

Proefschrift voorgelegd voor het behalen van de academische graad van doctor in de wetenschappen.

Academiejaar 2007-2008

Acknowledgements

For you probably the beginning, for me finally the end, for at last I find myself faced with the hardest part of writing this thesis: the acknowledgment section. Not only hard because this will presumably be the part most frequently and perhaps firstly read, but more so because I find myself incapable of ever being satisfied with it: there always seem to be things left unsaid or people forgotten or not given the appropriate amount of credit.

Let me therefore start the impossible with expressing my gratitude to all the many people who have influenced and supported me in any way or another during the time leading to the work you are now holding in your hands.

One of the most important and influential people who had an unmeasurable impact on me and this work was of course my promoter, Luc Steels. Even though he runs two labs, plays the piano, is one of our most internationally respected and influential contemporary scientist and still finds the time to write plays that are performed in Paris and Avignon, he was almost always available for me to discuss wild ideas, to give me guidance and take care of me in many, many ways. I would like to thank him sincerely for believing in me, for the amazing amount of opportunities he gave me, not only the financially related ones but also simply for giving me the opportunity to work with him: Thank you for showing me how to be a scientist, and for being someone who was almost like a father figure to me sometimes.

Many other people have of course influenced me and helped me in many ways. One of them was Bart De Vylder with which it was always a pleasure to work and an amazement to see the crispness of his mind, as well as the optimism with which any next second of the day is always considered a pleasant and happily solved puzzle soon after. In terms of relatives among the Brussels AI lab family you are a brother to me.

Of course also many other members of the Brussels and Paris AI family have played more or less pivotal roles in this work. With Jelle I always felt there was a kind of deep friendship, and it is a shame that I haven't been able to put more time and energy into it. Martin, Wouter, Remi, Michael, Ben, Pieter, Bart, Joris, Bart, Wout, Bart, Joris, Bart, Tony, Bart, we should have played

more naming games instead of table-tennis, it was a pleasure working with all of you!

I would like to thank Bart de Boer especially for putting so much energy in reading the previous version of this thesis, and for making so many suggestions that significantly have contributed to the improvement of this work; I apologize for not having been able to incorporate all of them, but to my defense I would like to offer that you simply play in a higher league.

I also want to thank the other members of the commission, Walter Daelemans, Ann Nowé, Bernard Manderick and Dirk Vermeir, for their enthusiasm and making it so rough for me at the private defense, you convinced me that there is more to it and gave me the energy to make the necessary improvements.

Then of course there is my other family, my friends. Mom, Dad, you helped me make what I am, and nothing can ever be perfect, but in any case: I couldn't have done it without you, and still. Also, Jona, Wout, Lisa, Veronique, Judith, Sien, Isolde, Myrthe, Gerard. It feels strange to see you all listed together like that, but not as it is hard to tell you how much I love you all, each in different ways, but none less than the other. The same goes for Jan Bart and his master Jacques, Imannuel and Dirk, you could have done this better and you do.

The past few years haven't always been the easiest ones for me, which made it sometimes hard to put things in perspective. Roxanne, Jo, Debbe, Karel, Vanessa, Joke, Barbara... Thank you for being there and supporting me, you may not realize it but you all have played essential parts in my life!

To the others whose name may not be explicitly mentioned, please try to forgive me, and keep in mind that I will be the one regretting having forgotten you. Thank you to you all!

Ghent, October 22.

Summary

In this thesis I argue that a number of universal features of human languages can be explained as being emergent properties of the complex dynamics governing the establishment and (cultural) evolution of a language in a population of interlocutors, rather than being the result of a genetic selection process leading to a specialized language faculty that imposes those features upon language. It is argued that this happens mainly on an intra-generational time scale, with multi-generational mechanisms only having a second order effect.

In particular, I focus on compositionality, hierarchy and recursion, generally acknowledged to be universal features of human languages. Together, by combining words into hierarchical phrases which can then recursively be combined into larger phrases, they allow the construction of an unlimited number of phrases using only a limited number of words.

Previously, there have been two major hypothesis about why all languages share these features: nativism and iterated learning. Nativism explains these features as being dictated by a genetically encoded language faculty. They do not satisfactory explain however what this language faculty precisely is or what evolutionary pathway could have led to it. The iterated learning hypothesis is also unsatisfactory, mainly because it neglects to take into account intra-generational effects and the function of language.

Instead, in functional emergentism it is argued that the mere urge for successful communication can explain why language becomes compositional, hierarchical and recursive: because these features allow the re-use of already established bits of language which in turn increases the chance that at least part of the message is conveyed successfully. To support this claim, a language game experiment is presented in which a number of artificial agents bootstrap a communication system. It is shown that although the agents are not innately wired towards compositional or recursive language and without a learning bottleneck (e.g. in the form of a flux in the population), compositional and recursive language still emerges.

Contents

v

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Compositionality, Hierarchy and Recursion

In this thesis it is argued that a number of universal features of human languages should be explained as being emergent properties of the complex dynamics governing the establishment and (cultural) evolution of a language in a population of interacting language users, rather then being the result of a genetic selection process leading to a specialized language faculty that imposes those features upon language. It is argued that this happens mainly on an intra-generational time-scale with multi-generational mechanisms only having a second order effect.

In particular, we will focus on compositionality, hierarchy and recursion, generally acknowledged to be universal features of human languages. Together, by combining words into hierarchical phrases which can then recursively be combined into larger phrases, they allow the construction of an unlimited number of sentences using only a limited number of words. With only the words "big" and "ball", all of the phrases "big ball", "big big ball", "big big big ball" etc. can be formed.[1] Compositionality refers here to the fact that the phrase "big ball" can be decomposed into two constituents "big" and "ball", and that the meaning of the complete phrase is some function of the meaning of its composing constituents. Hierarchy refers to the fact that the resulting combination "big ball" is a genuine constituent itself, ready to be used as a single unit by other constructions. Although internally this unit is decomposable into several parts ("big" and "ball"), to the outside it behaves in the same way as the single unit "ball". The new unit can be viewed as 'one level up' in the hierarchy of the

[1] These are not all representative examples of everyday English, where probably something like "very big ball" or "huge ball" would be preferred instead. However, the examples are syntactically correct according to the rules of English, and therefore can be used to illustrate the phenomena described in a relatively simple way.

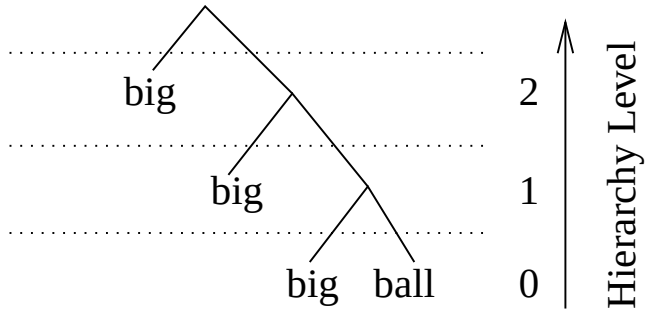


Figure 1.1: Hierarchy and recursion in the English phrase "big big big ball".

phrase, as is illustrated in figure 1.1. Recursion then refers to the fact that the rules for combining the units "big" and "ball" into a new unit can recursively be applied to the units "big" and "big ball" etc. It is easy to see that recursion requires (some degree of) hierarchy and that hierarchy requires (some degree of) compositionality.

There are a number of reasons why compositionality, hierarchy and recursion could be beneficial for language users. As already pointed out, they allow to construct an unlimited number of phrases with only a limited number of words and constructions. This power of language was already recognized by von Humboldt (1836) who stated that language "makes infinite use of finite means". Hence, even if a speaker never expressed a certain meaning before, she will probably be able to do so anyway by combining known bits of language into a larger phrase. And even though a hearer might never have heard the sentence before, she still will be able to interpret it. As the world constantly changes and since speakers and hearers constantly encounter new situations, this ability is a major advantage. In addition, because of compositionality, language users only need to *learn* and *remember* a limited number of things (i.e. the words and constructions of the language). The less things one needs to learn and negotiate about, the faster you will learn them, the less memory is needed and the faster a consensus can be reached.

1.2 Possible Explanations

There have been two major hypothesis about why all languages share these features: nativism and iterated learning.

1.2.1 Nativism

The nativists claim that humans are equipped with a language faculty (LF). (Chomsky, 1980; Pinker and Bloom, 1990; Bickerton, 1991; Jackendoff, 1999; Nowak et al., 2001; Jackendoff, 2002; Hauser et al., 2002; Jackendoff and Pinker, 2005).) Part of the LF is the language acquisition device (LAD). It is claimed that the LF is genetically encoded and passed on from one generation to the next. The LAD enables children to learn a language almost effortlessly. However, it only allows a limited set of possible languages to be learned. While learning, certain parameters are set that determine which particular instantiation of a so called universal grammar (UG) actually gets learned (Pinker, 1994). What all UG's apparently have in common is that they dictate language to be compositional and recursive. However, how this happens or how this came to be the case is left unspecified.

To answer these questions would require a detailed model of the LF and the LAD, explaining how they work, e.g. which parameters there are, how they get set and so on. However, the precise nature of the LF is very controversial, ranging from a large but very specific set of principles and parameters (Hyams, 1986; Radford, 1990) to maybe as little as the capacity for recursion (Hauser et al., 2002). In any case, how or where the LF is encoded in our genes or implemented in our brains is unknown. Moreover, in recent years, the evidence supporting a genetically encoded language faculty has become increasingly small. It even appears to be impossible to find any single capacity needed for language that is unique to humans (see e.g. Gentner et al. (2006).) All this makes it even harder to determine the nature and workings of the hypothesized LF.

An analysis at the phylogenetic level also doesn't seem to help, partly because there is no fossil data to build upon. But it is also very hard to explain the LF and UG as the result of natural selection in the first place. Some claim that they are an exaptation, something recruited for language but not specifically evolved for it (Chomsky, 1982; Premack, 1985; Mehler, 1985; Chomsky, 1988; Gould, 1987; Piattelli-Palmari, 1989). Others claim, mainly based on the argument of design, that they are an adaptation, something selected for and specialized by the process of natural selection (Lieberman, 1984; Pinker and Bloom, 1990; Jackendoff, 1999, 2002).

One of the reasons that makes it difficult to explain (the form of) human languages as mainly the result of a genetic selection process is the fact that language and communication serve intrinsically *social* and *cooperative* functions. Natural selection on the other hand would quickly lead to opportunistic and parasitic behavior undermining the evolution of a complex communication system. There have been many proposals to solve this problem, most of which can be reduced to linking language to some sexual or mating advantage (see e.g. Bickerton (1998);

Pinker and Bloom (1990); Lightfoot (1991); Miller (2001)). There is however no actual evidence to support this and language has totally different characteristics compared to things that indeed are linked to sexual advantage (e.g. mainly males should develop it instead of females, language competence starts earlier than sexual matureness and even decreases during puberty, etc. see Fitch (2004), where it is argued that the function of language initially is to be found in kin-relationships and only later received its more general function according to the reciprocal altruism evolutionary mechanism.)

In sum then, although nativists would claim that modern languages are compositional and recursive because this is dictated by a genetically encoded language faculty, nobody knows what this language faculty looks like or which mechanism it uses to enforce those features upon language. In other words, no real explanation is provided. Therefore, in this thesis, the problem will be tackled from a different angle. We too will presuppose certain capacities, like the capacity for recursion. However, we will not assume that this capacity has evolved specifically for language, thereby avoiding the difficulties that come with this assumption. Instead, we will start from the assumption that language users are cooperative and social beings, aiming at successful communication. If desired, they may use their capacity for recursion to increase their communicative skills, without *having* to do so. But *if* they do, then the resulting language will be recursive too.

1.2.2 Iterated Learning (IL)

Another approach claims that modern languages are compositional (and maybe hierarchical and recursive) because they are the result of an iterative learning process in the presence of a learning bottleneck (see Kirby (2000, 2001); Brighton and Kirby (2001); Conway and Christiansen (2001); Brighton (2002); Christiansen and Kirby (2003); Smith et al. (2003a) and others.)

The idea is the following. Language is passed on from one generation to the next. Each next generation learns the language from the previous one from example utterances. However, since the number of possible utterances is virtually infinite, only a limited set of them is available to learn from. There is, in other words, a *learning bottleneck.* This is not necessarily a problem, because many language constructs are productive in some sense. Assume for example that a learner on one occasion observes the word "red" used in combination with the word "ball" for describing a red ball. On another occasion he sees it combined with the word "bicycle" for describing a red bicycle. Then he can conclude that the word "red" can also be combined with other words describing things, as long as they are red. Hence, based on only two examples involving the word "red", the learner can produce many more utterances describing red

things. This is the power of compositional encoding.

If however different, holistic words would be used each time a red object is described, i.e. one holistic word is used for describing red balls, and another for describing red bicycles, then no generalization would be possible. In other words, when a language is passed on many times from one generation to the next, then holistic or, more general, *less learnable* bits of language will get filtered out while compositional or *more learnable* bits will survive to the next generation. As a result the language will progressively become compositional because that is the only learnable solution to the problem of iterated learning in the presence of a learning bottleneck. A similar argument can be made for recursion since this too is a productive element of language.

This is certainly an attractive alternative to the nativist explanation because it does not rely on the presence of an innate, language specific device in the human brain. Another attractive aspect of the approach is that several computational experiments have been performed, showing that the mechanism indeed works. This means that, in contrast to the nativist explanation, the required learning algorithms are known and were made explicit by implementing them on a computer for use in a simulation in which an artificial language indeed becomes compositional after being learned by the algorithms in an iterative fashion (see e.g. one of the references mentioned in the beginning of this section.)

On the other hand, the proposed mechanisms can only explain the emergence of compositionality on a multi-generational time scale. In contrast, humans are capable of evolving compositional language in the course of only one or two generations (see e.g. Senghas and Coppola (2001).) Moreover, in a typical computational iterated learning simulation, only one teacher and one learner are considered per generation. After having learned the teacher's language, the learner becomes a teacher himself for a new learner and so on. Learning also proceeds in a strictly one way fashion. Put blatantly, the teacher doesn't care whether the learner actually understands any of his example utterances. There is in other words no communication going on, only learning. And not until the language becomes learnable can communication occur, since before that the learner isn't capable of figuring out the teacher's language. This only happens after many generations.

It has been argued that taking into account also other, more functional aspects of language, could not invalidate the argument made. Even if only one teacher and learner are considered, the argument goes, and even if there is no actual communication, only one way learning, the simulations show that iterated learning is a shaping force of language that can explain the emergence of compositionality. The problem with this is that the function of language and the need for efficient communication also act as shaping forces of language, and

on a much faster timescale compared to iterative learning. We will show that if these are taken into account, compositionality emerges already in the course of one generation of language users only.

1.3 Functional Emergentism

The existing approaches, either nativist or iterated learning, heavily depend on specific properties of the language acquisition process to actively guide new language users towards compositional and recursive language, either because they are genetically programmed for it or because the learning mechanism selects for compositional language in the presence of a learning bottleneck and hence language itself will adapt to it after many learning iterations.

In other words, the role of the language learners and users (from now on called agents) is kept minimal, and language is seen almost as a sterile thing without function. We take a more functional and usage-based stance. We see language as a complex adaptive system (Steels, 1997), in which all interlocutors contribute and actively shape the language. They may refuse, adopt or propose new elements of language because they want to improve their communicative skills which in turn depends on the skills of other agents *and on whether they collectively succeed in reaching a consensus about how to express things.* All this already happens in the course of one generation of language users only.

So imagine that a population of social agents wants to bootstrap a communication system from scratch, without there being any central control or authority to decide on how things should be done. At each time step, two agents are randomly selected from the population and are faced with a communication task. For example, both agents get to see some scene and one of the agents has to describe it to the other. The second agent in turn has to signal whether he agrees with the description. We'll call an interaction like this a *language game* (Steels, 1995).

The agents start from scratch, so in the beginning agents have to invent new words and communication will fail. After a while however, it will become possible to reuse previously introduced words. It might even occur that several words could be reused. For example, assume that the scene that is to be described is about a lion approaching, and that the agent can choose between on the one hand a holistic description of the scene, i.e. using one word meaning 'a lion is approaching', and on the other a compositional phrase, i.e using two words meaning 'lion' and 'approaching' respectively. If another scene shows a sleeping lion, the word meaning 'lion' will be reusable, and by using it the probability of having at least a partially successful communicative interaction

will increase. Hence, we see that because holistic words can only be used in fewer situations compared to compositional words, and hence because holistic words will normally take longer to propagate across the population, it pays to use compositional words to increase the probability of (partial) communicative success.

A similar mechanism applies for hierarchical and recursive constructions. In short, the mere urge for successful communication can explain why languages become compositional, hierarchical and recursive: because these features allow us to reuse already established or acquired bits of language which in turn increases the probability that at least part of the message is conveyed successfully. This will be the main contribution of this thesis, and in the following chapters we will show how a computational experiment can be set up to support this claim.

To be clear, note that, according to the proposed mechanism, agents do not converge on a compositional language because they *prefer* compositional constructs over holistic ones. They are allowed to use whatever means available, compositional or holistic, hierarchical or not. The important thing is that, at all times, they try to optimize their chance of having a successful communicative interaction, and therefore prefer those bits of language they already acquired and were able to use successfully in the past. If this turns out to be a holistic construct then they will use it.

This means that in principle it is possible that the final language to which the agents converge still is holistic to some degree. This can be the case for several reasons. First, the emergence and evolution of language are stochastic processes. Hence, there is always a possibility that a consensus is reached about some holistic construct before it is suppressed by a compositional one. This will be more probable the more frequent the holistic construct can be used. It may even be beneficial to have ready-made and short but specific idiomatic expressions for frequently re-occurring situations because they are easily recognized, require less effort to construct and parse etc. This is a second reason why the resulting language might still be holistic to some extent and also is a possible explanation for frequency effects found in natural languages. This will be briefly touched upon in Chapter 7.

We also do not claim that multi-generational effects can have no influence at all on the shape of language. For example, the frequency effects mentioned in the previous paragraph clearly will also be governed by a multi-generational dynamics, especially when an idiomatic phrase survived it merely by chance (i.e. for stochastic reasons.) A flux in the population will filter out such constructs, because there will be too few occasions for newcomers to learn them, much as it is the case in the iterated learning model. Still, the point is that the agents do

not explicitly favor certain features of language like compositionality because they are more general or easier to learn. They favor those constructs that are estimated to optimize the communicative success, independent of whether the construct is a holistic or a compositional one.

1.4 Overview and Contributions

The rest of this thesis is organized into two parts. The first part gradually builds up the machinery needed to perform the language game experiment that will be presented in the second part to support the claim about the emergence of compositional and recursive language. It consists of chapters 2 to 4 and appendices A and B.

The basic setup of the experiment will consist of a population of artificial language users, called agents. Each agent is endowed with general (i.e. not necessarily language specific) learning mechanisms and the capacity to parse and produce both holistic and recursive language constructs. They start of with empty language inventories. By engaging in pairwise interactions they may tryout already acquired bits of language and, if necessary, propose new ones. There is no central authority, so different agents will propose different and sometimes conflicting language constructs. Hence, one of the main tasks faced by the agents is to reach a consensus through negotiation. Some of the problems related to this task have already been studied in detail previously, and an overview of some of the relevant work will be given in chapter 2 (specifically in section 2.3.) This chapter also introduces a number of basic notions that will prove useful for the rest of the thesis. Finally it presents a cross-situational learning algorithm that was developed by the author together with Bart De Vylder and which is used by the agents in the experiment. This chapter does not cover anything that has to do with grammar.

Chapter 3 introduces *Fluid Construction Grammar* (FCG), which is a general unification based formalism built specifically for implementing language games that do involve grammar. Some extensions to FCG will be presented, mainly consisting of a number of special operators used extensively in all of the current work on FCG. In particular, Chapter 4 introduces the so called *J-operator*, a powerful operator enabling the handling of hierarchy and recursion in FCG. As it turns out, this operator also allows to solve other imperfections of FCG.

The second part then covers the specifics of the language game experiment that was implemented to support the claim regarding the emergence of compositional and recursive language. It consists of chapters 5 to 8 and is entirely due to the author.

Chapter 2

The Language Game Framework

As explained, one way to study the emergence and evolution of language and its features is using the concept of language games. This term, inspired by Wittgenstein (1953), was introduced into the field by Steels (see e.g. Steels (1996)). In a typical language game experiment, pairs of robotic or software agents are subsequently placed before a communication task. While solving the task the agents may need to extend or change their language. In the end, all agents should be able to communicate with one another. Essential is that there is no central control, language is seen as a complex adaptive system (Steels, 1997).

Over the years, many such experiments have been carried out. Depending on the research question sometimes grounded and embodied robotic agents were used (see e.g. Steels and Vogt (1997); Vogt (2000); Steels (1999); Steels and Loetzsch (2007)) and sometimes purely computational experiments using only simulated agents were carried out to investigate many different aspects of language (see e.g. de Boer (1999, 2000b,a) on vowel systems, de Boer and Kuhl (2001, 2002) on infant-directed speech, Belpaeme (2001) and Steels and Belpaeme (2005) on color category formation, Steels et al. (2002) and Van Looveren (2001) on the origins of word-meaning, Van Looveren (2003) on determiners, Van Looveren (2000) on multiple-word naming games, Steels and Loetzsch (2007) on perspective alignment and spatial language, De Beule and De Vylder (2005) on the influence of language on conceptualization, De Beule (2006) on tense, De Beule et al. (2006) on homonymy, De Beule and Bergen (2006) on compositionality, Steels and Wellens (2006) on grammar and search in parsing, etc.)

Recently, there have also been a number of more theoretical language game studies, in particular concerning the naming game (see e.g. Lenaerts et al. (2005); Baronchelli et al. (2006a,b); De Vylder and Tuyls (2006)). Some of the relevant results will be discussed in this chapter.

2.1 Basic Notions

A single language game normally involves two interlocutors trying to communicate some aspect of the world to one another, thereby adapting themselves in order to become better at it. Depending on the focus of the particular experiment and on the technical advancedness of e.g. the robotic equipment used, different aspects of the communication task can be abstracted away and different types of language games can be defined.

Typically, one of the interlocutors in a game is called the speaker agent. The other is called the hearer agent. All agents in a population can function as speaker or hearer, but at the beginning of a particular game the two agents involved are assigned their role for the remaining of the game. Both agents are then presented a scene or setting about which the game will be played. For example in the talking heads experiment (Steels, 1999), an agent consisted of a camera connected to a computer processing the image data and running the rest of the agent program, and both the speaker and the hearer's cameras were focused towards a region on a white board showing geometrical figures of various shape and color (see figure 2.1.)

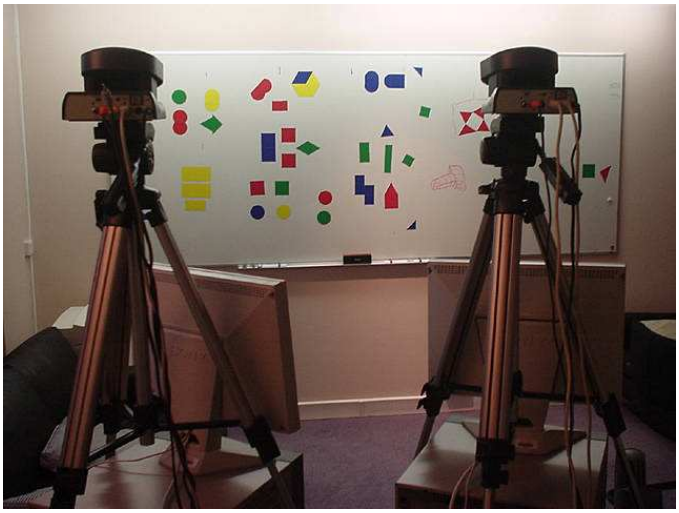


Figure 2.1: The Talking Heads experiment setup at Sony CSL, Paris, 2003.

The part of the world or scene relevant to and observable by the agents during a game is called the context. In the talking heads it is the part of the white-board covered by the camera image. The context contains a number of possible topics for the game. In case of the talking heads this would be one of the colored figures included in the context.

2.1.1 Speaker Agents

Figure 2.2 shows a general architecture of a speaker agent. Rectangular blocks

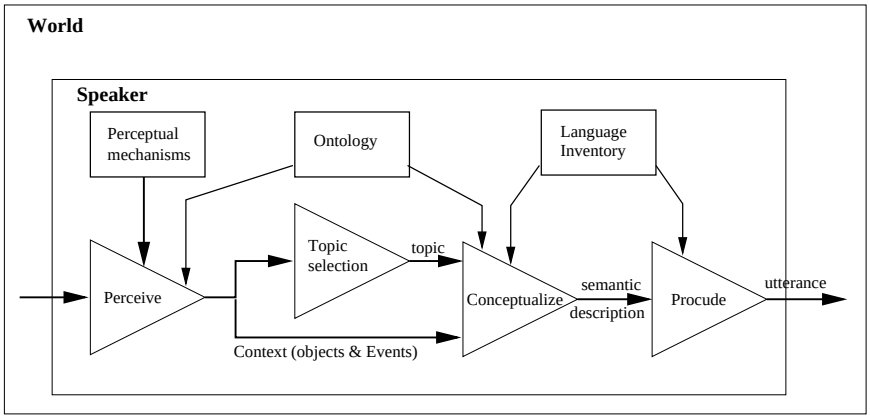


Figure 2.2: General architecture of the speaker part of an agent.

represent parts of the state of the agent, for example its ontology, which could be a list of conceptual categories, or its language inventory, which could be a bidirectional associative memory linking categories to words. Triangular blocks represent processing parts, they take some input and calculate some output using the agent's internal state.

The speaker observes the world (the scene) with its sensory devices. The data coming from these devices typically is continues, like the rgb values for all the pixels in the camera image and this for subsequent images over time. An agent is equipped with built-in perceptual mechanisms that pre-process this data and transforms it to a more abstract domain.

For example, the Talking Heads were equipped with edge and region detection algorithms allowing them to detect the objects in the image as coherently colored regions. They were also equipped with additional mechanisms to classify the observed objects according to a certain set of features (average color, circularity, number of corners, ...) In this case the result of perception is a set of category-channel values (numbers), one for each channel (like the number of corners) per object.[1]

Next, the speaker selects a topic for the game from the context (see the topic selection process in figure 2.2.) He then should decide on what aspects of the

[1]Abstract concepts like 'circle' or 'blue object' are not yet defined at this stage, they have to be learned. The inventory of concepts an agent eventually learns is called the agent's ontology.

topic should best be used for describing it. For example, if it is the only red object in the context then a good description probably should include this fact.[2] This is called conceptualization.

The result of conceptualization is a semantic description which, in our case, will always be a conjunction of first order logic predicates. A semantic description is the basic meaning that needs to be verbalized. How this is done will be explained later.

2.1.2 Hearer Agents

Figure 2.3 shows the architecture for the hearer part of an agent. The perceptual

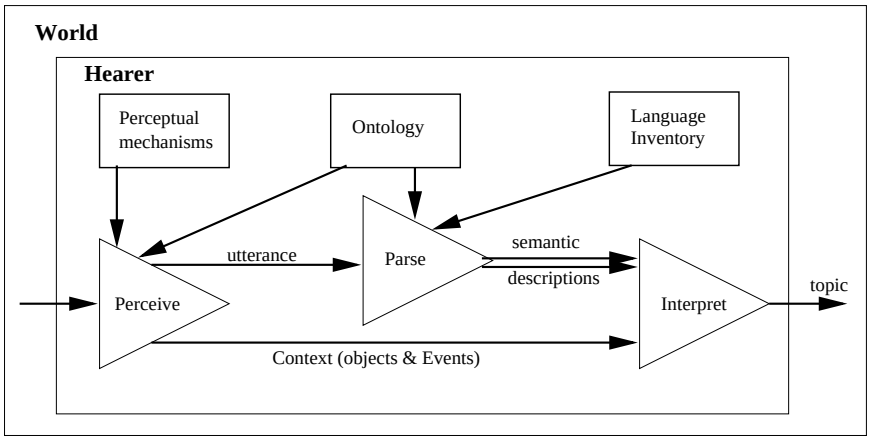


Figure 2.3: General architecture of the hearer part of an agent.

mechanisms, ontology and language inventory are the same as in the speaker case (an agent can function both as a hearer and as a speaker.) Perceptual output now also includes the utterance observed, which is parsed into a semantic description. This semantic description can then be interpreted in the world, leading to the hearer's guess about what the object or event was that the speaker choose as topic.

Finally, both agents receive feedback about whether the communication was successful.

Thus the general flow of a language game as considered in this work is as in figures 2.4 and 2.5. Steps one and six represent the perception and topic

[2]Here is an example of how abstract concepts could be learned in a usage-based way: if the speaker can't find a good description he might consider extending his ontology.

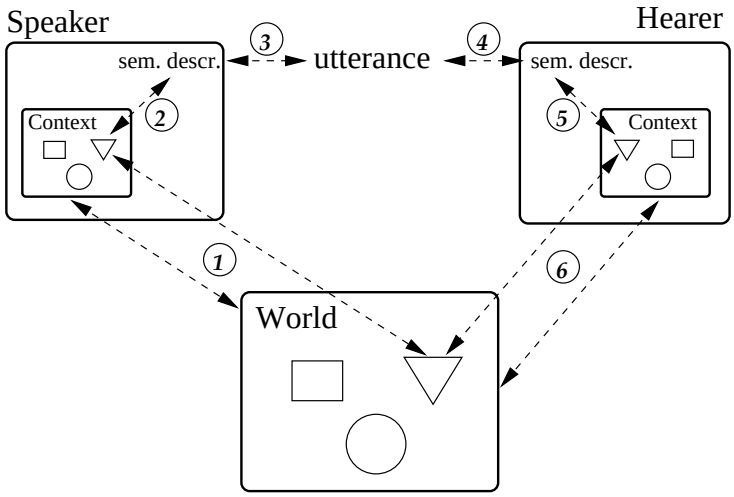


Figure 2.4: Flow of a Language game (1).

selection actions. These result in agent-internal representations of the context. The speaker's topic is part of the context.

Step two is the speaker's conceptualization action, in which a semantic description is constructed for the chosen topic. Conceptualization might depend on the context, on the available conceptual categories and and on the available language constructions for expressing them. In some experiments (but not in this thesis) this step may also involve the invention of new categories when no adequate description can be found.

In step three the description is transformed into an utterance which is observable to the hearer agent. This typically involves a choice between competing analysis based on expected success as estimated from previous interactions.

The hearer then parses the utterance resulting in a set of (competing) semantic descriptions called hypotheses which he tries to interpret in the context (steps four and five).

Finally, depending on whether how this works out, he sends the speaker a certain non-verbal success signal.

The speaker and hearer architectures and the different steps in a language game as introduced here allow the investigation of many aspects of language. In Chapters 7 and 8, a number of such experiments will be presented. The assumptions and simplifications made compared to the general agent architectures presented here will be explicitly stated. Other experiments (not reported on in this thesis) make other assumptions and focus on other aspects.

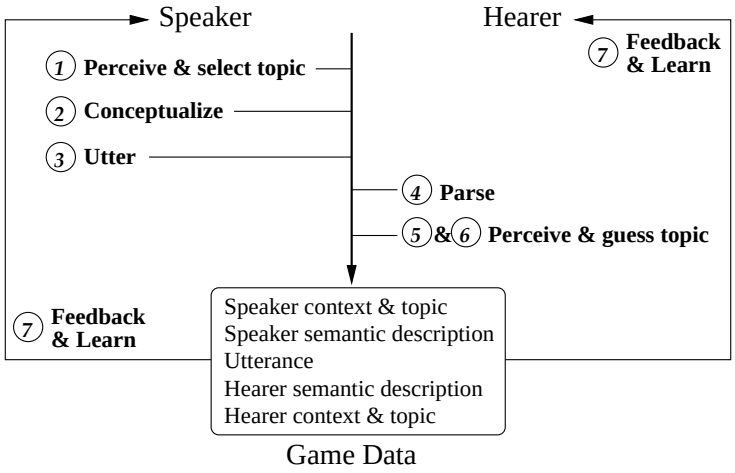


Figure 2.5: Flow of a Language Game (2).

2.2 The Semiotic Landscape

The notion of semiotic graphs allow us to define a number of concepts and measures that can be used to monitor language game experiments.

2.2.1 Semiotic Graphs

Basically, in every game there are three different spaces involved which have to do with the real world (or scene), how it is perceived, and how it is described respectively. During a game, a speaker first performs a mapping from world space into (his private) conceptual space. Then he performs a second mapping from conceptual space to utterance space. The hearer in turn performs a mapping from utterance space to conceptual space and from conceptual back to world space.

In every experiment certain assumptions are made about the structure and elements of each of these spaces, as well as about the possible mappings between them. For example, real word robots equipped with camera's or other sensorimotor devices play language games about real-world objects and events. Still, for complexity and controlling reasons, the robots are only exposed to a restricted portion of the real world: some restricted arena containing objects of only a certain color or shape, sometimes only under well controlled lighting conditions, etc.

In this thesis, the mapping from world space to category space is abstracted away. As will be explained in chapter 5, all experiments presented will involve

agents that are endowed with a universal ontology and it is assumed that every agent perceives a scene in exactly the same way. Therefore only the mapping between conceptual and utterance space is relevant to us.

Hence, we can define a semiotic graph as consisting of a set of nodes in conceptual space, a set of nodes in utterance space, and a set of weighted and directed links (or edges or arrows) between them. Every conceptual node represents a different semantic description (a 'meaning'), and every utterance node a different utterance. An arrow going from a conceptual to an utterance node represents a production (verbalization.) An arrow going in the other direction represents a parse (interpretation.)

The meaning of the weight of a link depends on how the graph is constructed: with or without looking into the agents' heads. If one would simply observe all games, continuously recording which topic is chosen, how it is verbalized etc, and construct the semiotic graph on the basis of these recordings, then the weights represent the probabilities of a certain meaning being verbalized as (or being the interpretation of) a certain utterance in the particular population of agents observed. Such a graph can tell you for example what the probability is that an arbitrary agent would be understood correctly if he would have to verbalize an arbitrary meaning. It does not tell you anything however about how the agent would do this, or about which other possible verbalizations he was considering.

When you are allowed to look into the agents' heads, more can be recorded. For example, the number of synonyms the agent *knows* for a particular meaning but *never uses*.. Weights will then represent the probability that an agent associates a certain meaning with a certain form and vice versa.

2.2.2 Measures on a Semiotic Graph

As mentioned, semiotic graphs can tell you things about the state of the agents playing language games. One thing that will be crucial for this is the concept of the effective out-degree of a node in the graph.

Effective Out-Degree

The effective out-degree of a node is related to the number of its outgoing edges, but takes into account the weights of the edges. We assume that the weights of the outgoing edges are normalized such that their sum equals 1. If there are k edges each having an equal probability $1/k$, then the effective out-degree equals k. If however the k edges have differing probabilities (weights), then the effective out-degree should be smaller. Moreover, if one of the edges has a probability close to one, the effective out-degree should also be close to one

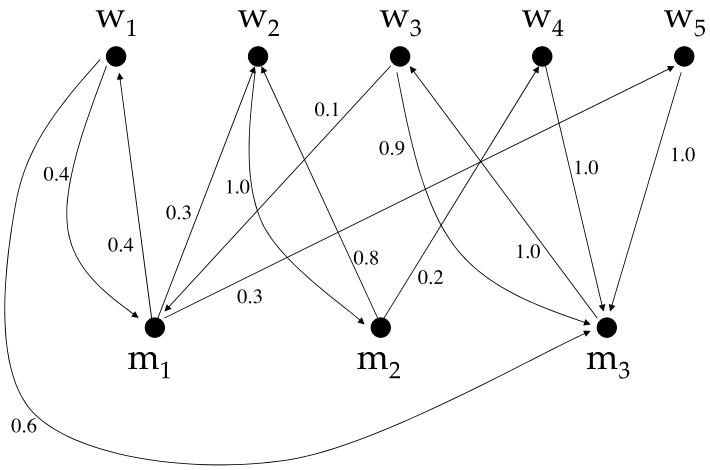


Figure 2.6: An example of a semiotic graph representing the observable state of a population at some point in time. The nodes marked m_i represent a meaning (a conjunction of predicates), those marked w_i represent utterances. The weights on the edges refer to probabilities of observing a certain production or interpretation.

(but still slightly higher). Therefore we define the effective out-degree of a node as the number of edges with equal probability needed for a hypothetical node to have the same associated Shannon information as the original node. More precisely, if a node has k outgoing edges with weights $x_i, 1 \leq i \leq k$, its Shannon information is given by

$$\sum_{i=1}^{k} -x_i \log(x_i).$$

A node with k' outgoing edges with equal probability $1/k'$ thus has a Shannon information of $\log(k')$. By definition this information should be equal to the information associated with the original node, from which it follows that

$$k' = \exp(\sum_{i=1}^{k} -x_i \log(x_i)).$$

The effective out-degree k' is not necessarily integer. For example, the effective out-degree of m_1 in figure 2.6 is 2.97 and of w_3 it is 1.38.

(Generalized) Synonymy

Synonymy exists when a conceptual node is associated with several utterance nodes. Strictly speaking this is called synonymy only when the points in utterance space involved represent single words, i.e. synonyms. Figure 2.7 illustrates this.

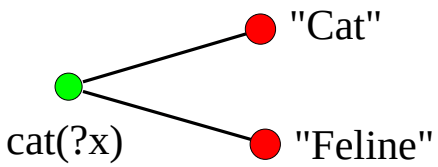


Figure 2.7: Synonymy as represented in a semiotic graph: both the words "cat" and "feline" are shown to map to the same (conceptual) meaning.

The synonymy of a semiotic graph is defined as the average of the effective out-degree of its meaning nodes minus 1. The synonymy of the graph in figure 2.6 is thus 0.87.

Possible sources of synonymy include:

- Wrong guess by the hearer of the meaning of a word, resulting in the adoption of an additional word for a concept which was already associated with another word.
- Different words proposed by different agents for the same concept.

Both mechanisms will occur frequently in the experiment of chapter 5. Under more realistic circumstances, for example when there is noise during the transmission of utterances, additional sources of synonymy are imaginable, but these will not be considered.

(Generalized) Homonymy

Figure 2.8 illustrates homonymy. Again, one normally only speaks of homonyms when the points in utterance space represent words.

The homonymy of a semiotic graph is defined as the average of the effective out-degrees of the word nodes minus 1. The homonymy of the graph in figure 2.6 is thus 0.27.

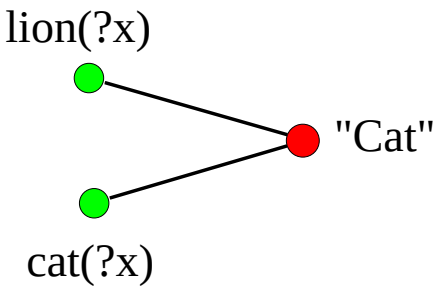


Figure 2.8: Homonymy as represented in the semiotic landscape.

There are at least two mechanisms that introduce homonymy:

- Two agents could independently propose the same word for a different concept or, equivalently, one agent could propose the same word for two different concepts. These possibilities are excluded in this thesis: the possibility of re-inventing an existing word is zero.
- The meaning of an unknown word could be uncertain and hence used differently than originally intended. This will happen frequently in the experiments presented.

Return and Communicative Success

The return of a semiotic graph is the probability that one returns to a random meaning node after taking two steps from it (thus first going to an utterance node and then back to a meaning node), with the probability of taking an edge equal to the edge's weight. For example, in figure 2.6, the chance of returning to m_1 starting from it is $0.4 \times 0.4 = 0.16$, for m_2 it is $0.8 \times 1.0 = 0.8$ and for $m_3 = 1.0 \times 0.9 = 0.9$. Hence, the return of this graph is the 0.62. If a hearer does not take the context into account to guess the meaning of a given utterance, then the return equals the communicative success.[3]

2.3 Developmental Stages

Language games come in different flavors, and it has proved useful for our research to classify them according to a number of development stages of increasing complexity (although no claim is made about any human-related develop-

[3]Strictly speaking this only holds if the semiotic graphs of the speaker and hearer are the same.

mental stages.) The different stages all fit naturally into the language game framework. The more simple stages involve purely lexical language with grammar only kicking in later. Because the more simple stages already provide some lessons and insights useful for games involving grammar, we will briefly discuss some of them.

2.3.1 The Naming Game

The simplest type of language game is the naming game. It is used to study the development of a purely lexical language without grammar or syntax, only proper names. Moreover, the possibility for homonymy is also excluded. Agents talk about uniquely identifiable objects, like people, but unlike chairs: if you switch two people's location, they take their identity with them, and the different names that were given to them can still be used to identify them.

The defining characteristics or assumptions of the naming game are as follows:

- The world consists of O uniquely identifiable objects.
- An agent's ontology consists of a set of uni-referential categories, each category uniquely identifying one and only one object in the world.
- Utterances are single words, i.e. labels or proper names.
- Every time step, two agents are chosen randomly from a population of N agents, one is appointed the role of speaker and the other of hearer.
- The speaker selects a topic from O and describes it with a word (a name).
- The hearer guesses what the topic was. If the guess is incorrect then the speaker is able to reveal the correct topic by pointing to it.

Under these assumptions it suffices to have $|O| = 1$ (i.e. the world contains one object only) since all objects are uniquely identifiable and there can be no confusion about the topic in the learning phase. Hence, no homonymy can arise. The game and agent architecture are depicted schematically in figure 2.3.1, which is much simpler compared to figures 2.2 and 2.3.

The simplest solution to the naming game consists of letting all agents remember every word they hear. The agents start of with empty lexicons, so initially no words are known and new words must be introduced.

The (average) dynamics of this setup can be studied analytically: they are governed entirely by the probability of picking a speaker that didn't play a game before and by the probability of picking a hearer that doesn't know a

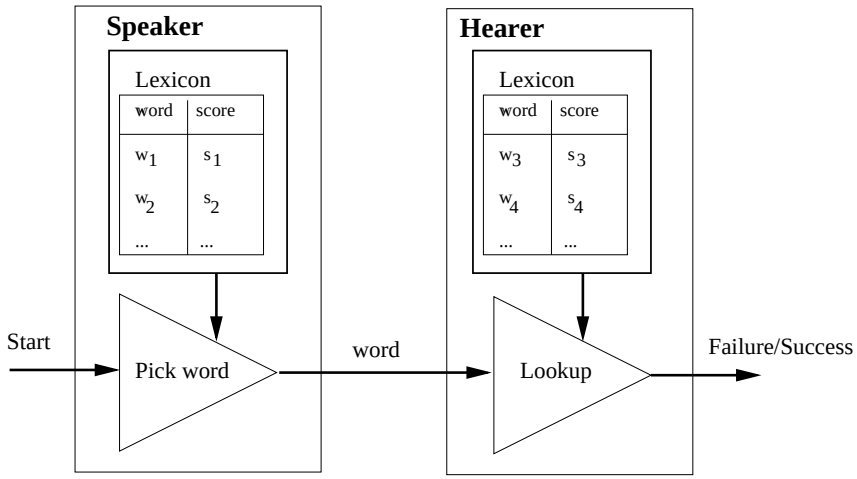


Figure 2.9: Speaker and hearer agent architecture under naming game conditions.

word yet (see equations (2.1) and Figure 2.10, with n_a the population size, $A(t)$ the number of agents that was involved in at least one game at time t, $W(t)$ the total number of different words proposed at time t, $I(x)$ the total number of words known by all agents together, $R(t)$ the return and $S(t)$ the synonymy at time t).

$$\begin{aligned} A(t) &= n_a \left[1 - (1 - \frac{2}{n_a})^t\right], \\ W(t) &= A(t)/2, \\ I(t+1) &= \frac{(n_a - 1)(2n_a - A(t))A(t)}{2(A(t) - 1)} \\ &\quad \times \left[1 - \left(1 - \frac{2(A(t) - 1)}{n_a(n_a - 1)A(t)}\right)^{t+1}\right], \\ R(t) &= \frac{2I(t)}{n_a^2}, \\ S(t) &= \left(\frac{n_a}{2}\right)^{\frac{2W(t)}{n_a}} - 1. \end{aligned} \tag{2.1}$$

Synonymy Reduction

As can be seen, in the simplest solution all agents eventually learn and remember every word ever introduced (on average this is about $n_a/2$ words). When this

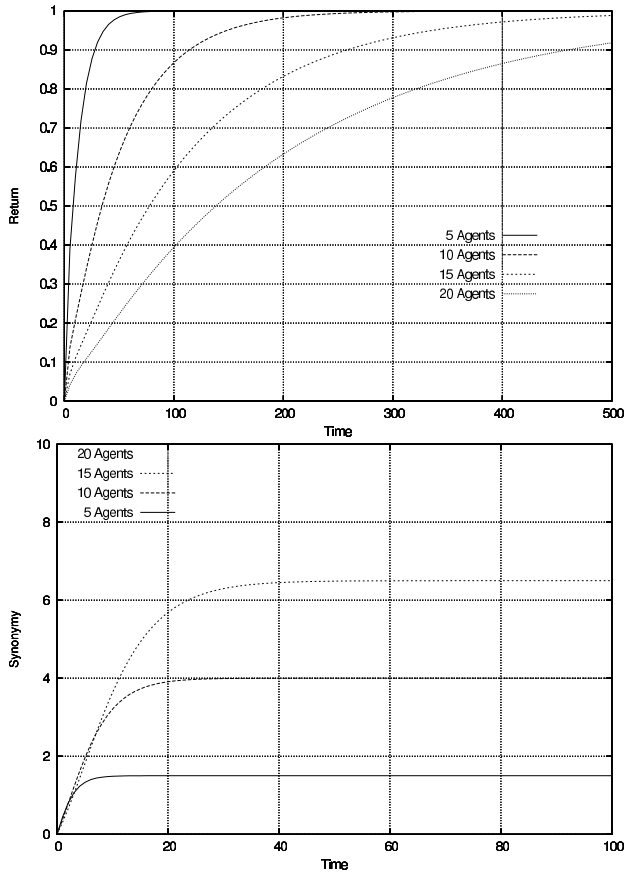


Figure 2.10: Evolution of the return (top) and synonymy (bottom) according to equations (2.1) for different population sizes.

happens the return evidently becomes maximal and nothing changes anymore. In particular, the synonymy remains positive.

What is missing is some synonymy damping mechanism: the agents need to agree upon a preference among all the synonyms floating around in the population. One such mechanism is lateral inhibition: every word w is associated with a synonymy score $\phi(w)$. Whenever an agent hears a word he enforces the word's associated score according to:

$$\phi(w) \leftarrow \theta + (1-\theta)s\phi(w), \tag{2.2}$$

with $\theta \in]0,1[$ a learning parameter.[4] The words of all competing synonyms $w' \neq w$ (all the other words in the agent's lexicon) are inhibited according to:

$$\phi(w') \leftarrow (1-\theta)\phi(w'). \tag{2.3}$$

[4]Note that there is enforcement only for $\theta > 0.5$.

Speakers prefer to use words with a higher score.

This setup cannot be analyzed analytically. The effect of lateral inhibition is illustrated in Figure 2.11. As can be seen, convergence is speed up and the

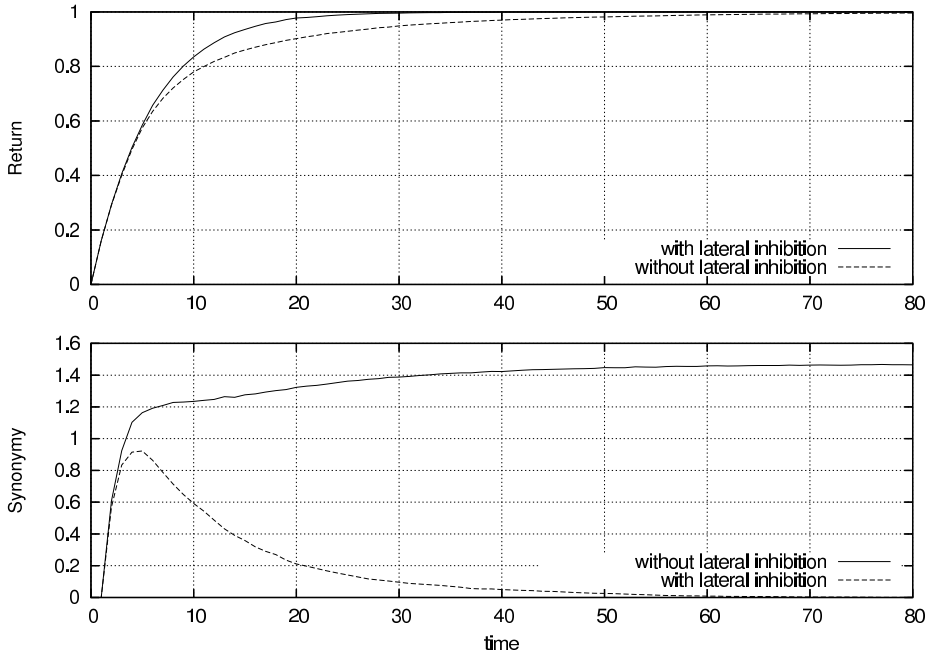


Figure 2.11: Illustration of the effect of lateral inhibition on the evolution of the return (top) and synonymy (bottom). (Both graphs were obtained experimentally for a population of 5 agents. Note that the labels are reversed across graphs. In both graphs they are such that the top label corresponds to the top curve.)

agents now do manage to agree upon which word to use (synonymy becomes 0.)

The take home message here is that a preference mechanism like lateral inhibition is needed to allow agents to reach agreement amongst a number of competing elements of language.

2.3.2 The Guessing Game

Like the naming game, the guessing game is also about lexical languages. The main difference is that now a hearer doesn't know the topic.

This can be because the topic is not a uniquely identifiable object but might be only an aspect (a feature) of an object, like it's color or shape. This setup opens up the possibility of a hidden underlying dynamics at the conceptual level

which is coupled with the visible negotiation dynamics about what words to use. We will not go into this here, the interested reader is referred to Steels (1999); Steels and Loetzsch (2007); De Beule and De Vylder (2005) and others.

A similar situation arises when the hearer doesn't know which of a number of (uniquely identifiable objects) is described by the speaker. The defining characteristics are then identical to those of the naming game, except that after the game the speaker doesn't reveal which of the objects in the context was the topic. Because the hearer might associate an unknown word with the wrong object, the possibility for homonymy arises and the agents need to perform some sort of cross situational learning.

Cross Situational Learning

Consider a game in which a population of n_a agents has to bootstrap a common lexicon for some set O of objects. As usual, at each time-step t a speaker and a hearer are randomly selected from the population. They are both presented with the current context $O_t \subset O$ which always contains a random subset of at least two objects from O. The speaker randomly selects one of them as the topic, but his choice is not revealed to the hearer.

Thus, each time step t, the only information transmitted between a speaker and a hearer is the fact that the speaker produced a particular word w for one of the objects in the context O_t. This information is insufficient to determine the intended meaning of the word w and cross-situational learning is required.

If the hearer was to learn a stable language, he could wait until the word w is observed again at time $t' > t$, concluding that the meaning of w should be in $O_t \cap O_{t'}$ and so on. However, because new words are introduced and because the preferred meaning of existing words may change, this strategy may fail because of inconsistent observations that reduce $O_t \cap O_{t'}$ to the empty set. Thus, a more intelligent cross-situational learning scheme is required, capable of estimating a word/meaning mapping that changes over time.

The information consists of consecutive $\langle w, O_t \rangle$ pairs of a word w and a set of objects O_t of which one is apparently referred to by w. Previously, Smith (2003b) proposed a Bayesian learning mechanism that estimates the probability of some object o occurring given the occurrence of a word w in a similar setting. Basically, it consists of storing all co-occurrences of words and meanings. One thing that can be regarded as a negative aspect of such a mechanism is that it will become less and less sensitive to changes in the learning target over time.[5] Moreover, it insufficiently uses available information. For example, if an

[5]However, under certain circumstances this could equally be regarded as a positive thing. For example, in section 7.4, simulation results will be presented that involve a population turnover. As it turns out, if teacher agents remain their flexibility towards newcomers in

agent at time t observes a word w_1 with a context $O_t = \{o_1, o_2\}$ and, at some later time step t', observes the same word with context $O_{t'} = \{o_1, o_3\}$, it seems logical to conclude that the meaning of w_1 should be o_1. However, only taking co-occurrences into account results in w_1 referring to o_1 with a probability of 0.5 and either to o_2 or o_3 with a probability of 0.25.

The mechanism that is proposed here works independent for different words so we will explain it for a single word w^* given the subsequent contexts with which it appears. If w^* occurs first at time t with context O_t, the agent associates a probability distribution $s_t : O \rightarrow [0, 1]$ with it, such that

$$s_t(o) = \begin{cases} \frac{1}{|O|} & \text{if } o \in O \\ 0 & \text{otherwise} \end{cases}$$

This implements the fact that all objects in O_t have an equal probability of being referred to by w^*, while all other objects have a zero probability.

Next, if w^* is observed again at time t' in a context $O_{t'}$, the probability distribution $s_{t'}$ is defined as follows. Let $\delta = 1 - \gamma$ with

$$\gamma = \sum_{o \in O_t} s_t(o).$$

Furthermore, let $\gamma' = (1 - \alpha)\gamma + \alpha$ and $\delta' = 1 - \gamma'$.
Then:

$$s_{t'}(o) = \begin{cases} \beta(\gamma) s_t(o) \frac{\gamma'}{\gamma} + (1 - \beta(\gamma)) \frac{\gamma'}{|O_t|} & \text{if } o \in O_t \\ s_t(o) \frac{\delta'}{\delta} & \text{if } o \notin O_t. \end{cases}$$

with $\beta(\gamma) = \sqrt{1 - (1 - \gamma)^2}$, a definition which is motivated further on.

In words, γ is the total probability of all objects consistent with the current observation (all objects in O_t). At time t', this probability is increased to $\gamma' \geq \gamma$ according to the parameter α. As such, α represents the strength with which the new information at time t is validated more important than the information gathered before time t. Furthermore, this new probability γ' should be distributed among the consistent meanings such that if it is in agreement with the current state (γ close to 1), then the relative probabilities among the consistent meanings should be more or less conserved (i.e. strong associations remain strong and weak ones remain weak). Therefore we require $\beta(1) = 1$. However, if the new information is not in agreement with the current state (γ small), then we want γ' to be more or less distributed evenly among all objects in O_t. Therefore we also require $\beta(0) = 0$. Moreover, we want that all scores of objects in O_t increase if $\gamma < 1$, because this guarantees convergence to a unique

the population then the language can de-stabilize. This might be different if a less adaptive learning algorithm would have been used.

interpretation if the contexts are random but always contain a certain object. It is easily verified that a necessary condition for this is that

$$\beta(\gamma) > \frac{\gamma}{\gamma'} = \frac{\gamma}{(1-\alpha)\gamma + \alpha}$$

for $\gamma < 1$. From this it follows that $\beta'(1) \leq \alpha$.[6] In order for the update mechanism to work for all values of α, we chose $\beta'(1) = 0$. The specific definition of β given meets all of these requirements.

In any case, the total probability δ of inconsistent associations is weakened to $\delta' \leq \delta$.

We will refer to this updating mechanism which transforms s_t in $s_{t'}$ as a function u such that

$$s_{t'} = u(s_t, O_t).$$

To illustrate this estimation function, assume that $\alpha = 0.3$ and that $O_t = \{o_1, o_2\}$ and $O_{t'} = \{o_1, o_3\}$. Then initially $s_t(o_1) = s_t(o_2) = 0.5$ and $s_t(o_3) = 0$. After observing $O_{t'}$ we have $s_{t'}(o_1) \simeq 0.61$, $s_{t'}(o_2) \simeq 0.35$ and $s_{t'}(o_3) \simeq 0.04$. Notice the asymmetry between the scores of o_2 and o_3. That o_3 receives a low score is understandable: there is no need to really consider it as the meaning of w^* (the word being learned) because no inconsistency has occurred: one of the other possibilities (o_1) is still in agreement with the previous state. At the same time, the value of the learning parameter α is such that learning is not very eager, resulting in a relatively high score for o_2 in the example. If the value of α would be increased then learning would be more eager and symmetry would be higher (see Figure 2.12.) On the other hand, the possibility of overshoot phenomena would increase as well.

Agent Architecture

For the current discussion, agents can be schematically depicted as in the naming game (see Figure 2.3.1.)[7] At time-step t, an agent can be described by a tuple $\langle W_t, \sigma_t, \phi_t \rangle$. W_t is the the agent's lexicon containing the set of words the agent has encountered until that moment. Initially, for each agent holds $W_0 = \emptyset$.

$\sigma_t : W_t \times O \rightarrow [0, 1]$ is a function which associates objects with words, such that, $\sigma_t(w, o)$ gives the agent's estimation of the probability that word w refers to object o.[8]

[6] β' is the derivative

[7] Note that in the other, more complicated version of the guessing game, in which only features of objects are described, the schema's should be extended with an ontology and the perceive, topic selection and conceptualize processes, (see figures 2.2 and 2.2.)

[8] It might seem that the agents would have to know all the possible objects in O in advance, but this is not the case: as can be verified in the following, σ_t will always be zero for categories not yet encountered.

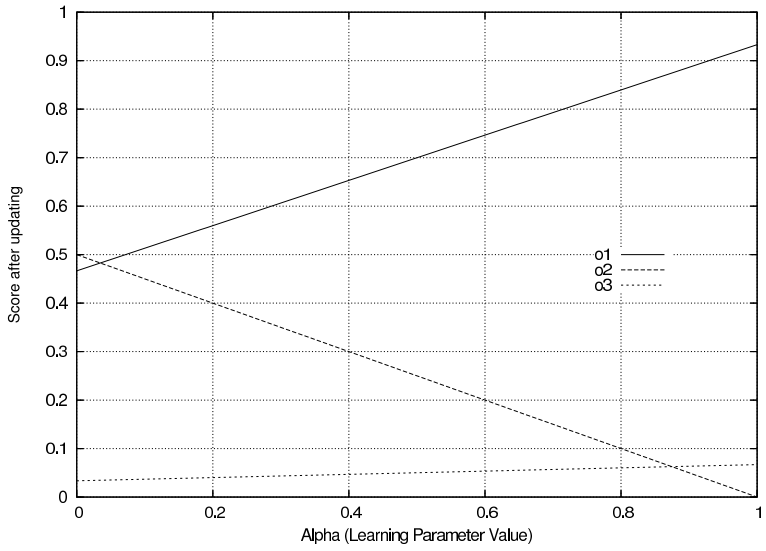


Figure 2.12: Resulting probabilities of a word meaning either o_1, o_2 or o_3 after observing contexts $\{o_1, o_2\}$ and $\{o_1, o_3\}$.

$\phi_t : W_t \to [0, 1]$ is a function which associates synonymy scores with words, which will be used to dampen synonymy as in the naming game.

The table below represents an agents' state for $O_t = \{o_1, o_2, o_3\}$ and $W_t = \{w_1, w_2, w_3\}$. The lower-right matrix contains the values $\sigma_t(w, o)$ and the values above the words are the word scores $\phi_t(w)$.

	1.0	0.8	0.9
	w_1	w_2	w_3
o_1	0.1	0.7	0.4
o_2	0.5	0.2	0.4
o_3	0.4	0.1	0.2

Before explaining the production and interpretation behavior of an agent and the way a state is updated, we first define the interpretation function of an agent $\langle W_t, \sigma_t, \phi_t \rangle$ as the function $g : W_t \to O$ with

$$g(w) = \text{argmax}_{o \in O} \left(\sigma(w, o) \right).$$

If multiple objects in O have a maximum value then one is chosen at random. Therefore $g(w)$ is possibly a stochastic value. For instance in the example agent above we have $g(w_1) = o_2$, $g(w_2) = o_1$ and w_3 interprets as o_1 or o_2, both with a probability of 1/2.

Production

Suppose that the speaker at time t is given by $\langle W_t, \sigma_t, \phi_t \rangle$, the context is O_t and the topic he will speak about is $o^* \in O_t$. The production behavior of a speaker does not depend on the context. He searches for words $W' \in W_t$ which according to him interpret as o^*: $W' = \{w \in W_t | g(w) = o^*\}$. As $g(w)$ can be stochastic, also can W'. If $W' = \emptyset$ (which is always the case if o^* is encountered for the first time) the speaker invents a new word $w^* (\notin W_t)$. If $W' \neq \emptyset$ then he chooses the word w^* from W' with the highest preference (synonymy) score: $w^* = \text{argmax}_{w \in W'} \phi_t(w)$. Again, if multiple words have a highest score, one is selected at random.

The speaker only updates his internal state if he invented a new word. Obviously, we then have: $W_{t+1} = W_t \cup \{w^*\}$. The word-meaning scores of known words remain the same, but for the new word we have:

$$\sigma_{t+1}(w^*, o) = \begin{cases} 1 & \text{if } o = o^*, \\ 0 & \text{otherwise.} \end{cases}$$

Finally the new word gets synonymy score 1: $\phi_{t+1}(w^*) = 1$ (scores for other words are left unchanged.)

To illustrate this consider the example agent introduced before. If this agent is a speaker and he has to verbalize o_3 he will invent a new word, say w_4. The new agent's state then becomes (changed values in bold):

	1.0	0.8	0.9	**1.0**
	w_1	w_2	w_3	$\mathbf{w_4}$
o_1	0.1	0.7	0.4	**0**
o_2	0.5	0.2	0.4	**0**
o_3	0.4	0.1	0.2	**1**

If he has to verbalize o_2, he will look for words which interpret as o_2, hence there is 1/2 chance that he will use w_1 and 1/2 chance that he will have to choose between w_1 and w_3 according to the score function ϕ. As $\phi(w_1) > \phi(w_3)$ he will choose w_1 in this case. The state of the agent does not change.

To summarize, the speaker has produced a word w^* for object o^* in context O_t, thereby possibly changing its internal state. As will be described next, the major state change occurs at the hearer side.

Parsing

Suppose that the hearer at time t is given by $\langle W_t, \sigma_t, \phi_t \rangle$, the context is O_t and the word received is w^*. If this word is unknown to him ($w^* \notin W_t$) then we

obviously have $W_{t+1} = W_t \cup \{w^*\}$ and the word-meaning association scores for w^* are initialized as follows:

$$\sigma_{t+1}(w^*, o) = \begin{cases} \frac{1}{|O_t|} & \text{if } o \in O_t \\ 0 & \text{otherwise} \end{cases}$$

If the agent does know the word ($w^* \in W_t$), the association scores involving w^* are altered according to the updating function u defined before:

$$\sigma_{t+1}(w^*, \cdot\,) = u(\sigma_t(w^*, \cdot\,), O_t).$$

Synonymy scores are updated according to the usual mechanisms of reenforcement plus lateral inhibition of competing synonyms. Using the auxiliary definition $\phi_t(w^*) = 1$ if $w^* \notin W_t$ we have first that the interpretation o' of w^* is determined as $o' = g(w^*)$ (with g using σ_{t+1}). Next, the set of synonyms S for w^* is determined as those words in $W_{t+1} \setminus \{w^*\}$ which also have interpretation o' (according to g). Finally, the synonymy score of w^* is increased: $\phi_{t+1}(w^*) = (1-\theta)\phi_t(w^*) + \theta$, and the scores of the synonyms are 'laterally inhibited': $\phi_{t+1}(w) = (1-\theta)\phi_t(w)$ for $w \in S$. In the rest of this chapter we assume that θ equals 0.3.

To illustrate the interpretation and updating, consider the example agent introduced before. If he is a hearer and would hear the word w_4 with context $O_{t'} = \{o_1, o_3\}$ then w_4 is added with new entries for σ. In addition, synonyms of w_4 are inhibited. With 1/2 chance $g(w_4) = o_3$ and there are no synonyms. With 1/2 chance $g(w_4) = o_1$ in which case w_2 is a synonym and with 1/2 chance also w_3. Suppose both w_2 and w_3 are synonyms then the agent's state becomes:

	1.0	**0.56**	**0.63**	**1.0**
	w_1	w_2	w_3	$\mathbf{w_4}$
o_1	0.1	0.7	0.4	**0.5**
o_2	0.5	0.2	0.4	**0**
o_3	0.4	0.1	0.2	**0.5**

If he would hear the word w_3 with context $O_{t'} = \{o_2, o_3\}$, the associated scores will be updated as follows. The total probability of the consistent meanings in $O_{t'}$ is $\gamma = 0.4 + 0.2 = 0.6$. With $\alpha = 0.3$ this will be transformed to $\gamma' = 0.72$, giving rise to $\beta \simeq 0.92$.

With regard to the synonymy scores we have that $\phi(w_3)$ will increase according to the lateral inhibition parameter θ, and since now w_3 and w_1 are synonyms $\phi(w_1)$ will be inhibited. The new state of the agent is given by:

	0.7	0.8	**0.93**
	w_1	w_2	w_3
o_1	0.1	0.7	**0.28**
o_2	0.5	0.2	**0.47**
o_3	0.4	0.1	**0.25**

Simulation results

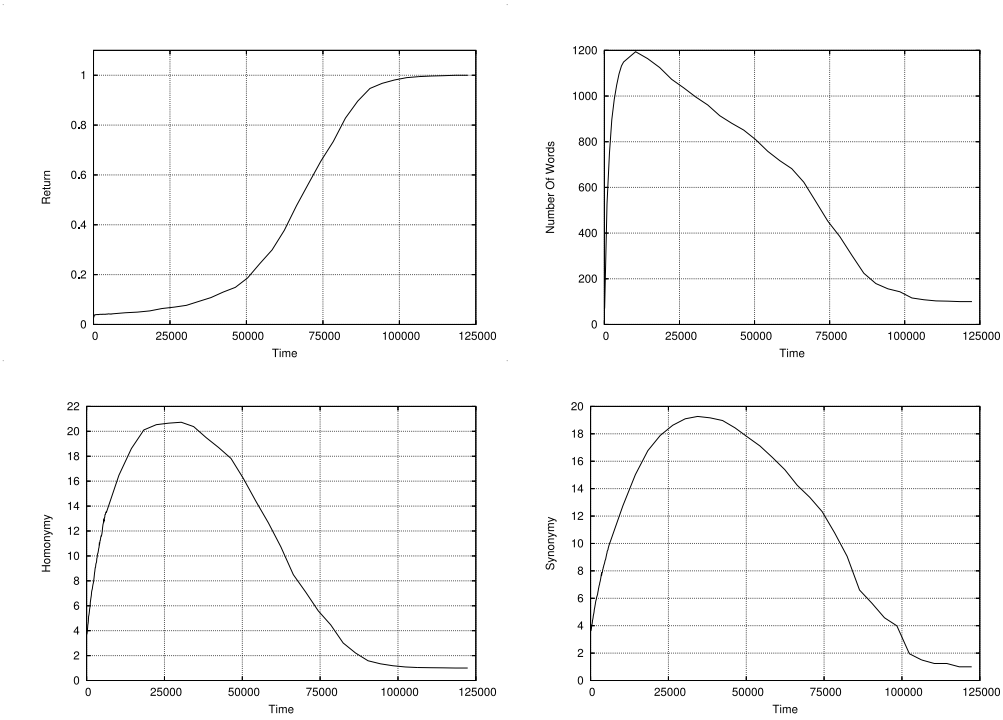


Figure 2.13: Evolution over time of the return, the number words used by the agents in the population, the homonymy and the synonymy as defined in the text. The graphs were obtained for $N = 20$ agents, $|O| = 100$ objects and a context size of $|O_t| = 5, t \geq 0$. The forgetting parameter α was set to 0.2 and the synonymy inhibition parameter θ was 0.3.

We will now present some results of a simulation involving $N = 20$ agents evolving a lexicon to communicate about 100 objects ($|O| = 100$). Each interaction the speaker and the hearer are presented with a context containing 5 objects ($\forall t \geq 0 : |O_t| = 5$.) The evolution over time of the return, the number of words used by the agents in the population, the homonymy and the synonymy are presented in figure 2.13.

As can be seen, the agents eventually reach a coherent and efficient language without synonyms or homonyms. The return, which is related to the communicative success (and necessarily equivalent with it when it is 1), starts at a small (chance level) value. As new words are introduced and as their meanings starts to settle, the return gradually increases until finally it becomes maximum at around $t = 105000$.

The maximum number of words present in the population is reached approximately at $t_{\max} \simeq 10000$ and is equal to 1194, which is of the order of $N|O|/2 = 1000$ as would be expected. The cross-situational learning mechanism as proposed allows the agents to eliminate homonyms, which partly explains why the number of words decreases steadily after $t_{\max}$.

Figure 2.14 shows some subsequent snapshots of how the semiotic landscape looks like over time during a typical experiment.

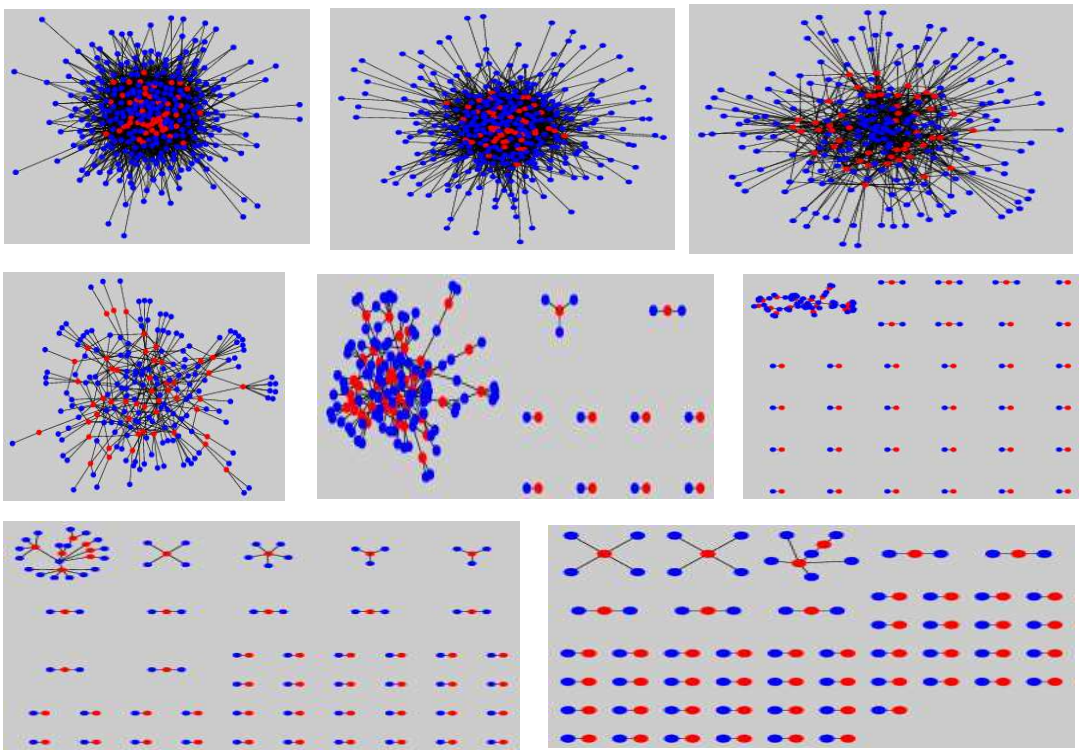


Figure 2.14: Snapshots of the semiotic landscape in a cross-situational learning guessing game experiment involving $N = 20$ Agents, a world size $|O| = 50$ and context size $|O_t| = 5$. Time proceeds from left to right and from top to bottom.

In contrast to earlier findings by Vogt and Coumans (2003), our agents do reach a complete coherent language, where coherence is defined as the chance of two random speakers producing the same word for the same meaning. The main difference between Vogt and Coumans' agents and the ones defined in this section is the use of the synonymy-damping mechanism which explains further why the number of words used by the agents eventually drops to the number of objects $|O|$.

In a nutshell, this section introduced a cross-situational learning mechanism capable of estimating a word/meaning mapping that changes over time without the need for explicit examples. What also should be remembered is that the conditions that determine when a speaker needs to invent a new element of language (e.g. a new word but it might also be a new grammatical construction) need to be extended beyond the obvious case of uncoveredness: also when he does not know a word which he himself would *interpret* as the target meaning should he invent a new one.[9]

2.3.3 Games Requiring Grammar

Until now, the languages considered didn't involve any grammar or syntax. One of the reasons is that so far all language games only involved a single object as the topic. Therefore, it sufficed to consider only simple meaning-word associations, where the meaning of a word was a single category directly referring to an object.

Most conceptual categories are multi-referential, meaning that they may apply to several objects or events at a time. We can explicitate this by letting the categories take arguments, like predicates instead of constants. Hence, the meaning of a word w_i becomes $c_i(?x)$, where any symbol starting with a question mark represents a variable. When the variable is replaced by a constant referring to an actual object as in $c_i(o_1)$, the result is a predicate stating that category c_i is true for the object o_1.

Crucially, the introduction of variables allows to represent meanings involving several objects. For example, the expression:

$$\text{ball}(?x) \wedge \text{box}(?y)$$

describes a scene containing both a ball and a box. If the ball is red and the

[9] As it turns out, the proposed learning mechanism can be simplified somewhat. Moreover, it is insufficient when the ratio of the number of objects in the world compared to the number of objects per scene is too high (private communication with Bart De Vylder.)

box is blue it can be extended to:

$$\text{red}(?x) \wedge \text{ball}(?x) \wedge \text{blue}(?y) \wedge \text{box}(?y).$$

Notice that the equalities of variables in this expression are important: the fact that the variable to the 'red' and 'ball' predicates are the same actually means that it is the ball that is red. One of the functions of grammar is precisely to express these kind of linking dependencies between different parts of an utterance's meaning. English mainly uses word order to express such dependencies.

So far, an entry in an agent's language inventory consisted of a mapping between a meaning and a form, telling the agent how to express the meaning or how to parse the form. It sufficed to consider only very simple mappings, going from a single part of meaning (an object identifier) to a single part of form (a word.) However, the introduction of equalities in the meaning forces us to consider mappings between *several* parts of meaning and *several* parts of form: for example between two predicates linked by a variable equality and the order between the words expressing the predicates.

Moreover, we need a way to refer to the different parts of meaning and form without having to specify in detail what they should be. It doesn't make sense to have a specific construction for a red ball, and another one for blue balls or red boxes. Instead, what is needed are form-meaning mappings at the level of variable equalities and semantic and syntactic categories like objects and features and adjectives and nouns.

In short, we should be able to represent phrases, sub-phrases, syntactic features like number etc. Many contemporary linguistic formalisms use feature structures or attribute-value matrices for this. In the next sections we will also introduce a variant of these, called unit-structures. Every part of an utterance will be represented by a unit. Every unit holds information about a word or phrase.

A unit structure holds several units together, which allows cross-referencing between units. The entries in an agent's language inventory will be represented as mappings between abstract unit-structures, typically one specifying a set of semantic features (the meaning pole) and one specifying a set of syntactic features (the form pole). Whether such a mapping between a semantic and a syntactic pole applies to a particular unit structure will require the unification between one of its poles and the actual structure representing the utterance being processed. The application of the mapping will require a merge operation between the opposite pole of the mapping with the structure. This will be fleshed out in the following two chapters.

Chapter 3

Fluid Construction Grammar

3.1 Introduction

In the previous chapter, a number of developmental stages were introduced that are relevant for research in the origins and evolution of (artificial) languages. Most of the solid results so far have been obtained for the earlier stages and concerned mainly the emergence of lexicons (see overviews and representative samples of current work in Batali (2002), Cangelosi and Parisi (2001), Minett (2005), Steels (2005)). Although there have been some experiments on the role of syntax and grammar (see e.g. Hashimoto and Ikegami (1996), Steels (1998), Batali (1999), Smith et al. (2003b)), there are as yet only very few demonstrations where non-trivial grammars arise in grounded situated interactions between robotic agents.

Part of the reason is of course that the problem of grammar emergence is much more encompassing than that of lexicons and many fundamental questions remain unanswered. In addition, the world models of agents and the nature of their interactions needs to be much more complex than in lexical experiments. But another reason, we believe, has to do with the nature of the computational apparatus that is required to do serious systematic experiments. Whereas lexicon emergence can be studied relatively easy, grammar requires much more powerful techniques from symbolic processing. To this end, Luc Steels triggered the development of Fluid Construction Grammar (FCG)

One of the reasons why it was decided to start the development of Fluid Construction Grammar was that not many existing and operational formalisms were available, and especially not any that could be used both for parsing and production. This is however one of the key features on the list of requirements that any formalism adequate to support our research should exhibit.

In addition, most existing formalisms were (and still are) rather strongly committed to one or the other linguistic theory.

For example, Head Driven Phrase Structure Grammar (HPSG) and any flavor of dependency grammar are centered around *head-dependent* relations like *head-complement*, *head-modifier* (or *head-adjunct*) and *head-specifier*. The head of a phrase is its core element determining a number of its key properties. For example, in the head-modifier phrase "beautiful Mary", the head role is fulfilled by "Mary" because both phrases have the same semantic and syntactic categories. The "beautiful" part is called the *modifier*. The distinction between the different kind of relations like head-complement and head-modifier is typically defined in terms of valency, which is usually related to the semantic predicate-argument structure associated with certain classes of lexemes.

While notions like head and modifier are very useful, not every linguistic phenomenon lends itself equally well to be analyzed in terms of them. For example, for grammatical function words, such as articles, complementizers and auxiliary verbs, but also for structures involving prepositional phrases or coordination, there is no general consensus as to whether they can or should be analyzed in terms of head-dependent relations.

As a second example, most linguistic formalisms assume a fixed set of semantic and syntactic categories. In contrast, FCG wants to leave open the possibility to investigate the origins and evolution of such categories in a relatively easy and unconstrained way[1], which is another reason why the development of FCG was started.[2]

Yet another thing that triggered the development of FCG is the fact that most linguistic formalisms are primarily concerned with the *characterization* of *human* linguistic competence. Although HPSG tries to relate as much as possible to aspects of linguistic performance and psycholinguistic findings, the primary goal is to build a theory of the knowledge embodied in the human brain, and not to devise working systems and computational implementations that *use* this knowledge (Pollard, 1997).

For example, and like most other formalisms, much of HPSG is concerned with well-formedness and basically is about finding a minimal but necessary

[1] First results in this direction were already obtained by Steels (2004) and Van Trijp (2008).

[2] It must be noted however that, depending on the particular aspect of language being investigated, every FCG simulation does make a number of assumptions commonly made in other formalisms and theories of language. For example, for the simulations reported on in this thesis (see chapter 5 and following) we did assume the existence of a pre-defined and universal set of semantic and syntactic categories. The point is, at least from the point of view that FCG was developed as a supporting framework for doing research on the origins and evolution of a wide range of linguistic phenomena, that I had a choice to do so, but did not have to.

set of grammatical rules and principles such that a certain body of empirical linguistic data is *licensed* by it, or, in other words, such that the data *satisfies* the grammar. While certainly being very valuable and informative about the structure of human languages, this leaves out somewhat the questions of how and why such data could be *produced, learned* and *evolve*, precisely the questions we are interested in. Hence, while many formalisms are primarily geared towards checking for grammaticality, FCG instead is more concerned with things like flexibility, learning, invention, usage and other creative aspects of language.

Finally, and in contrast with much of mainstream linguistics, the nature of our research and our research methodology force us to treat semantics and especially how it is linked with syntax as one of the starting points and key elements of language. It is therefore no coincidence that FCG shares many commonalities with ideas advocated in usage-based cognitive approaches to language in general and construction grammar in particular. Unfortunately, most cognitive and constructional approaches to language do not come with any computational implementation that could be used for our research. An exception to this is *Embodied Construction Grammar* (Bergen and Chang, 2003), however only algorithms for parsing are available and, until now, any attempt to do production has failed or was abandoned.

For all these reasons, the Artificial Intelligence Lab of the University of Brussels together with people at Sony CSL in Paris have for many years now been working on a formalism that can handle both production and parsing and that would be adequate for handling phenomena typically found in natural language grammars, but that would at the same time support highly flexible parsing and production (even of ungrammatical sentences or partially unconventionalized meanings) and invention and learning operators that could lead to the emergence, propagation, and further evolution of grammar in a multi-agent setting. At this point the FCG system is ready for use by others.[3]

FCG uses many existing and widely accepted notions in theoretical and computational linguistics, specifically feature structures for the representation of syntactic and semantic information during parsing and production, and abstract templates or rules for the representation of lexical and grammatical usage patterns, as in Sag et al. (2003) or Bergen and Chang (2003).

FCG is unification based and relies on general operations like *unify* and *merge*. Other contemporary linguistic theories also propose general operators for building up syntactic and semantic structures, such as Merge in Chom-

[3]An implementation on a LISP substrate has been released for free download through `http://arti.vub.ac.be/FCG/` and `http://www.emergent-languages.org`. These sites also contain very specific examples on how to do experiments in language evolution with FCG.

sky's Minimalist Grammar (Chomsky, 1995) or Unify in Jackendoff's framework (Jackendoff, 2002). Other unification based grammars (Pollard and Sag, 1994; Kay, 1984; Bergen and Chang, 2003) are similarly based on a unification operator, and, more generally, many generic inference systems, particularly within the logic programming framework, use some form of unification (Sterling and Shapiro, 1986). Unfortunately there are quite substantial differences between the use of the terms unify and merge in all these different frameworks. One of the goals of this chapter, and even more so of Appendix A, is to clarify in detail the unify and merge operators that form the core of FCG.

The remainder of this chapter has two parts. The next section defines first the requirements for grammar formalisms needed to do experiments in the emergence of non-trivial grammars and the basic ideas behind the Fluid Construction Grammar approach. Then there is a section introducing the basic concepts of FCG which should be sufficient to understand the rest of the thesis. Those who wish to dive into the details or prefer formal definitions are referred to Appendix A.

Keep in mind however that only a fraction of the machinery offered by the current FCG implementation is discussed. For example, FCG includes functions and methods for defining agents, their learning mechanisms and many other things relevant for doing language game experiments. Moreover, the simulation experiments that will be presented later on will make a number of assumptions that are not necessarily part of FCG, like the existence of a universal set of semantic and syntactic categories. This will be discussed in Chapters 5 and following.

3.2 Requirements

3.2.1 Linguistic Assumptions

The linguistic perspective of FCG is in the general line of cognitive grammar (Langacker, 2000) and more specifically construction grammar (Goldberg, 1995). This means the following:

1. FCG *is usage-based*: The inventories available to speakers and hearers consist of templates which can be highly specialized, perhaps only pertaining to a single case, or much more abstract, covering a wide range of usage events. There is no sharp distinction therefore between idiomatic and general rules. New sentences are constructed or parsed by assembling the templates using the unify and merge operators (defined later in this chapter.)

2. *The grammar and lexicon consist of symbolic units*: A symbolic unit associates aspects of meaning with aspects of form. The templates of FCG are all symbolic units in this sense. They feature a semantic pole and a syntactic pole. Templates are always bi-directional, and so are usable both for production *and* for parsing. This makes FCG unique not only with respect to derivational formalisms (like generative grammar or HPSG) but also with respect to other construction grammar formalisms which tend to be uni-directional (such as Embodied Construction Grammar).

3. *There is a continuum between grammar and the lexicon.* Not only can templates be at different levels of abstraction, but there is also no formal distinction in the structure or the processing of lexical and grammatical entries. In the case of lexical entries, the syntactic pole concerns mostly a lexical stem and the semantic pole tends to be equal to some concrete predicate-argument structure. In the case of grammatical constructions, the syntactic pole contains various syntactic categories constraining the sentence, and the semantic pole is based on semantic categories, but otherwise there is no formal difference between the two types of templates.

4. *Schematization occurs through variables and categorization.* A template has the same form as an association between a semantic structure and a syntactic structure, in other words both poles of a template are feature structures. However, templates are more abstract (or schematic) in three ways: Some parts of the semantic or syntactic structure are left out, variables are used instead of units and values, and syntactic or semantic categories are introduced to constrain the possible values of the semantic and syntactic pole. These categories are often established by syntactic or semantic categorization rules. In some experiments the categories have been defined using a memory-based approach with typical examples and prototypes.

5. . *Syntagmatic and Paradigmatic Compositionality*: To produce or parse a sentence, templates can be combined (several templates all matching with different parts of the meaning in production or with parts of the sentence in parsing are simply applied together) or integrated (using hierarchical templates that combine partial structures into larger wholes, possibly after a modification of the syntactic or semantic aspects of the component units). Apart from this syntagmatic composition, there is also the possibility that several templates are overlayed and each contribute additional constraints to the final sentence. This is paradigmatic compositionality. Both forms of compositionality are completely supported with the unify and merge operators defined later. Of particular importance is that the

FCG unify and merge operators handle linking (resolving variable equalities introduced by separate lexical items) by unifying variables in merge. This topic is discussed more extensively in Steels et al. (2005).

3.2.2 Additional Requirements

In addition to these characteristics, which we take to be general features of cognitive grammars and construction grammars, there are some additional requirements for a formalism if it is to be adequate for modeling emergent natural language-like grammars.

1. *All inventory entries have scores*: We assume that speakers and hearers create new templates which are often competing with each other. Not all templates are widely accepted (entrenched) in the population and there is a negotiation process. To enable this capability we associate with every item in the lexico-grammar a number of scores that reflect the degree of entrenchment of that item. It is based on feedback from success or failure in the language games in which the item has been used. We know from our other experiments and from the previous chapters about lexicon emergence that an appropriate lateral inhibition dynamics is the most adequate way for driving a population to a sufficiently co-ordinated repertoire to have success in communication and the same dynamics is part of FCG.

2. *The set of syntactic and semantic categories is open.* Often linguistic formalisms posit specific sets of semantic categories (for example semantic roles like agent, patient, etc.) or syntactic categories (such as parts of speech, syntactic features, and others). Because we are interested in how all these categories arise (although not in this thesis), we made the formalism completely open in this respect. Constraints on feature values are expressed as propositions or predicate-argument clauses so that the set of categories can be expanded at any time. In our experiments there should typically be hundreds or even thousands of new categories built. This openness of categories is in line with the Radical Construction Grammar approach which argues that linguistic categories are not universal and subject to evolution (Croft, 1991).

3. *The multi-agent perspective*: In traditional Chomskyan linguistics only the grammar or lexicon of an 'idealized speaker or hearer' is considered. In contrast, when we want to study how grammar arises in a population we need to take a multi-agent perspective, where every agent possibly has a different inventory. We need to understand in particular how agents co-ordinate their inventories to be successful in communication. This raises

a large number of computational issues (for example linguistic knowledge is always local to an agent) as well as issues in how to track and measure 'the grammar' in the population.

In the next section we will dive into the details of FCG in a way that should allow the reader to get trough the following chapters. As already said, appendix A goes into deeper detail.

3.3 Basic Fcg Concepts

3.3.1 Unit Structures

As mentioned at the end of the previous chapter, unit structures hold the information about the utterance being processed. They are represented as a list of units. Consider the sentence "Mary kisses John". The task of a hearer agent is to reconstruct the following meaning of the sentence:

```
(Mary ?x) (John ?y) (kiss ?e) (kisser ?e ?x) (kissee ?e ?y)
```

As a first step we create a unit for each word in the sentence, and one additional super-unit called `Top-unit` to hold the other three units together. Units are represented as lisp like lists (see e.g. Steele (1990a).) This means that prefix notation is used and the entire list is delineated by plain brackets as in:

```
(John-unit (form ((String "John")))).
```

The above expression represents a unit. It is a list with first element the symbol `John`, which is the unit's name, and second element the list `(form ((String "John")))`, which is the only feature of this unit. The feature's name again is the symbol `form` and its value is the list `((String "John"))` etc.

In general, units will be represented as a list, with the first element in the list the unit's name and all remaining elements its features. Features will also be represented as lists, again with a name as first element and a value as second. The order of all but the first element (i.e. of all features but not of the name) is of no importance. The name must however always remain the first element.

Unit structures will be lists of units. Hence, a unit structure containing a unit for each word in the sentence "Mary kisses John" and one additional super-unit called `Top-unit` to hold the other three units together looks like:

```
((Top-unit (syn-subunits (Mary-unit John-unit Kiss-unit))
           (form ((meets Mary-unit Kiss-unit)
                  (meets Kiss-unit John-unit))))
```

```
(Mary-unit (form ((string "Mary"))))
(John-unit (form ((string "John"))))
(Kiss-unit (form ((string "kisses"))))                    (1)
```

This structure contains four units named `Top-unit`, `Mary-unit`, `John-unit` and `Kiss-unit`. The `Top-unit` contains two features: the `syn-subunits` feature and the `form` feature.

As can be seen, in FCG, form constraints are also specified using predicate logic: the `meets` predicates in the `Top-unit`'s `form` predicate for example specify the order in which the other units should be rendered into a surface form (or parsed starting from a surface form.) On the other hand, the order in which the different units appear in the unit-structure itself is unimportant, so the above structure is equivalent to the one below:

```
((Mary-unit (form ((string "Mary"))))
 (Top-unit (syn-subunits (Mary-unit John-unit Kiss-unit))
           (form ((meets Mary-unit Kiss-unit)
                  (meets Kiss-unit John-unit))))
 (Kiss-unit (form ((string "kisses"))))
 (John-unit (form ((string "John"))))).                   (1)
```

Many linguistic formalisms (e.g. HPSG and ECG) represent feature structures with a boxed notation or as *attribute value matrices* (AVM's) instead of with the bracketed lisp-like notation adopted here. In such a notation the above unit structure for the sentence "Mary kisses John" could be represented as:

$$\left[\begin{array}{ll} \text{Top-unit} & \\ \text{FORM} & \langle \text{meets}(\boxed{1},\boxed{2}), \text{meets}(\boxed{2},\boxed{3}) \rangle \\ \text{SYN-SUBUNITS} & \left\langle \begin{array}{l} \boxed{1}\left[\begin{array}{ll} \text{Mary-unit} & \\ \text{FORM} & \langle \text{STRING} \quad \text{“Mary”} \rangle \end{array}\right], \\ \boxed{2}\left[\begin{array}{ll} \text{Kiss-unit} & \\ \text{FORM} & \langle \text{STRING} \quad \text{“kisses”} \rangle \end{array}\right], \\ \boxed{3}\left[\begin{array}{ll} \text{John-unit} & \\ \text{FORM} & \langle \text{STRING} \quad \text{“John”} \rangle \end{array}\right] \end{array} \right\rangle \end{array}\right]$$

In this notation lists are typically delineated with hooked brackets (like <this>.) Both representations are more or less equivalent.

Whatever the notation used, a unit structure can easily be extended. For example, the fact that the string "kisses" is a third person singular form of the verb "to kiss" can be added to the structure in (1) to yield:

```
((Top-unit (syn-subunits (Mary-unit John-unit Kiss-unit))
           (form ((meets Mary-unit Kiss-unit)
                  (meets Kiss-unit John-unit))))
 (Mary-unit (form ((string "Mary"))))
 (John-unit (form ((string "John"))))
 (Kiss-unit (form ((string "kisses")
                   (stem "kiss")))
            (syn-cat ((lex-cat verb)
                      (number sing)
                      (person 3rd)))))                    (2)
```

In Fluid Construction Grammar, semantic and syntactic information is kept in separate unit structures. Syntactic units normally have the features `syn-subunits`, `form` and `syn-cat`. Semantic units typically contain the features `referent`, `sem-subunits`, `meaning` and `sem-cat`. The `sem-cat` feature holds information about the semantic category of the unit, like whether it is about an object or person or maybe a CAUSE-MOVE event.

The fact that semantic and syntactic information is kept in different structures reflects that constructions in language are meaning-form mappings. So a lexical construction for "Mary" is a mapping between a syntactic pattern selecting for the string "Mary" in the form feature of a syntactic unit, and a semantic pattern introducing the predicate 'Mary(?x)' in the meaning feature of the corresponding semantic unit.

3.3.2 Templates

The question arises how the constructional 'patterns' should be represented. It would be convenient to formulate them as partially specified unit-structures like:

```
((?unit (form ((string "Mary"))))),                       (3)
```

for the syntactic part of the lexical "Mary" construction and:

```
((?unit (meaning (((Mary ?x))))))),                       (4)
```

for the semantic pole. These again look like unit structures containing only one unit but with a variable name (remember that any symbol starting with a question mark is a variable), reflecting the fact that the construction shouldn't care about the name of the unit it selects for. Unit structures that contain variables like this are called unit structure *templates* or short templates. They actually specify a *set of* unit structures.

The operation that decides whether a template matches a specific unit structure is called *unification* of the template and the unit structure. This operation allows that the unit structure contains more units than specified by the template. It also allows a particular unit in the template to contain more features than specified. The operation is also insensitive to the order of the units in the structure or of the features in a unit.

The result of unifying a template with a particular unit structure is a set of sets of bindings of variables to actual values. For example, the unification of the template (3) with the unit structure (1) is a set containing one set of bindings:

```
{{[?unit/Mary-unit]}}.                                    (5)
```

This will sometimes also be represented as:

```
(((?unit . Mary-unit))).
```

A set of bindings specifies how to make a structure from a template: by substituting the variables in the template by the values they are bound too. Unification is more properly defined in Appendix A.

3.3.3 The includes and other operators

FCG makes intensive use of so called special operators. These can be regarded as special directives informing the unification and merging engine that something special needs to be performed. An example operator that is used frequently is the *includes* operator 'verb+==+'. A list of which the first element is this operator unifies with all lists that at least contain the other elements in the includes list. For example, the includes list

```
(== a c b)
```

unifies with all of the lists below:

```
(a c b),
(a b c),
(a c b d e),
(e c d c b a)
```

and so on. As can be seen, the order in which the elements appear is of no importance, only the fact that they are present.

The use of operators like the includes operator is the reason why FCG-unification may return multiple results. For example, unifying the list

```
(== ?x)
```

with the list

```
(== a b c)
```

results in the following set of three solutions:

```
{{[?x/a]},
 {[?x/b]},
 {[?x/c]}}
```

Many other operators are defined in FCG. Most of them are defined in Appendix A.

3.3.4 Constructions and Merging

Now that we know how to specify patterns as structure templates, we can proceed with specifying constructions as mappings between a semantic and a syntactic template as in:

```
((?unit (meaning ((Mary ?x)))))
<-->
((?unit (form ((string "Mary"))))) (6)
```

The semantic pole is written above the double arrow and the syntactic pole below it. The unification of the syntactic pole with the unit structure in (1) results in the set of bindings (5), and substitution of these bindings in the semantic pole leads to:

```
((Mary-unit (meaning ((Mary ?x)))))
```

which is a specification of the changes to be made (things to add) to the semantic structure during parsing.

The operation that takes a unit structure and forms a new extended structure as specified by a template is called *merging*.[4] It is formally defined in Appendix A.

While parsing an utterance, the right (syntactic) pole of a construction is unified with the syntactic structure to see whether it applies. If it does, the left (semantic pole) is merged with the initially empty semantic structure to yield a new semantic structure. While producing an utterance the semantic pole is unified with the semantic structure and, if successful, the syntactic pole is then merged with the initially empty syntactic structure to yield a new syntactic structure.

[4]This has nothing to do with Chomsky's notion of merge.

To sum up, so far we have represented the information about an utterance with unit structures. Because it is always possible to add units to a structure, or features to a unit or values to a feature value if it is a list, this representation seems to be powerful enough to carry us a long way. We have also set first steps towards representing rules of language as bi-directional mappings between structure templates. These mappings can be used both for parsing forms and for generating them, similar to the simpler associative mappings between categories and words in the naming and guessing game.

3.3.5 Variable Equalities and Grammar

While parsing "John kisses Mary" and after applying lexical constructions as was briefly discussed in the previous section, the semantic and syntactic structures could look like:

```
((Top-unit  (sem-subunits (Mary-unit John-unit Kiss-unit)))
 (Mary-unit (meaning ((Mary ?m)))
            (sem-cat ((person ?m))))
 (John-unit (meaning ((John ?j)))
            (sem-cat ((person ?j))))
 (Kiss-unit (meaning ((kiss ?e)
                      (kisser ?e ?x)
                      (kissee ?e ?y)))
            (sem-cat ((event ?e)
                      (agent ?e ?x)
                      (patient ?e ?y)))))                (7)
```

and

```
((Top-unit  (syn-subunits (Mary-unit John-unit Kiss-unit))
            (form ((meets Mary-unit Kiss-unit)
                   (meets Kiss-unit John-unit))))
 (Mary-unit (form ((string "Mary")))
            (syn-cat ((lex-cat Noun)
                      (number singular)
                      (person 3rd))))
 (John-unit (form ((string "John")))
            (syn-cat ((lex-cat Noun)
                      (number singular)
                      (person 3rd))))
 (Kiss-unit (form ((string "kisses")))
            (syn-cat ((lex-cat Verb)
```

```
                        (number singular)
                        (person 3rd)))))                    (8)
```

The collection of predicates contained in the semantic structure is:

```
(Mary ?m)
(John ?j)
(kiss ?e) (kisser ?e ?x) (kissee ?e ?y).
```

This almost looks like the desired result (see the beginning of this chapter), except that the variable `?m` still needs to be identified with the variable `?x` and `?j` with `?y`, reflecting that it is Mary who is doing the kissing and John who is being kissed. As already mentioned, this is (one of) the function(s) of grammar.

The `meets` constraints in the `Top-unit`'s `form` feature together with the `syn-cat` features of the other units tell us that we are dealing with a plain English SVO construction (a subject noun phrase followed by a verb followed by an object noun phrase and the verb agrees with the subject in person and number.) Therefore it can indeed be be concluded that it is Mary who is kissing John and not the other way around. We can directly translate this into a mapping between structure templates:

```
((?Top (sem-subunits (== ?S-unit ?V-unit ?O-unit)))
 (?S-unit (sem-cat ((person ?a))))
 (?V-unit (sem-cat (== (event ?e)
                       (agent ?e ?a)
                       (patient ?e ?p))))
 (?O-unit (sem-cat ((person ?p)))))
<-->
((?Top (syn-subunits (== ?S-unit ?V-unit ?O-unit))
       (form (== (meets ?S-unit ?V-unit)
                 (meets ?V-unit ?O-unit))))
 (?S-unit (syn-cat (== (lex-cat Noun)
                       (number ?number)
                       (person ?person))))
 (?V-unit (syn-cat (== (lex-cat Verb)
                       (number ?number)
                       (person ?person))))
 (?O-unit (syn-cat (== (lex-cat Noun))))).                  (9)
```

Let us examine this construction in more detail. First, on the meaning side, the rule's left pole essentially specifies an event involving two participants: one referred to by the variable `?a` in the `?S-unit` and one by the variable `?p` in the $?O-unit$. The exact name of the variables and units is of no importance, what

is important is that both in the `?S-unit` and in the `?V-unit` the *same variable* is used to refer to the agent. The same holds for the patient in the `?V-unit` and the `?O-unit`. Hence, given appropriate bindings for the unit variables, the effect of merging this pole with a unit structure will essentially be that these variables are made equal if they are not.

On the other hand, while producing a sentence the rule will only apply (unify with the semantic structure) if the equalities are already present in the semantic structure since in production mode the predication will be done over actual values instead of variables. This is exactly what we need.

Second, on the form side, the right pole of the rule essentially constrains the form of the utterance by specifying the order of the different constituents, by requiring that they are of the appropriate lexical category and by enforcing agreement between the subject and verb phrases.

Third, one can see how the includes special operator `==` is used extensively to make the rule more general: there might be more subunits allowed or parts of meaning or form or syntactic categories, and the order in which the required ones appear is of no importance, all because of the use of the includes operator.

Fourth, the rule is made more general with respect to the exact nature of the event or its participants by making use of the `sem-cat` feature. The rule does not require the event to be a kiss event, or the agent to be Mary. It only requires the presence of units of the right semantic category.

Finally, note that the construction does not change anything to the hierarchical structure of the semantic and syntactic structures. This is required however for dealing with more complex sentences containing sub-phrases like "I see that Mary kisses John" or "John kisses beautiful Mary". An analysis of these sentences requires the possibility to refer to the phrases "Mary kisses John" and "beautiful Mary" as a whole. How this is handled in Fluid Construction Grammar is explained in the next chapter.

Chapter 4

Hierarchy and the J-operator

The previous chapter introduced Fluid Construction Grammar (FCG) as a general formalism for representing and processing linguistic knowledge which was developed to push research about the emergence and evolution of language from the domain of lexical languages to the domain of grammar. With the machinery presented so far, many aspects of grammar can already be investigated, like for example the emergence of semantic and syntactic categories or of simple argument structure constructions. An example of this is given in (De Beule and Bergen, 2006).

However, many other aspects, and in particular those that require manipulation of the hierarchical structure of unit structures, require more powerful operations than the basic unification and merging of unit structures. To remedy this flaw, the author extended FCG with the so called *J-operator*.[1] In this chapter, the problem that needs to be tackled is set and it is shown how it is solved by the J-operator. As it turns out, the J-operator also allows to remedy some other imperfections in FCG. This will be illustrated in section 4.3.

4.1 Hierarchy in Syntax

It is fairly easy to see what hierarchy means on the syntactic side and there is a long tradition of formalisms to handle it. Suppose we want to parse the sentence "John kisses beautiful Mary". The phrase "beautiful Mary" combines two lexical units to form a new whole. In terms of syntactic categories, we say that "beautiful" is an adjective, that "Mary" is a kind of noun phrase, and that the combination is again a noun phrase which can function as a unit in a larger structure as in "John kisses beautiful Mary". Traditionally, this kind of phrase

[1] 'J' stand for Joachim, a choice that was not particularly encouraged by the author but is nevertheless commonly used within the FCG community.

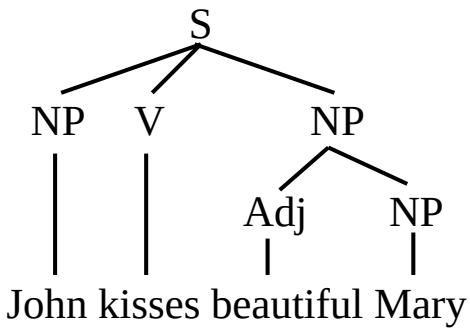


Figure 4.1: Hierarchical syntactic structure for a simple sentence.

structure analysis is represented in graphs as in figure 4.1. In the terminology of Fluid Construction Grammar it is said that "beautiful Mary" is a noun phrase construction of the type `Adj-NP`. It consists of a noun-phrase (NP) unit which has two subunits, one for the adjective and one for the NP. The NP is in this case realized by a specific type of NP, namely the Proper Noun "Mary". So constructions come in different types and subtypes.

Various syntactic restrictions are associated with the realization of a construction of a specific type or subtype which constrain its applicability. For example, in the case of the `Adj-NP` construction, there is a particular word order imposed, so that the adjective comes before the NP. Some languages also impose agreement in number between the adjective and the NP. This is the case in French where there are different `Adj-NP` constructions with different word order patterns depending on the type of adjective and/or its semantic role ("une fille charmante" vs. "un bon copain").

When a parent-unit is constructed that has various subunits, there are usually properties of the subunits (typically called the *head features*, see section 3.1) that are carried over to the parent-unit. For example, if a definite article occurs in an NP construction of type `Article-Adjective-Noun`, then the NP as a whole is also definite. Or if the head noun is singular than the noun phrase as a whole is singular as well. It follows that the grammar must be able to express (1) what kind of units can be combined into a larger unit, and (2) what the properties are of this larger unit.

So assume that a lexical analysis of the sentence "John kisses beautiful Mary" results in the following syntactic structure:[2]

[2]Note that for simplicity we have changed the lexical category of "John" and "Mary" from `Proper-Noun` to `NP`. In principle it would be possible to devise rules that specify `Proper-Noun` as a special case of `NP`, and extend the `syn-cat` feature of the `John-unit` and `Mary-unit` accordingly. However, since this would not contribute to the current discussion about hierarchy

```
((Top-unit (syn-subunits
              (John-unit kiss-unit beautiful-unit Mary-unit))
           (form ((meets John-unit kiss-unit)
                  (meets kiss-unit beautiful-unit)
                  (meets beautiful-unit Mary-unit))))
 (John-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (kiss-unit (syn-cat ((lex-cat Verb)
                      (number sing)
                      (person 3rd))))
 (beautiful-unit
            (syn-cat ((lex-cat Adjective))))
 (Mary-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))).                    (1)
```

This structure is still flat in the sense that all units except for the `Top-unit` are defined at the same level: all are direct subunits of the `Top-unit`. Also, the transitive SVO construction as defined in the previous chapter (see (9) on page 45) does not apply.[3] What is needed is some construction that takes together the adjective "beautiful" and the NP "Mary" and puts them together in a new unit of lexical category `NP`. In other words, we need a rule that transforms the structure (1) into:[4]

```
((Top-unit (syn-subunits
              (John-unit kiss-unit beautiful-Mary-unit))
           (form ((meets John-unit kiss-unit)
                  (meets kiss-unit beautiful-Mary-unit))))
 (John-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (kiss-unit (syn-cat ((lex-cat Verb)
                      (number sing)
                      (person 3rd))))
```

this is left out.

[3]Strictly speaking it would not even apply in the absence of the adjective "beautiful" because we changed the lexical categories of "John" and "Mary" from `Proper-Noun` to `NP`, however, there is a more fundamental problem, namely that the form constraints cannot be satisfied because the verb is followed by an `adjective` instead of an `NP` or `Proper-Noun`.

[4]We do not refer to the meaning of transform as it is used in transformational-generative grammar.

```
(beautiful-Mary-unit
           (syn-subunits (beautiful-unit Mary-unit))
           (form ((meets beautiful-unit Mary-unit)))
           (syn-cat ((lex-cat NP)
                     (number sing)
                     (person 3rd))))
(beautiful-unit
           (syn-cat ((lex-cat Adjective))))
(Mary-unit (syn-cat ((lex-cat NP)
                     (number sing)
                     (person 3rd))))).                    (2)
```

The transformation from (1) to (2) involves several things.

First, a new unit called `beautiful-Mary-unit` is created as a subunit of the `Top-unit`. It takes itself the `beautiful-unit` and the `Mary-unit` as subunits. These units are no longer direct subunits of the `Top-unit`.

Second, some manipulations of the `form` feature value of the `Top-unit` was performed: it is no longer the `beautiful-unit` that follows the `kiss-unit`, but the `beautiful-Mary-unit`.

Finally, the `beautiful-Mary-unit` inherits the syntactic categories from its head constituent `Mary-unit`.

In addition, this transformation needs to be performed both while parsing and producing the sentence "John kisses beautiful Mary". Because during parsing the syntactic pole of the construction is *unified* with the original source structure, but during generation the syntactic pole is *merged* with it, this transformation should be performed both after the unification and merging phases of rule application.

The key idea to handle hierarchy is to construct a new unit as a side effect of the unification and merging processes. Specifically, we can assume that, for the example under investigation, the syntactic pole of the `Adj-NP` construction should at least contain the units shown below (as always, variable names are arbitrary):

```
((?top (syn-subunits (== ?Adjective-unit ?NP-unit))
       (form (== (meets ?Adjective-unit ?NP-unit))))
 (?Adjective-unit
       (syn-cat (== (lex-cat Adjective))))
 (?NP-unit
       (syn-cat (== (lex-cat NP)
                    (number ?number)
                    (person ?person)))))
```

This unifies with the syntactic structure given in (1) and can therefore be used to test whether a `Adj-NP` construct occurs. But it does not yet create the new `NP` unit.

This is achieved by the J-operator. Units marked with the J-operator are ignored during unification. When the construction applies, the new unit is introduced and bound to the first argument of the J-operator. The second argument should already have been bound by the unification process to the parent unit from which the new unit should depend. The third argument specifies the set of units that will be pulled into the newly created unit. The new unit can contain additional slot specifications, specified in the normal way, and all variable bindings resulting from the unification are still valid. An example is shown in below:

```
((?top (syn-subunits (== ?Adjective-unit ?NP-unit))
       (form (== (meets ?Adjective-unit ?NP-unit))))
 (?Adjective-unit
       (syn-cat (== (lex-cat Adjective))))
 (?NP-unit
       (syn-cat (== (lex-cat NP)
                    (number ?number)
                    (person ?person))))
 ((J ?new-unit ?top (?Adjective-unit ?NP-unit))
       (syn-cat ((lex-cat NP)
                 (number ?number)
                 (person ?person)))))                    (3)
```

Besides creating the new unit and adding its features, the overall structure should change as well. Specifically, the new-unit is declared a subunit of its second argument (i.e. `?top` in the example above) and all feature values specified in this parent unit are moved to the new unit (i.e. the specified parts of the `syn-subunits` and `form` feature values above.) This way the precedence relations in the form-slot of the original parent or any other categories that transcend single units (like intonation) are automatically moved to the new unit as well. If there are any relations left in the original parent which involve only some of the units now taken by the new unit, then all occurrences of these unit names are replaced by the new unit name. This ensures for example that the precedence relations which still remain in the original parent unit now involve the new unit instead of the units that were pushed down into the hierarchy.

Thus the example syntactic structure (1) for "John kisses beautiful Mary" is indeed transformed into the target structure (2) by applying the pole (3). The operation is illustrated graphically in figure 4.2. Note that now the new np-unit is itself a unit ready to be part of for example a transitive construction.

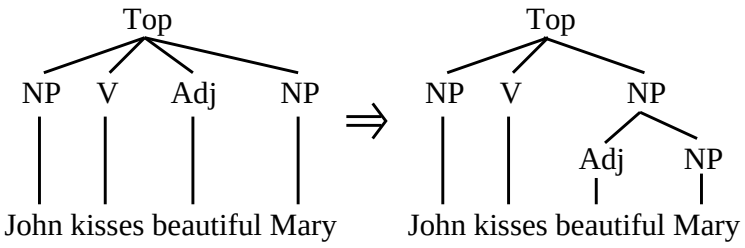


Figure 4.2: Restructuring performed by the J-operator

More formally, the J-operator is a tertiary operator and is written as (variable names are arbitrary):

```
(J ?new-unit ?parent (?child-1 ... ?child-n))
```

The second argument is called the parent argument of the J-operator and the third the children-argument.

Any unit-name in the left or right pole of a rule may be governed by a J-operator. The parent and children-arguments should refer to other units in the same pole. The children-argument is optional and the FCG engine will try to figure it out for itself if it is left unspecified.

Unifying a template containing a number of J-operators against a source structure is equivalent to unifying the template *without* the units marked by the J-operator.

Assume a set of bindings for the parent and children in the J-operator, then merging a template containing a J-operator with a target is defined as follows:

1. Unless a binding is already present, the set of bindings is extended with a binding of the first argument `?new-unit` of the J-operator to a new constant unit-name `N`.

2. A new template is constructed by removing the unit marked by the J-operator from the original template and adding a unit with name `N`. This new unit has as slots the union of the slots of the unit marked by the J-operator and of the unit in the original template as specified by the parent argument. This template is now merged with the target structure.

3. The feature values of the parent specified in the J-pattern are removed from the corresponding unit in the target structure and the new unit is made a subunit of this unit.

4. Finally, all remaining references to units specified by the children are replaced by references to the new unit.

The subsequent steps are illustrated below for the original source structure (1) and pattern (3), which are for convenience repeated as well. The source structure was:

```
((Top-unit (syn-subunits
              (John-unit kiss-unit beautiful-unit Mary-unit))
           (form ((meets John-unit kiss-unit)
                  (meets kiss-unit beautiful-unit)
                  (meets beautiful-unit Mary-unit))))
 (John-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (kiss-unit (syn-cat ((lex-cat Verb)
                      (number sing)
                      (person 3rd))))
 (beautiful-unit
            (syn-cat ((lex-cat Adjective))))
 (Mary-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))),                    (1)
```

while the template is given by:

```
((?top (syn-subunits (== ?Adjective-unit ?NP-unit))
       (form (== (meets ?Adjective-unit ?NP-unit))))
 (?Adjective-unit
       (syn-cat (== (lex-cat Adjective))))
 (?NP-unit
       (syn-cat (== (lex-cat NP)
                    (number ?number)
                    (person ?person))))
 ((J ?new-unit ?top (?Adjective-unit ?NP-unit))
       (syn-cat ((lex-cat NP)
                 (number ?number)
                 (person ?person))))).                     (3)
```

Because the J-unit is ignored, unification is successful and results in the following bindings:

```
[?top/Top-unit, ?Adjective-unit/beautiful-unit,
 ?NP-unit/Mary-unit, ?number/sing, ?person/3rd].
```

Because the variable `?new-unit` is still unbound, it is bound to a new constant unit name, say `New-unit`. With these bindings, the new template (step 2) becomes:

```
((Top-unit (syn-subunits (== beautiful-unit Mary-unit))
           (form (== (meets beautiful-unit Mary-unit))))
 (beautiful-unit
           (syn-cat (== (lex-cat Adjective))))
 (Mary-unit
           (syn-cat (== (lex-cat NP)
                        (number sing)
                        (person 3rd))))
 (New-unit (syn-subunits (== beautiful-unit Mary-unit))
           (form (== (meets beautiful-unit Mary-unit))))
           (syn-cat ((lex-cat NP)
                     (number sing)
                     (person 3rd))))).
```

Merging this with the source structure (1) results in:

```
((Top-unit  (syn-subunits
               (John-unit kiss-unit beautiful-unit Mary-unit))
            (form ((meets John-unit kiss-unit)
                   (meets kiss-unit beautiful-unit)
                   (meets beautiful-unit Mary-unit))))
 (John-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (kiss-unit (syn-cat ((lex-cat Verb)
                      (number sing)
                      (person 3rd))))
 (New-unit  (syn-subunits (beautiful-unit Mary-unit))
            (form ((meets beautiful-unit Mary-unit))))
            (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (beautiful-unit
            (syn-cat ((lex-cat Adjective))))
 (Mary-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))).
```

Step 3 repairs the hierarchical structure and deletes the feature values that were moved to the new unit:

```
((Top-unit  (syn-subunits
               (John-unit kiss-unit New-unit))
            (form ((meets John-unit kiss-unit)
                   (meets kiss-unit beautiful-unit))))
 (John-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (kiss-unit (syn-cat ((lex-cat Verb)
                      (number sing)
                      (person 3rd))))
 (New-unit  (syn-subunits (beautiful-unit Mary-unit))
            (form ((meets beautiful-unit Mary-unit))))
            (syn-cat ((lex-cat NP)
                     (number sing)
                     (person 3rd)))))
 (beautiful-unit
            (syn-cat ((lex-cat Adjective))))
 (Mary-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))).
```

Finally, in step 4, all remaining occurrences of the names of units that were absorbed by the new unit are replaced by the name of the new unit:

```
((Top-unit  (syn-subunits
               (John-unit kiss-unit New-unit))
            (form ((meets John-unit kiss-unit)
                   (meets kiss-unit New-unit))))
 (John-unit (syn-cat ((lex-cat NP)
                      (number sing)
                      (person 3rd))))
 (kiss-unit (syn-cat ((lex-cat Verb)
                      (number sing)
                      (person 3rd))))
 (New-unit  (syn-subunits (== beautiful-unit Mary-unit))
            (form (== (meets beautiful-unit Mary-unit))))
            (syn-cat ((lex-cat NP)
                     (number sing)
                     (person 3rd)))))
```

```
(beautiful-unit
          (syn-cat ((lex-cat Adjective))))
(Mary-unit (syn-cat ((lex-cat NP)
                     (number sing)
                     (person 3rd))))).                    (4)
```

Note that the application of a construction now actually consists of three instead of two phases: first the unification of a pole against the source structure, next the merging of the *same* structure with the altered pole as specified above and finally the normal merging phase between the rule's other pole and the corresponding source structure (although here too J-operators might be involved).

Note also that no additional mechanisms are needed to implement the transfer from daughter to parent. Indeed, in (3), because the new-unit has its number category specified by the same variable `?number` as is used for unifying with the `?NP-unit`, it will inherit the number of its 'head' constituent as specified. On the other hand, note that this transfer mechanism is not as automatic as it is the case in for example HPSG with the so called head feature principle: what is to be transferred has to be stated explicitly.

4.2 Hierarchy in Semantics

The mechanisms proposed so far implement the basic mechanism of syntactically grouping units together in more encompassing units. Any grammar formalism supports this kind of grouping. However constructions have both a syntactic and a semantic pole, and so now the question is how hierarchy is to be handled semantically in tight interaction with syntax.

Assume that the relevant parts of the semantic structure after performing a lexical analysis of the phrase "John kisses beautiful Mary" looks as follows:

```
((Top-unit (sem-subunits
              (John-unit kiss-unit beautiful-unit Mary-unit)))
 (John-unit ...)
 (kiss-unit ...)
 (beautiful-unit
          (sem-cat ((feature ?b)))
          ...)
 (Mary-unit (sem-cat ((object ?m)))
          ...)                                            (5)
```

The function of our `Adj-NP` construction is now on the one hand to identify the variables `?b` and `?m`, and on the other to again create a new unit absorbing the `beautiful` and `Mary` units.

Again the J-operator can be used with exactly the same behavior as seen in the previous section. The J-unit is ignored during unification but introduces a new unit during merging as explained before: the new unit is linked to the parent and the parent slot specifications contained in the template are moved to the new unit.

Hence, the semantic pole of the `Adj-NP` construction looks as follows:

```
((?top (sem-subunits (== ?Adjective-unit ?NP-unit)))
 (?Adjective-unit
       (sem-cat ((feature ?o))))
 (?NP-unit
       (sem-cat ((object ?o))))
 ((J ?new-unit ?top (?Adjective-unit ?NP-unit))
       (sem-cat ((object ?o)))))                          (6)
```

While parsing, the unification phase of the syntactic structure (1) and the syntactic pole of the construction (3) already yielded bindings for the variables in (6):

```
[?top/Top-unit, ?Adjective-unit/beautiful-unit,
 ?NP-unit/Mary-unit, ?number/sing, ?person/3rd].
```

Also the `?new-unit` variable received a binding to a new constant `New-unit`. With these bindings a new template can be constructed from (6):

```
((Top-Unit
       (sem-subunits (== beautiful-unit Mary-unit)))
 (beautiful-unit
       (sem-cat ((feature ?o))))
 (Mary-unit
       (sem-cat ((object ?o))))
 (New-Unit
       (sem-subunits (== beautiful-unit Mary-unit))
       (sem-cat ((object ?o))))).
```

Merging this pattern with the source structure (5) yields:

```
((Top-unit (sem-subunits
              (John-unit kiss-unit beautiful-unit Mary-unit)))
 (John-unit ...)
 (kiss-unit ...)
 (New-Unit  (sem-subunits (beautiful-unit Mary-unit))
            (sem-cat ((object ?b))))
```

```
(beautiful-unit
           (sem-cat ((feature ?b)))
           ...)
(Mary-unit (sem-cat ((object ?b)))
           ...)).
```

Note that the `sem-cat` variables of the `New-unit`, the `beautiful-unit` and the `Mary-unit` are all equal after merging, which was as intended. Finally, the remaining steps repair the hierarchical structure:

```
((Top-unit  (sem-subunits
               (John-unit kiss-unit New-Unit)))
 (John-unit ...)
 (kiss-unit ...)
 (New-Unit  (sem-subunits (beautiful-unit Mary-unit))
            (sem-cat ((object ?b))))
 (beautiful-unit
            (sem-cat ((feature ?b)))
            ...)
 (Mary-unit (sem-cat ((object ?b)))
            ...)).
```

The entire construction thus looks as follows:

```
((?top (sem-subunits (== ?Adjective-unit ?NP-unit)))
 (?Adjective-unit
       (sem-cat (== (feature ?o))))
 (?NP-unit
       (sem-cat (== (object ?o))))
 ((J ?new-unit ?top (?Adjective-unit ?NP-unit))
       (sem-cat ((object ?o)))))
<-->
((?top (syn-subunits (== ?Adjective-unit ?NP-unit))
       (form (== (meets ?Adjective-unit ?NP-unit))))
 (?Adjective-unit
       (syn-cat (== (lex-cat Adjective))))
 (?NP-unit
      (syn-cat (== (lex-cat NP)
                   (number ?number)
                   (person ?person))))
 ((J ?new-unit ?top (?Adjective-unit ?NP-unit))
      (syn-cat ((lex-cat NP)
```

```
(number ?number)
(person ?person))))).                    (7)
```

To summarize, we have introduced a new operator called the J-operator which allows us to specify how the hierarchical structure should be changed during rule application. Rule application now proceeds in three major steps. In the first step, one pole of the rule (the semantic pole for production and the syntactic pole in parsing) is unified with the corresponding unit structure. Units marked with a J-operator are ignored during this phase. The rule applies if unification is successful and results in a set of bindings.

In the second step, a new template is created from the pole, adding new units to it for every unit marked by the J-operator. Merging this new template with the source structure and repairing the hierarchical structure yields the new output structure.

In the third and final step, a new template is constructed from the rule's opposite pole (the syntactic pole in production or the semantic pole in parsing), using the bindings obtained in the first and second steps. This template is then merged with the corresponding unit structure (the syntactic structure in production or the semantic structure in parsing), and the hierarchical structure is repaired. This is summarized in table 4.1.[5]

4.3 Additional Uses of the J-operator

In this section we will explore some additional uses of the J-operator. To begin with, we have been rather vague about how the initial semantic and syntactic structures are constructed. For example, in parsing we have assumed that the agent directly constructs a syntactic and semantic structure based on word boundaries and containing units for every word in the sentence that needs to be parsed. However, some words are made out of several morphemes, contain suffixes etc. Also, in production, when a meaning has to be mapped onto a form, it is unclear in advance which units should be created, and this is certainly even language dependent: the particular language one speaks might determine how the meaning should be fragmented and which fragments are mapped onto words or morphemes or other type of constructs. The J-operator can be used to model this.

To illustrate this, we'll redo the exercise of parsing and producing the sentence "Mary kisses John", but this time in some more detail. [6]

[5]The notion of 'hat' of a unit structure will be defined later, for now one can safely ignore it and therefore it is put in between parentheses in the table.

[6]It should be kept in mind that most of the FCG examples presented in this thesis are not in

Step	Description	result (if successful)
Production:		
(1)	unification of the semantic pole and the (hat of the) semantic structure	a (set of) set of bindings
(2)	Construction of new semantic template from semantic pole which is merged with the semantic structure	new semantic structure
(3)	Construction of syntactic template from syntactic pole which is merged with the syntactic structure	new syntactic structure
Parsing:		
(1)	unification of the syntactic pole and the (hat of the) syntactic structure	a (set of) set of bindings
(2)	Construction of new syntactic template from syntactic pole which is merged with the syntactic structure	new syntactic structure
(3)	Construction of semantic template from semantic pole which is merged with the semantic structure	new semantic structure

Table 4.1: Summary of steps taken during the application of a rule.

4.3.1 Parsing Mary kisses John

Instead of assuming that the initial syntactic structure already contains units for the different words in this sentence, we'll assume it to be simply:

any way authoritative. In particular, they are not in any way motivated from a linguistics or psycho-linguistics point of view, they merely serve as examples with a purpose of illustrating FCG and its possibilities.

```
((Top-unit                                                    (1)
    (form ((string s1 "Mary")
           (string s2 "kiss")
           (string s3 "es")
           (string s4 "John")
           (meets s1 s2) (meets s2 s3) (meets s3 s4))))),
```

specifying that 4 strings were observed in a particular order.[7] The corresponding initial semantic structure during parsing is simply:

```
((Top-unit))                                                  (2)
```

The syntactic pole of the following lexical construction unifies with the syntactic structure because the J-unit is ignored:

```
((?Top (meaning (== (Mary ?m))))
 ((J ?new ?Top) (sem-cat ((person ?m) (object ?m)))))
<-->
((?Top (form (== (string ?new "Mary"))))
 ((J ?new ?Top)
       (form (== (stem "Mary")))
       (syn-cat ((constituent NP)
                 (number sing)
                 (person 3rd)))))                           (3)
```

The variable `?new`, which is the second argument to the J-unit but is also contained in the form feature of the `?Top` unit, gets bound to the symbol `s1`. Hence, the extra merging phase changes the syntactic structure into:

```
((Top-unit
     (syn-subunits (s1))
     (form ((string s2 "kiss")
            (string s3 "es")
            (string s4 "John")
            (meets s1 s2) (meets s2 s3) (meets s3 s4))))
 (s1 (form ((string s1 "Mary") (stem s1 "Mary")))
```

[7] One could of course ask the question how an agent can know what a string is, i.e. how he should know that the character sequence "marykissesjohn" actually exists out of 4 parts. And indeed we do assume that an agent is at least capable of determining the different characters making up an observed sentence, but apart from that the analysis presented here could easily be made more accurate by e.g. splitting the sequence into 5 or more parts as in "Ma"+"ry"+"kiss"+"es"+"John" or "M"+"a"+...+"o"+"n". For the sake of clarity this is not done in the text.

```
        (syn-cat ((lex-cat Proper-Noun)
                  (number sing)
                  (person 3rd)))))
```

Finally, merging the semantic pole with the semantic structure yields:

```
   ((Top-unit
        (sem-subunits (s1)))
    (s1 (sem-cat ((person ?m)))
        (meaning ((Mary ?m))))).
```

Similar constructions can be defined for the strings "John" and "kisses". After the application of these constructions, the semantic and syntactic structures become:

```
   ((Top-unit
       (sem-subunits (s1 s2 s4)))
    (s4
       (meaning ((john ?j)))
       (sem-cat ((person ?j))))
    (s1
       (meaning ((mary ?m)))
       (sem-cat ((person ?m))))
    (s2
       (meaning ((kiss ?k) (kisser ?k ?a) (kissee ?k ?j)))
       (SEM-CAT ((event ?k)
                 (agent ?k ?a) (patient ?k ?p))))),          (4)
```

and

```
((Top-unit
    (form
     ((string s3 es)
      (meets s1 s2)
      (meets s2 s3)
      (meets s3 s4)))
    (syn-subunits (s1 s2 s4)))
 (s2
    (form ((stem "John") (string s4 "John")))
    (syn-cat
     ((constituent NP) (lex-cat proper-noun)
      (number sing) (person 3rd))))
```

```
(s1
   (form ((stem "Mary") (string s1 "Mary")))
   (syn-cat
    ((constituent NP) (lex-cat proper-noun)
     (number sing) (person 3rd))))
(s2
   (form ((stem "kiss") (string s2 "kiss")))
   (syn-cat
    ((constituent V) (lex-cat verb regular)
     (number ?n) (person ?per)))))                          (5)
```

respectively.

As explained, we'll need an SVO construction to identify the appropriate variables in the different units:

```
(def-con-rule SVO
  ((?top (sem-subunits (== ?subject ?predicate ?object)))
   (?subject
         (sem-cat ((person ?s))))
   (?predicate
         (sem-cat
           (TAG ?sem-cat
                (== (agent ?p ?s) (patient ?p ?o)))))
   (?object
         (sem-cat (((OR person object) ?o))))
   ((J ?new ?top)
         (sem-cat ?sem-cat)
         (head ?predicate)))
  <-->
  ((?top (syn-subunits (== ?subject ?predicate ?object))
         (form (== (meets ?subject ?predicate)
                   (meets ?predicate ?object))))
   (?subject (syn-cat (== (constituent NP)
                          (number ?num)
                          (person ?per))))
   (?predicate (syn-cat (TAG ?syn-cat
                             (== (constituent V)
                                 (number ?num)
                                 (person ?per)))))
   (?object (syn-cat (== (constituent NP))))
   ((J ?new ?top) (syn-cat ?syn-cat)))).                    (6)
```

This construction imposes some word order constraints on the phrase and enforces agreement in number and person between the verb and the subject.

It also uses some features not encountered before, like the `TAG` operator. This operator has the form

```
(TAG ?variable pattern).
```

Whenever such a list is unified against a source, first only the `pattern` element of the list is unified with the source. If successful, this results in a list of bindings lists. Finally the entire source is bound to the `?variable` variable. For example, unifying the pattern

```
(TAG ?syn-cat (== (constituent V)
                  (number ?num)
                  (person ?per)))))
```

with the list

```
((constituent V) (lex-cat verb regular)
 (number 3rd) (person sing)),
```

results in the following set of bindings:

```
((?num . sing) (?per . 3rd)
 (?syn-cat . ((constituent V) (lex-cat verb regular)
              (number 3rd) (person sing)))).
```

This allows to select on only a subset of feature values (like the `constituent`, `number` and `person` features) but at the same time also allows to reference to the entire set of feature values (in this case also including the `lex-cat feature`.) This is similar to the use of tags in other unification based formalisms like HPSG. Another feature not encountered before is the `OR` special operator: a list `(OR p1 ... pn))` unifies with every source that unifies with one of the elements `p1` to `pn`.

However, after applying the lexical constructions, the transitive construction (6) does not apply because the "es" suffix is not processed as being part of the verb yet; it still 'stands in the way'. This is most easily seen in the `meets` constraints in the `Top` unit of the syntactic structure. The top part of Figure 4.3 illustrates the problem: first an "es" suffix construction needs to be applied before the transitive construction can be.

Let's now proceed with defining an "es"-suffix construction. While parsing, this construction should recognize the "es" string directly following a regular verb form. Therefore, the right pole of the construction should look like:

```
((?top (syn-subunits (== ?verb-unit))
       (form (== (meets ?verb-unit ?es)
                 (string ?es "es"))))
 (?verb-unit
       (syn-cat (== (lex-cat verb regular))))
 ((J ?es ?top)
  (syn-cat ((constituent V)
            (number sing)
            (person 3rd))))).                                    (7)
```

This pole unifies with any structure that contains a `?verb-unit` subunit which is a regular verb and that is followed by an "es"-suffix.

Notice also that when applied, a new `es` unit will be created that covers the verb unit and that is of the appropriate syntactic category as required by the transitive construction: it will be a `(constituent V)` and have the correct person and number values to agree with the subject of "Mary":

```
((Top-unit
     (form
      ((meets s1 e3)
       (meets s3 s4)))
     (syn-subunits (s1 s4 es)))
 (s4 ...)
 (s1 ...)
 (s3 (syn-subunits (s2)
     (form ((STRING s3 "es") (MEETS s2 s3) ))
     (syn-cat ((constituent V)
               (number sing)
               (person 3rd)))))
 (s2 ...)).                                                      (8)
```

Structure (8) corresponds to the second phrase structure in the top of figure 4.3. Note that the appropriate changes are made to the top unit's `form` feature so that the transitive construction indeed applies (as far as the syntactic side is concerned.)

At first sight the "es"-suffix construction doesn't need a semantic pole since no meaning is added or no equalities need to be resolved. However, the pole is needed to change the hierarchical structure of the semantic structure in accordance to the changes made on the syntactic side. If not, the semantic pole of the transitive construction would not merge properly with it. Hence, a first attempt for the semantic pole of the "es"-suffix construction could look like:

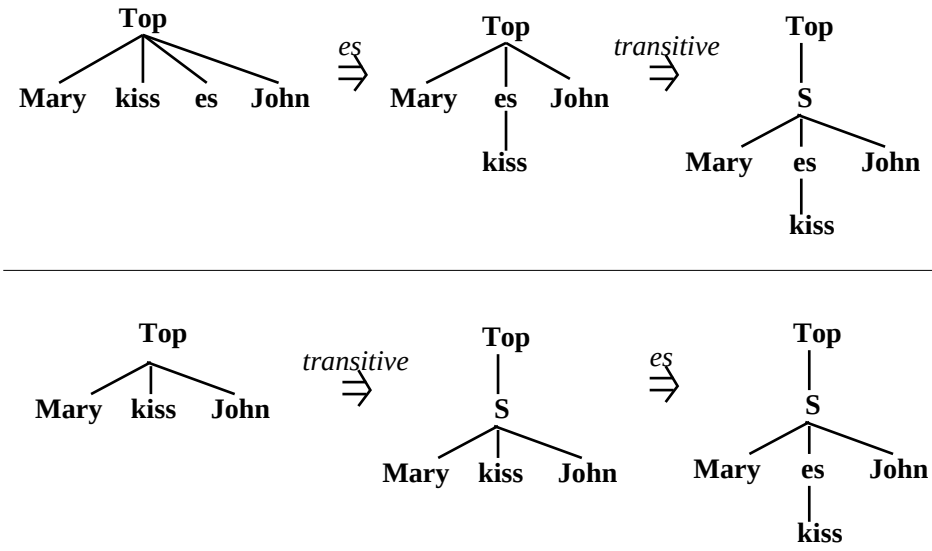


Figure 4.3: Top: subsequent structures when parsing the sentence "Mary kiss es John". The left structure is the result of applying the lexical constructions for "John", "kiss" and "Mary" to the initial structures (1) and (2). This is transformed into the middle structure by the "es"-suffix construction, which is finally transformed into the right most structure after applying the transitive construction. **Bottom:** subsequent structures when producing the sentence "Mary kiss es John". The left structure is the result of applying the lexical constructions for "John", "kiss" and "Mary" to the initial structures (10) and (11). This is transformed into the middle structure by the transitive construction, which is finally transformed into right most structure after applying the "es"-suffix construction.

```
((?top (sem-subunits (== ?verb-unit))
       (head ?verb-unit))
 (?verb-unit
       (sem-cat ?sem-cat))
 ((J ?es ?top)
       (sem-cat ?sem-cat))).                                    (9)
```

When merged with the semantic structure (4), this pole will indeed introduce the `s3` unit as a new parent unit of the `s2` unit and inheriting the `sem-cat` feature from it as required by the transitive construction. Unfortunately, things are not as simple as that. To see why, we'll have to look at the production of "Mary kisses John".

4.3.2 Producing Mary kisses John

For production, the initial semantic and syntactic structures look like:

```
((Top
     (meaning ((Mary M)
               (kiss K) (kisser K M) (kissee K J))
               (John J)))),                                    (10)
```

and

```
((Top)).                                                       (11)
```

After the application of the lexical and transitive constructions these are transformed into:

```
((top
    (sem-subunits (s)))
 (s
    (head kiss)
    (sem-cat ((agent k m) (patient k j)))
    (sem-subunits (mary kiss john)))
 (kiss
    (meaning ((kiss k) (kisser k m) (kissee k j)))
    (referent k)
    (sem-cat ((agent k m) (patient k j))))
 (john
    (meaning ((john j)))
    (referent j))
 (mary
    (meaning ((mary m)))
    (referent m))),                                            (12)
```

and

```
((top
    (syn-subunits (s)))
 (s
    (form ((meets mary kiss) (meets kiss john)))
    (syn-cat
     ((constituent v) (lex-cat verb regular)
      (number sing) (person 3rd)))
    (syn-subunits (mary kiss john)))
 (kiss
```

```
      (form ((stem "kiss") (string kiss "kiss")))
      (syn-cat
       ((constituent v) (lex-cat verb regular)
        (number sing) (person 3rd))))
   (john
      (form ((stem "John") (string john "John")))
      (syn-cat
       ((constituent np) (lex-cat proper-noun)
        (number sing) (person 3rd))))
   (mary
      (form ((stem "Mary") (string mary "Mary")))
      (syn-cat
       ((constituent np) (lex-cat proper-noun)
        (number sing) (person 3rd)))))                    (13)
```

respectively. As can be seen, the use of the J-unit allows to cut up the initial meaning according to the available lexical constructions.

Now the "es"-suffix construction still needs to be applied. This is illustrated in the bottom part of figure 4.3. For reasons that will be explained in the next chapter, the applicability of a construction is determined on the basis of the *hat* of the current structure only. The hat of a structure consists of only the structure's top unit and its direct children. For example, the hat of structure (12), which is the one that should trigger the "es"-suffix construction (see figure 4.3) consists of the `Top` and `S` units only.

In this case, this means that no candidate will be found for the *?verb-unit* in pole (9) when it is unified with it (12). During the merging phases of rule application however, the entire structure is available. This allows us to illustrate another use of the J-operator. Recall that every unit marked with the J-operator is ignored during the unification phase. Hence, if we change pole (9) into:

```
((?top (sem-subunits (== ?verb-unit))
       (head ?verb-unit))
 ((J ?verb-unit NIL)
       (sem-cat ?sem-cat))
 ((J ?es ?top)
       (sem-cat ?sem-cat))),                              (14)
```

then the `?verb-unit` needn't be present during the unification phase. Still, the `?verb-unit` variable will receive a binding because it is also present as a semantic subunit in the `?top` unit. This means that no additional unit will be created but changes will be made to the *existing* unit specified by the `?verb-unit` variable. And because the second argument to the J-operator in `(J ?verb-unit`

NIL) is not specified (written explicitly as NIL), application of this J-operator during the merging phase will not change the hierarchical structure or move any features around.

In sum then, the J-operator can also be used to change existing units without making changes to the hierarchical structure or moving around features. In this particular example there are not even changes made to the (existing) ?verb-unit, merely the ?sem-cat variable is given the appropriate binding such that this information, which is required by the transitive construction, can be propagated to the newly created ?es unit.

To be complete, note that there is an additional problem with the proposed "es"-suffix construction, this time on the syntactic side. To see this, consider the production of the sentence "I kiss Mary" starting from the following initial semantic structure:

```
((Top-unit
     (meaning ((Speaker S)
               (kiss K) (kisser K S) (kissee K M))
               (Mary M)))),
```

After applying the usual set of lexical and transitive (SVO) constructions, the syntactic structure will look as follows:

```
((Top-unit
    (syn-subunits (s)))
 (s
    (form ((meets i kiss) (meets kiss mary)))
    (syn-cat
     ((constituent v) (lex-cat verb regular)
      (number sing) (person 3rd)))
    (syn-subunits (mary kiss john)))
 (kiss
    (form ((stem "kiss") (string kiss "kiss")))
    (syn-cat
     ((constituent v) (lex-cat verb regular)
      (number sing) (person 1st))))
 (i
    (form ((stem "I") (string I "I")))
    (syn-cat
     ((constituent np) (lex-cat proper-noun)
      (number sing) (person 1st))))
 (Mary
```

```
      (form ((stem "Mary") (string Mary "Mary")))
      (syn-cat
       ((constituent np) (lex-cat proper-noun)
        (number sing) (person 3rd)))))).
```

The main difference with structure (12) to which the "es"-suffix construction still needs to be applied, is that now the `kiss` unit contains a `(person 1st)` instead of a `(person 3rd)` value. Hence the "es"-suffix construction should not apply, but it does. Indeed, when the syntactic pole (7) of the "es" construction:

```
((?top (syn-subunits (== ?verb-unit))
       (form (== (meets ?verb-unit ?es)
                 (string ?es "es"))))
 (?verb-unit
       (syn-cat (== (lex-cat verb regular))))
 ((J ?es ?top)
  (syn-cat ((constituent V)
            (number sing)
            (person 3rd))))),                              (7)
```

is merged with the above structure, the `(person 3rd)` value would simply be added to the `kiss` unit, which is clearly not intended. This can be prevented by changing the pole into:

```
((?top (syn-subunits (== ?verb-unit))
       (form (== (meets ?verb-unit ?es)
                 (string ?es "es"))))
 (?verb-unit
       (syn-cat (==1 (lex-cat verb regular))
                     (number sing)
                     (person 3rd))))
 ((J ?es ?top)
  (syn-cat ((constituent V)
            (number sing)
            (person 3rd))))).
```

Two changes are made: first the includes directive `==` in the `syn-cat` value of the `?verb-unit` is changed into an *includes-uniquely* directive `==1`. This instructs the merger that the result of merging may not contain elements with identical names. Hence, although the pattern

```
(== (person 3rd))
```

merges with the source

```
(== (person 1st))
```

resulting in the list

```
((person 3rd) (person 1st)),
```

the pattern

```
(==1 (person 3rd))
```

does not merge with the given source because the result would contain two elements named `person`.

Having explained this, it is now easy to understand the second change to the pole which consists of adding the elements `(number sing)` and `(person 3rd)` to the pole. Together with the includes-uniquely directive this will prevent the "es"-suffix construction to apply *unless the verb is actually of lexical category* `regular-verb` *and in the third person singular form.*

The entire set of rules needed in this section and some output of running them is given in appendix B.

4.3.3 Dependency Grammar in FCG

All FCG examples presented so far more or less follow a construction grammar and constituent structure approach to grammar. This will also be the case in the following chapters, where a number of simulation experiments will be set up using similar types of FCG rules as those presented so far. This choice was made for no particular reason. The goal was to set up experiments in which compositional, hierarchical and recursive language emerges when this increases communicative success. But whether recursion is captured in rules complying to construction grammar or to any other theory of language is of minor concern; we are confident that the same point can be made in all cases.

However, to at least show that other choices are possible in FCG, we'll end this chapter with giving a small example resembling more a dependency grammar approach to language. As was mentioned briefly in the introduction of Chapter 3, the basic assumption in dependency grammars is that syntactic structure consists of lexical elements linked by binary asymmetrical relations called dependencies. Thus, the common formal property of dependency structures, as compared to representations based on constituency, is the lack of phrasal nodes (Nivre, 2005). This can be seen by comparing the constituency representation of the English phrase "beautiful Mary kisses John" in Figure 4.4, to the corresponding dependency representation in Figure 4.5. The dependency relations are represented with labeled arrows. The label SBJ stands for subject, OBJ for object and NMOD for (noun) modifier.

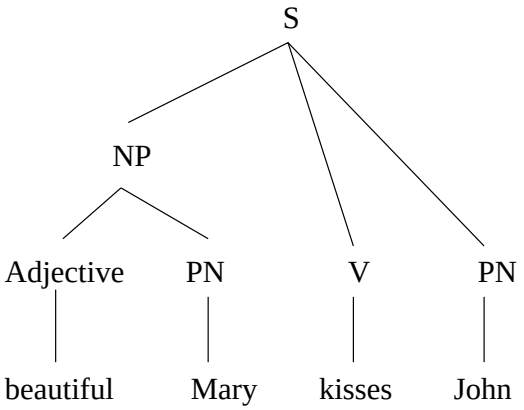


Figure 4.4: Constituent structure of the English phrase "beautiful Mary kisses John". S stands for sentence, NP for Noun Phrase and V for Verb.

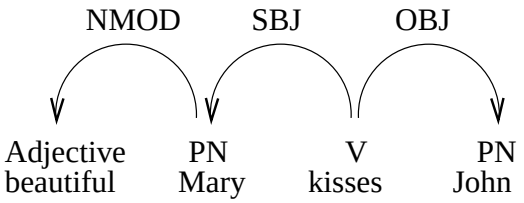


Figure 4.5: Dependency structure of the English phrase "beautiful Mary kisses John". SBJ stands for subject, OBJ for object and NMOD for modifier of a noun.

We have already shown how a constituency representation can be captured in FCG. We will now show one way in which a dependency analysis could be performed if desired. We will restrict ourselves to analyzing the phrase "beautiful Mary", and we will assume that the lexical analysis in both cases is the same. Hence, the starting point in production could be the following semantic unit structure:

```
((Top (sem-subunits (beatiful-unit Mary-unit)))
 (beatiful-unit
      (sem-cat ((feature X))
      (meaning ((beautiful X)))))
 (Mary-unit
      (sem-cat ((object X))
      (meaning ((Mary X))))).                                    (1)
```

The corresponding syntactic structure would then be:

```
((Top (syn-subunits (beatiful-unit Mary-unit)))
 (beatiful-unit
      (syn-cat ((lex-cat Adjective)))
      (form ((string beatiful-unit "beatiful"))))
 (Mary-unit
      (syn-cat ((lex-cat NP)
                (number singular)
                (person 3rd)))
      (form ((string Mary-unit "Mary")))))                       (2)
```

.

A constituent structure analysis would now require a linking construction like the one labeled (7) at the end of section 4.2 and repeated below:

```
((?top (sem-subunits (== ?Adjective-unit ?NP-unit)))
 (?Adjective-unit
       (sem-cat (== (feature ?o))))
 (?NP-unit
       (sem-cat (== (object ?o))))
 ((J ?new-unit ?top (?Adjective-unit ?NP-unit))
       (sem-cat ((object ?o)))))
<-->
((?top (syn-subunits (== ?Adjective-unit ?NP-unit))
       (form (== (meets ?Adjective-unit ?NP-unit))))
 (?Adjective-unit
       (syn-cat (== (lex-cat Adjective)))))
```

```
(?NP-unit
      (syn-cat (== (lex-cat NP)
                   (number ?number)
                   (person ?person))))
((J ?new-unit ?top (?Adjective-unit ?NP-unit))
      (syn-cat ((lex-cat NP)
                (number ?number)
                (person ?person))))).                    (3)
```

This construction would create a new, phrasal subunit of the top unit, covering both the `beautiful` and `Mary` units and inheriting the semantic and syntactic category values of its head constituent (the `Mary-unit`).

In a dependency grammar analysis on the other hand phrasal units are not allowed. Instead, the `beautiful-unit` should be recognized as being a modifier of its `Mary` head unit and a dependency link should be created between them accordingly. This is achieved by the following rule:

```
((?top (sem-subunits (== ?Adjective-unit)))
 (?Adjective-unit
      (sem-cat (== (feature ?o))))
 (?NP-unit
      (sem-cat (== (object ?o))))
 ((J ?Adjective-unit NIL)
      (dependency ((NMOD ?NP-unit))))
 ((J ?NP-unit ?top)))
<-->
((?top (syn-subunits (== ?Adjective-unit))
       (form (== (meets ?Adjective-unit ?NP-unit))))
 (?Adjective-unit
      (syn-cat (== (lex-cat Adjective))))
 (?NP-unit
      (syn-cat (== (lex-cat NP))))
 ((J ?Adjective-unit NIL)
      (dependency ((NMOD ?NP-unit))))
 ((J ?NP-unit ?top)))                                    (4)
```

The binary head-modifier relation is captured by on the one hand the `subunits` features (for the head) and on the other by introducing a new feature called `dependency` (for the modifier), which in this case will have a value of `((NMOD ?NP-unit))`. It should be clear that other and more elaborate choices are also possible. Notice how the problem of inheritance (the percolation of semantic and syntactic

category values to a new phrasal unit) do not have to be taken care of since the head unit (the `?NP-unit`) remains at its original hierarchical level.

The result of applying rule (3) to the structures in (1) and (2) are the following semantic and syntactic structures:

```
((Top (sem-subunits (Mary-unit)))
 (Mary-unit
      (sem-subunits (beatiful-unit))
      (sem-cat ((object X))
      (meaning ((Mary X)))))
 (beatiful-unit
      (dependency ((NMOD Mary-unit)))
      (sem-cat ((feature X))
      (meaning ((beautiful X))))),
```

and

```
((Top (syn-subunits (Mary-unit)))
 (Mary-unit
      (syn-cat ((lex-cat NP)
                (number singular)
                (person 3rd)))
      (form ((meets (beatiful-unit Mary-unit))
             (string Mary-unit "Mary"))))
 (beatiful-unit
      (dependency ((NMOD Mary-unit)))
      (syn-cat ((lex-cat Adjective)))
      (form ((string beatiful-unit "beatiful"))))).
```

These correspond to the dependency structure of Figure 4.5. It should be clear that parsing is also possible with the same rule and results in a similar set of unit structures. This concludes the current chapter.

Chapter 5

Problem Setting

In the following chapters we'll use the machinery developed so far to perform a number of simulation experiments that will allow us to test the main hypothesis of this thesis, namely that when a population of language users is bootstrapping a language from scratch then compositional and recursive language emerges because it allows the agents to re-use already established bits of language and hence increase their communicative success.

The experiments will be language game experiments. More concretely, they will consist of a number of pairwise communicative interactions between members of a population of language users. Every interaction two agents will be selected randomly from the population. One of them is the speaker, the other is the hearer. Both agents are presented with a scene. The speaker has to describe some aspect of the scene to the hearer. The hearer then has to signal back whether he agrees with the description or not.

We will of course need to take care not to bias the agents towards compositionality by allowing them to prefer an alternative solution to the problem of successful communication if they wish so. The alternative will be holistic encoding and, as will be explained shortly, agents may choose to introduce and use holistic bits of language if they expect that it will improve the communication. To test whether the agents indeed make use of this opportunity under certain conditions, a number of baseline experiments will be carried out in which the agents evolve a mixed holistic-compositional language depending on the frequencies with which certain combinations of predicates need to be expressed. This will be dealt with in Chapter 7. Chapter 8 then will focus on the emergence of grammar.

Before we can move on to the simulations however, we still need to formulate the problem statement and explain the agent architecture. This will be the topic of the following two chapters.

5.1 Problem description

5.1.1 Scenes and Topics

If we want the agents to consider compositional language, then we must at least make them communicate about things that could be expressed in a compositional fashion.

The scenes about which the agents will have to communicate would in English typically be described with transitive constructs like "John kisses beautiful Mary". A simple but standard syntactic analysis of this sentence requires two constructions, the transitive construction 'S → NP V NP' and the recursive construction 'NP → Adjective NP' (S stands for Sentence, V for Verb and NP for Noun Phrase.)

As before, the meaning of phrases will be represented in first order logic. For example, the meaning of the above sentence is described as:[1]

$$(\textit{John}\ J)\ (\textit{Kiss}\ E\ J\ M)\ (\textit{beautiful}\ M)\ (\textit{Mary}\ M).$$

Each of the simple predicates $(\textit{John}\ J)$, $(\textit{beautiful}\ M)$ and $(\textit{Mary}\ M)$ contributes a fact about some object or person. For example, $(\textit{John}\ J)$ states that the thing to which the pointer J refers is John and $(\textit{beautiful}\ M)$ states that the thing to which M refers is beautiful.

The predicate $(\textit{Kiss}\ E\ J\ M)$ introduces a kiss-event. The first argument E can be used to refer to the event itself, although that will not be needed in the experiment. The second argument J denotes the kisser (the agent) and the third argument M refers to the kissee (the patient).

Because our experiment is a computational one and does not make use of robotic but of software agents that are not embodied and grounded in the real world, the actual meaning of these facts cannot make sense to the agents. They are not aware of kiss-events, or of what it means to kiss. They are not even aware of the very existence of events and objects; in fact, they are simply not aware at all.

So, the first assumption we will make for this experiment is:

Assumption 1. *All agents share the notion of a certain set of events and featured event participants.*[2]

[1] When a sequence of predicates is shown like here, it is implicitly assumed that the logical conjunction of them is taken.

[2] There have been some experiments that didn't make this assumption. For example in all of Steels (1999); Belpaeme (2001); Steels and Belpaeme (2005) and Steels and Loetzsch (2007) the agents started of with empty ontologies (although they are all aware of the same (possibly infinite) set of ontological categories.)

More concrete, we'll assume that all agents collectively are aware of some number n_e of event types, and a certain number of n_p of possible event participants, each of which can also have some number n_f of features, like their color or size etc. All of the n_e event types have two participants: an agent and a patient. Hence, we can describe them using the facts $(E_i\ ?e\ ?a\ ?p)$, $i = 1..n_e$. The n_p participants are described by the facts $(P_i\ ?o)$, $i = 1..n_p$, and their features by $(F_i\ ?f)$, $i = 1..n_f$.

Any of the n_p participants can act as an agent or a patient. The scenes $S(e, a, p)$ about which the agents have to communicate during an interaction will consist of an event e with agent a and patient p. Their descriptions are generated according to the following prescription:

1. $S(e, a, p) \rightarrow A(a) \wedge A(p) \wedge E(e, a, p)$
2. $E(e, a, p) \rightarrow (E_i\ e\ a\ p), i \in \{1, ..., n_e\}$, each with probability $1/n_e$.
3. $A(x) \rightarrow P(x)$ or $F(x) \wedge A(x)$, each with probability $1/2$.
4. $P(x) \rightarrow (P_i\ x), i \in \{1, ..., n_p\}$, each with probability $1/n_p$.
5. $F(x) \rightarrow (F_i\ x), i \in \{1, ..., n_f\}$, each with probability $1/n_f$.

We can immediately calculate a number of properties about scenes. First, the minimum scene length, counted as the number of predicates in its description, is 3 and is of the form

$$(E_i\ e\ a\ p)(N_j\ a)(N_k\ p).$$

Due to the third rule, the maximum scene length is unbound. Fortunately, the probability of generating an infinitely long scene description is zero. The length of a description of the type $A(x)$ follows a binomial distribution with average 2 and variance $1/2$. Hence, the average scene length is $1+2\times 2 = 5$, with variance $2 \times 1/2 = 1$.

Since a scene of length 5 always contains exactly two participant predicates, two feature predicates and 1 event predicate, the number of different scenes of length 5 is easily calculated to be $n_p^2 \times n_f^2 \times n_e$.

The probability of generating a scene of length l with $l \geq 3$ is given by $(l-2)(1/2)^{l-1}$. The number of scenes of length l $(l \geq 3)$ is given by $n_e n_p^2 n_f^{l-3}$.

5.1.2 Topics

Agents do not always describe the entire scene to one another. We already made a major assumption that all agents share some universal way of analyzing

and describing things that happen in the world. If we in addition would tell the hearer in an interaction the exact meaning of the utterance as constructed by the speaker, it would be as if the agents can look into each others heads, as if they were endowed with telepathic powers. Therefore, for each scene, we construct a set of possible meanings, possible parts of the scene that can serve as the topic in a communicative interaction. Topics are selected as follows:

1. All the $A(a)$'s and $A(p)$'s present in the scene can be the topic.
2. All of the $S(e, a, p)$'s present in the scene can also be the topic.

For example, if the scene would be

$$(E_3\ e\ a\ p)(F_4\ a)(F_1\ a)(P_1\ a)(F_4\ p)(P_7\ p),$$

then the set of possible topics is:

$$\begin{aligned}\{&(E_3\ e\ a\ p)(F_4\ a)(F_1\ a)(P_1\ a)(F_4\ p)(P_7\ p),\\ &(F_4\ a)(F_1\ a)(P_1\ a),\\ &(F_1\ a)(P_1\ a),\\ &(F_4\ a)(P_1\ a),\\ &(P_1\ a),\\ &(F_4\ p)(P_7\ p),\\ &(P_7\ p)\}\end{aligned}$$

For each interaction a new scene is generated and one of its topics is selected randomly to be verbalized by the speaker. The average number of topics contained in a scene is 8 and the average length of a topic, i.e. the average number of predicates contained in it, can easily be calculated to be $22/8 = 2.75$.

Note that the only topic descriptions that contain exactly one predicate are descriptions of the form $(P_i\ x), i = 1, .., n_p$. On average, the number of topic descriptions for a scene of length l is given by $T(l) = 2^{l-3} + 2(2^{l-2} - 1)/(l - 2)$, although about half of these might actually be the same.

Furthermore, a topic description is always connected in the sense that all predicates in it are linked through their arguments, either directly or else via other predicates they are linked to. For example, the expression $(P_1\ a)(P_3\ p)$ can never describe a topic because the two predicates are not linked. Adding an E predicate can solve this as in $(P_1\ a)(P_3\ p)(E_2\ e\ a\ p)$. This means that every topic description containing k predicates contains exactly $k - 1$ variable equalities as part of its meaning.

Finally, the number of topic descriptions that do not contain variable equalities is always 2, independent of the scene length (namely the two participant

predicates on their own.) Hence, the average ratio of the number of topic descriptions that contain equalities and the total number of topic descriptions can be calculated to be 66.6%.[3]

5.2 Lexical Constructions

The task of a speaker is to map the topic description onto some utterance. He may therefore propose new lexical and grammatical constructions if needed. Later on, we will explain how lexicon lookup is performed exactly and when it is decided to propose a new word etc. For now, we'll simply explain what type of lexical and grammatical constructions are allowed in the experiment and how they look like.

Every combination of predicates in the topic may be mapped onto a word. In other words: the meaning of a word can be any combination of predicates, possibly containing any number of equalities between the variables involved. Below follow three valid lexical constructions in FCG notation:

```
((?Top (meaning (== (P3 ?x))))
 ((J ?new ?top)
       (sem-cat ((participant ?x)))))
<-->
((?Top (form (== (string ?new "wibidakulala"))))
 ((J ?new ?top)
       (syn-cat ((NP ?x))))),                                    (1)

((?Top (meaning (== (F1 ?x) (P3 ?x))))
 ((J ?new ?top)
       (sem-cat ((participant ?x)))))
<-->
((?Top (form (== (string ?new "wibidakulala"))))
 ((J ?new ?top)
       (syn-cat ((NP ?x))))),                                    (2)
```

[3]One way to calculate this is by numerically determining the sum

$$\sum_{l=3}^{\infty} (\text{Prob}[\text{topic of length } l] \times \text{Prob}[\text{topic contains variable equalities} \mid \text{length } l])$$

$$= \sum_{l=3}^{\infty} (l-2) \left(\frac{1}{2}\right)^{(l-1)} \times \frac{T(l)-2}{T(l)},$$

which already converges after a few dozen terms.

```
((?Top (meaning (== (F1 ?x) (P3 ?y))))
 ((J ?new ?top)
       (sem-cat ((feature ?x)
                 (participant ?y)))))
<-->
((?Top (form (== (string ?new "obedo"))))
 ((J ?new ?top)
       (syn-cat ((Adjective ?x)
                 (NP ?y))))).                              (3)
```

All of these rules may be occur in an experiment. As you can see, they include semantic and syntactic category information. For example, the first rule, which maps the string "wibidakulala" to the meaning $(P_3\ ?x)$, specifies that $?x$ is of semantic category `participant` and that the syntactic category of "wibidakulala" is the list `((NP ?x))`. The second rule (2) maps the same string to a different meaning $(F1\ ?x)\ (P3\ ?x)$. It too specifies that $?x$ is a `participant` and that the string is of syntactic category `((NP ?x))`. Rules (1) and (2) are said to define homonyms because they map the same string to a different meaning. Finally, the third rule (3) maps the string "obedo" to almost but not quite the same meaning as specified in rule (2). The difference lies in the fact that rule (2) *requires* that the arguments to the predicates in its meaning are equal whereas rule (3) *forbids* them to be equal.[4] This is also reflected by the different semantic and syntactic category specified in rule (3). We'll now proceed with explaining why semantic and syntactic categories are needed and how they are defined.

5.2.1 Semantic Categories

Semantic and syntactic categories are needed to allow the definition of abstract grammatical constructions. For example, the semantic pole of a grammatical construction for linking (identifying) the variables of a feature and a participant predicate could look as follows:

```
((Top (sem-subunits (== ?feature-unit ?participant-unit)))
 (feature-unit
      (sem-cat (== (feature ?x))))
 (participant-unit
      (sem-cat (== (participant ?x))))).
```

[4] In most of the FCG work reported so far rule (3) would allow an equality between these arguments.

Instead of selecting for particular feature and participant predicates (i.e. one of the F_i and P_i's respectively), this pole selects for the more abstract semantic categories `feature` and `participant`. This makes the pole more general and applicable to all feature-participant predicate combinations instead of to only one.

Just as we have assumed that all agents share the notion of a universal set of predicates for events, event participants and participant features, we have:

Assumption 2. *All agents share a universal way of determining the semantic category of a conjunction of predicates.*

The semantic category of a single predicate is simply its type:

- `(event e a p)` for any of the $(E_i\ e\ a\ p)$ predicates,
- `(participant a)` for any of the $(P_i\ a)$ predicates and
- `(feature f)` for any of the $(F_i\ f)$ predicates.

Furthermore, the semantic category of a conjunction of predicates is determined as the smallest reduction according to the following rules:

- The category of a `(feature x)` $\wedge$ `(participant x)` combination is again `(participant x)`.
- The category of a `(participant a)` $\wedge$ `(participant p)` $\wedge$ `(event e a p)` combination is `(scene e a p)`.

Below is given a table with example descriptions and their semantic categories.

Description	Type
$(F_i\ ?x)$	`(feature ?x)`
$(P_i\ ?x)$	`(participant ?x)`
$(F_i\ ?x) \wedge (F_j\ ?x) \wedge (P_k ?x)$	`(participant ?x)`
$(F_i\ ?x) \wedge (F_j\ ?y) \wedge (P_k ?y)$	`(feature ?x)` $\wedge$ `(participant ?y)`
$(F_i\ ?x) \wedge (F_j\ ?y) \wedge (P_k ?z)$	`(feature ?x)` $\wedge$ `(feature ?y)` $\wedge$ `(participant ?z)`
$(F_i\ ?x) \wedge (P_j\ ?x) \wedge (E_k\ ?e\ ?x\ ?y)$	`(participant ?x)` $\wedge$ `(event ?e ?x ?y)`
$(F_i\ ?x) \wedge (P_j\ ?x) \wedge (E_k\ ?e\ ?x\ ?y) \wedge (P_l\ ?y)$	`(scene ?e ?x ?y)`

5.2.2 Syntactic Categories

Assumption 3. *Syntactic categories are in a one to one correspondence with semantic categories according to the following mapping:*

1. *A* `(feature ?x)` *semantic category maps to a* `(Adjective ?x)` *syntactic category.*
2. *A* `(participant ?x)` *semantic category maps to a* `(NP ?x)` *syntactic category.*
3. *A* `(event ?e ?a ?p)` *semantic category maps to a* `(Verb ?e ?a ?p)` *syntactic category.*
4. *A* `(scene ?e ?a ?p)` *semantic category maps to a* `(S ?e ?a ?p)` *syntactic category.*

The careful reader has noticed that the production rules for scenes or, equivalently, the reduction rules for types actually implement a kind of generative grammar which, in terms of syntactic categories, looks like:

$$\begin{aligned} (\text{S } e\ a\ p) &\rightarrow (\text{NP } a)(\text{NP } p)(\text{Verb } e\ a\ p) \\ (\text{NP } a) &\rightarrow (\text{Adjective } a)(\text{NP } a) \\ (\text{NP } a) &\rightarrow (\text{Noun } a), \end{aligned}$$

except that, unlike in normal generative grammars, the order of constituents is not fixed by the type system (e.g. (NP a) just as well rewrites to (NP a)(Adjective a) as it does to (Adjective a)(NP a).)

Although techniques exist by which the agents could deduce this structure from the scene and type data, they *do not* have any such capacity.

5.2.3 Applying a number of Lexical Constructions

The result of applying a set of lexical constructions to a topic description is a unit structure containing one top-unit and a number of subunits, just as it was the case in chapters 3 and 4.

Depending on the set of lexical constructions selected for application, the top unit may still contain a number of uncovered meaning predicates. All subunits are the result of applying lexical constructions similar to (1) to (3) above and hence do not contain any subunits themselves. They do contain a `meaning` and `sem-cat` feature on the semantic side, as well as `form` and `syn-cat` features on the syntactic side.

Hence, after application of a set of lexical constructions in production, the semantic and syntactic structures will look like:

```
((Top (sem-subunits (S1 ... Sn))
      (meaning U))
 (S1  (meaning M1)
      (sem-cat SemCat1))
 ...
 (Sn  (meaning M2)
      (sem-cat SemCatn)))
```

and

```
((Top (syn-subunits (S1 ... Sn)))
 (S1  (form (string S1 F1))
      (syn-cat SynCat1))
 ...
 (Sn  (form (string Sn Fn))
      (syn-cat SynCatn))),
```

where `U` is a (possibly empty) list of uncovered predicates, `M1` to `Mn` are (nonempty) lists of predicates with arguments that are also present in the associated semantic and syntactic categories `SemCat1` to `SemCatn` and `SynCat1` to `SynCatn`.

How an agent decides which set of available lexical constructions to apply will be fleshed out later. But we already state the following:

Assumption 4. *When, after application of the lexicon, there are still uncovered meaning predicates left in the semantic top unit (i.e. the predicates present in* `U` *above), then one new lexical entry is proposed by the speaker covering all of them.*

For the example, and assuming that `U` is a nonempty list `(U1 .. Un)`, and that the semantic and syntactic categories of this conjunction (determined as explained) are `Sem` and `Syn`, then the proposed lexical construction would look like:

```
((?Top (meaning (== U1 ... Un)))
 ((J ?new ?top)
       (sem-cat Sem)))
<-->
((?Top (form (== (string ?new new-string))))
 ((J ?new ?top)
       (syn-cat Syn))).                                    (4)
```

The `new-string` is a randomly generated but unique new string of three syllables, like “wabudu”. This lexical entry is then also applied.

To conclude then, after lexicon lookup and after possibly inventing a new word, all predicates in the meaning are covered and the meaning feature of the top unit semantic structure is empty. However, the verbalization is not finished yet, because the structure might still contain equalities between variables in different units. In fact, it is easy to see that whenever there is more than one unit (apart from the top unit) there will be equalities. This is because the topic description is always completely connected as explained.

Assumption 5. *All equalities present in the topic description (i.e. all linking relations contained in it) need to be resolved (expressed) before the syntactic structure is rendered into an utterance. An equality can be resolved either because it is part of the meaning of one (holistic) lexical construction or else because it is covered by a grammatical construction.*

We have already seen an example of a lexical construction that resolves a variable equality: rule (2) above which maps the meaning $(F_1\ ?x)(P_3\ ?x)$ to the string "wibidakulala". However, if instead two simpler lexical constructions would have been used to cover the meaning, one mapping the predicate $(F_1\ ?x)$ to a string s_1, and another mapping the predicate $(P_3\ ?x)$ to a string s_2, then the semantic structure would look like:

```
((Top (sem-subunits (Unit1 Unit2)))
 (Unit1 (meaning ((F1 X)))
        (sem-cat ((feature X))))
 (Unit2 (meaning ((P3 X)))
        (sem-cat ((participant X))))).
```

Both units `Unit1` and `Unit2` refer to the same entity `X`. The syntactic structure would be:

```
((Top (syn-subunits (Unit1 Unit2)))
 (Unit1 (form ((string Unit1 s1)))
        (syn-cat ((Adjective X))))
 (Unit2 (form ((string Unit2 s2)))
        (syn-cat ((NP X))))).                                    (5)
```

Hence, nothing in the form reflects the equality in the meaning: the only thing that the syntactic structure specifies about the final form (the utterance) is that it should include two string values s_1 and s_2. No information is encoded about e.g. the order of these two strings. Hence, both utterances "$s_1 s_2$" and "$s_2 s_1$" are still consistent with it, and a grammatical construction is required resolve the equality between both units by imposing an order on them.

5.3 Grammatical Constructions

A construction that covers the variable equality left uncovered at the end of the previous section looks as follows:

```
((?Top   (sem-subunits (== ?unit1 ?unit2)))
 (?unit1 (sem-cat (== (feature ?x))))
 (?unit2 (sem-cat (== (participant ?x))))
 ((J ?new ?Top)
         (sem-cat ((participant ?x)))))
<-->
((?Top   (syn-subunits (== ?unit1 ?unit2))
         (form (== (meets ?unit1 ?unit2))))
 (?unit1 (syn-cat (== (Adjective ?x))))
 (?unit2 (syn-cat (== (NP ?x))))
 ((J ?new ?Top)
         (syn-cat ((NP ?x))))).
```

The fact that the new unit is of semantic category `participant` and of syntactic category `NP` can be deduced by the agents using the rules for determining semantic and syntactic categories as explained.

On the syntactic side, this construction imposes an order on its subunits. When an agent has to invent a new construction, an order is chosen randomly amongst all possible orderings. For example, in this case, since there are only two subunits involved (`Unit1` and `Unit2`), there is one other possible ordering putting `Unit2` directly before `Unit1`.

Applying the above construction to the structures in (5) results in:

```
((Top   (sem-subunits (New-Unit)))
 (New-Unit
        (sem-subunits (Unit1 Unit2))
        (sem-cat ((participant X))))
 (Unit1 (meaning ((F1 X)))
        (sem-cat ((feature X))))
 (Unit2 (meaning ((P3 X)))
        (sem-cat ((participant X)))))
```

and

```
((Top   (syn-subunits (New-Unit)))
 (New-Unit
        (syn-subunits (Unit1 Unit2))
        (form ((meets Unit1 Unit2)))
```

```
       (syn-cat ((NP X))))
(Unit1 (form ((string Unit1 s1)))
       (syn-cat ((Adjective X))))
(Unit2 (form ((string Unit2 s2)))
       (syn-cat ((NP X))))).                                    (6)
```

In sum, when a speaker has to verbalize a certain meaning, he will:

1. Decide which combination of known lexical constructions to apply. How this is done is explained later.

2. If needed, propose a new lexical construction for the entire set of predicates still left uncovered (i.e. still in the meaning feature of the top unit.)

3. Decide which set of known grammatical constructions to apply. Again this will be explained later.

4. If needed, propose a new grammatical construction covering all of the equalities still left uncovered.

The result is a semantic and syntactic structure of which the top units contain one subunit only. All and only the leaf units in the syntactic structure contain string information. All units in the syntactic structure are totally ordered by the form constraints present in all but the leaf units.

5.4 Utterances

A speaker always succeeds in constructing a syntactic structure completely specifying the utterance as explained at the end of the previous section. However, the hearer does not have access to this structure, or else he would have telepathic capacities. The utterance as specified in the speaker's syntactic structure first has to be rendered into an utterance accessible to the hearer. Therefore, the next assumption we will make is:

Assumption 6. *Only the concatenation of all string values, ordered as specified in the speaker's syntactic structure after verbalization, is accessible to the hearer.*

For example, the rendering of structure (6) is the concatenation of the strings s_1 and s_2. IF s_1 is "wabadu" and s_2 is "kilara", then the result is the string "wabadukilara". In other words, we do not assume that agents are aware of word boundaries, and for all the hearer knows, "wabadukilara" could be one

to 12 words (the number of characters, but not more.) In turn, the hearer will have to use his lexicon to figure out which words actually are present in the utterance. How this is done will be explained later.

5.5 Interactions

As explained, every time step, one speaker and one hearer are randomly selected from the population. They both are presented with a context which is a conjunction of predicates. Only the speaker is given a topic description, which is a fully connected subset of predicates from the context description.

Verbalization results in a fully connected string which is presented to the hearer. The hearer has to parse this string into a conjunction of predicates again called the parsed topic description. If all goes well, this description should be the same as the topic description given to the speaker (up to a renaming of variables), including all variable equalities. Many things could go wrong however. For example, the hearer might not know all the words used by the speaker. Or he might prefer a different word order. Another possibility is that he associates a different meaning to some (sub)string in the utterance. Or even that he splits the utterance differently than the speaker (remember that hearer agents do not get to see the word boundaries in the utterance as intended by the speaker.)

All of these problems might lead to different interaction results. A threefold distinction of possible scenario's is made: a successful parse, an incomplete parse or an erroneous parse.

5.5.1 Successful Parse

A successful parse is one in which all words are (assumed to be) known and in which the hearer's syntactic top unit again contains only one subunit. This means that, in the case of a (hypothesized) multiple-word utterance, a set of grammatical constructions could be found that resolves all (hypothesized) variable equalities as (presumed to be) intended by the speaker.

A successful parse does not necessarily lead to a successful interaction. For this, we also need a unique and successful interpretation. The interpretation of a parsed semantic structure is the unification of the predicates present in the structure with the given context (up to maybe a different order.) For example, if $\{p_1, ..., p_n\}$ is the set of predicates in the parsed semantic structure, and if C is the context (a list of predicates), then the set of interpretations is defined as the unification of $(== p_1 \ ... \ p_n)$ with C. This leads to a set of possible bindings lists for the variables present in the parsed topic description (in the predicates

p_1 to p_n.) If this set is empty then the interaction is a failure. If there is at least one interpretation (set of bindings) then the interaction is considered a success.

Note that having a successful interaction doesn't mean that the hearer's parsed meaning is the same as the speaker's, only that both meanings are compatible with the given context.

5.5.2 Incomplete Parse

There are two ways in which a parse can be incomplete: either because one ore more words in the utterance are unknown to the hearer (or better: one ore more *hypothesized* words because the hearer doesn't see the intended word boundaries), or else because after application of the grammatical constructions the top unit still contains more than one subunits, meaning that some variables in them still should be identified.

In any case, the hearer will only proceed if there is at most one unknown word and if the set of predicates parsed so far has at least one interpretation in the given context (i.e. if what is parsed so far already makes sense.)

If one of the words is unknown to the hearer, he will try to figure out the meaning of it and, if successful, signal the speaker that he learned something. This also ends the interaction. Hence, an interaction can end in three different ways: in a success, in a failure or in a situation where something is learned. In the following we will also label these cases 1 (for success), 0 (for learned) and -1 (for failure.)

If no words are unknown, the hearer tries to use the context to figure out which variables in the top unit's subunits still have to be identified. He then adopts a grammatical construction that would perform this on the semantic side while adopting the word order as specified by the speaker's utterance on the syntactic side. Again, but only in case the adopted construction still leads to a successful interpretation, the hearer signals that he has learned something. If anything goes wrong, for example if the hearer can't figure out the meaning of an unknown word, or if the adopted construction doesn't make sense, the interaction is a failure.

Let us examine these two possibilities in some more detail by example. Assume that both the context and the topic description are equal to the conjunction of two predicates $C = T = (p_1\ p_2)$. Assume that the speaker uses two words w_1 and w_2 for describing the topic description, w_1 meaning p_1 and w_2 meaning p_2. Assume further that the hearer knows word w_1 but associates the entire meaning $(p_1\ p_2)$ with it. This meaning does make sense in the context, but it is impossible for the hearer to figure out the meaning of word w_2 because there is no meaning left in the context and hence the interaction is a failure. In section 6 we will see however that the hearer is still able to learn something.

As an example of how an interaction can still fail even after the hearer adopts a new grammatical construction, assume that the utterance is a concatenation of three strings "$s_1 s_2 s_3$", with parsed meanings respectively $(f_1\ ?a)$, $(f_2\ ?b)$ and $(p_3\ ?c)$. Assume that interpreting this meaning in the context results in an unresolved equality between variables $?a$ and $?c$. This cannot be solved by a grammatical construction however because the allowed constructions require that the strings involved in the equality are put directly next to each other (because of the use of `meets` form constraints, i.e. in this case the string s_1 should be put directly next to the string s_3 but this is incompatible with the utterance.)

Note that both in the case of a successful and of an incomplete parse, there could be *several* interpretations. If the context contains the predicates $((f_1\ X)(p_1\ X)\ (f_1\ Y))$, and if the parsed topic description includes $((f_1\ ?a)(p_1\ ?b))$, then there are at least two interpretations:[5]

$$[?a/X,\ ?b/X]$$

and

$$[?a/Y,\ ?b/X].$$

If all else goes well, the first interpretation will lead to an incomplete parse in which, if possible, a new construction has to be adopted. The second interpretation will lead to a successful interpretation. At any time, if several interpretations are found, one is selected randomly.

5.5.3 Erroneous Parse (complete or incomplete)

Finally, if the (possibly partial) parse does not have any interpretations then the interaction fails. We'll see in section 6 that both the hearer and the speaker still learn in this case.

Although not everything in was explained yet, figure 5.1 summarizes the possible scenario's.

[5]See section 3.3.2 and Appendix A for the definition of this notation. In short, symbols starting with a question mark are variables. These can be bound to (or substituted by) a value. If $?v$ is a variable and V is a value, then $[?v/V]$ specifies V as the value to which $?v$ is bound or with which $?v$ should be substituted. Hence, an interpretation of a set of predicates is a way to give values to the arguments (variables) of the predicates in a way that is in accordance with the current context.

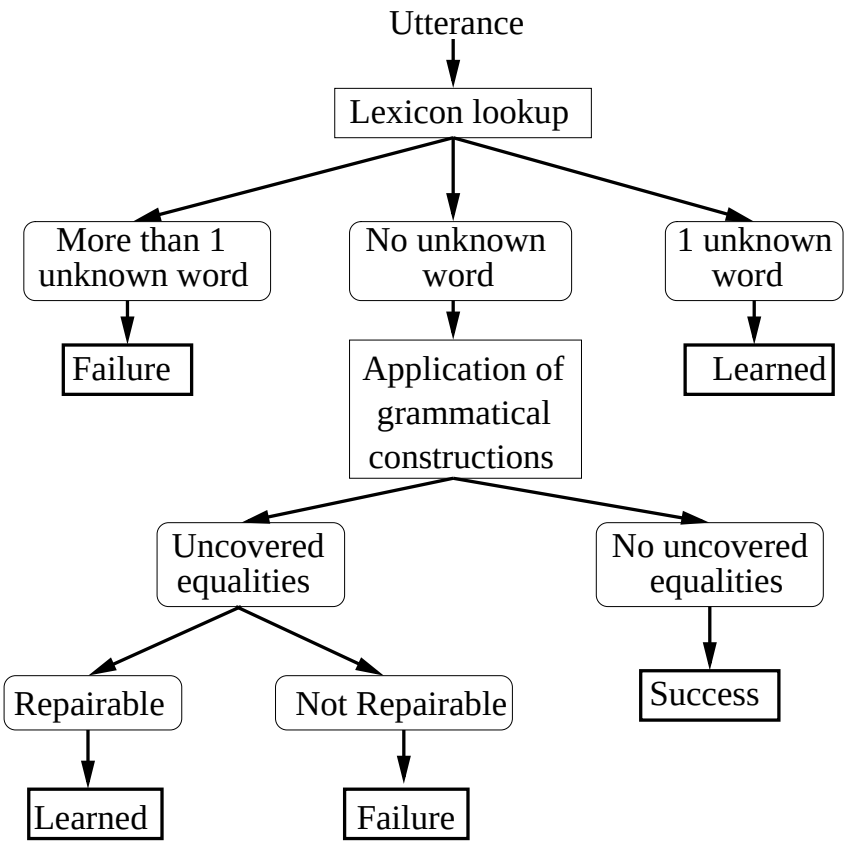


Figure 5.1: Flow chart of a hearer agent showing the possible ending scenario's of an interaction.

5.6 Overview of Solutions

In this chapter we have set the problem faced by the agents. The set of scenes they have to describe was defined, the way in which topics are chosen from a scene, the type system (how semantic and syntactic categories are determined), what lexical and grammatical constructions are allowed and how an utterance is rendered into a form observable by the hearer. We have also specified what other nonlinguistic information can be passed between the agents: the hearer may signal a success, a failure, or that he has learned something. Every interaction ends in one of these three signals and it is well defined what the meaning of these signals is.

The challenge for the agents is now to evolve a communication system, from scratch, and such that all possible interactions (i.e. all interactions given a predefined set of scenes and hence topics) between any two agents out of the population always end successfully. They may not share any other information besides specified: only the context description, the knowledge about the possible topics in the context, the rendered utterance and the final success signal.

We'll conclude this chapter by discussing a number of solutions to the problem and some of the difficulties that come with them.

5.6.1 Purely Holistic Encoding

In a purely holistic encoding, agents do not need grammatical constructions at all. Instead, a different word is used for every possible topic description. So the word associated with the meaning $((p_1\ X))$ is not used when the meaning to express is $((f_1\ X)\ (p_1\ X))$. Instead, another word that covers the entire meaning is used. In turn, this word is not re-used for the meaning $((f_1\ X)\ (p_1\ X)\ (e_1\ E\ X\ Y)\ (p_2\ Y)$ and so on.

If agents would agree in advance to use this strategy, things may appear to be simple. For example, agents do not have to care anymore about how to split an utterance into several words. On the other hand, agents now have to reach an agreement about a huge number of words. As was calculated in section 5.1.1, there are in theory an infinite number of distinct topic descriptions. On average however, the length of a scene is 5. The probability of generating a scene of length l ($l \geq 3$) was calculated to be $(l-2)(1/2)^{l-1}$. Hence the probability of generating a scene greater than length l is given by

$$\sum_{i=l+1}^{\infty} (i-2)(1/2)^{i-1} = 1 - \sum_{i=3}^{l} (i-2)(1/2)^{i-1}.$$

This is about 0.1% for $l = 14$, so we can safely restrict the rest of the discussion to scenes of length 14 or less.

The number of scenes of length l is $n_e n_p^2 n_f^{l-3}$. The average number of topic descriptions for a scene of length l was calculated to be $2^{l-3}+2(2^{l-2}-1)/(l-2)$. Hence, the total number of topic descriptions assuming that scenes of length greater than 14 never occur can be approximated to be

$$\sum_{i=3}^{14} n_e n_p^2 n_f^{l-3} \left(2^{l-3} + 2(2^{l-2} - 1)/(l-2)\right).$$

For $n_e = n_p = n_f = 5$, this is of the order 10^{13} topics, of which some are the same.

From recent theoretical and experimental results concerning the naming game we know that the time (the number of interactions) for a population to converge on a purely lexical language is directly proportional to the number of distinct topics. This is under the assumption that *agents are able to determine the intended meaning of an unknown word without mistake* (see Baronchelli et al. (2006a); De Vylder and Tuyls (2006) and also Chapter 2, section 2.3.1.) Under the assumptions we made, this is *not* the case. A hearer will adopt a new word if there is (presumably) exactly one unknown word in the utterance. Even if we neglect the fact that utterances might consist of multiple words and that the meaning of the other, presumed known words is guessed correctly, in many cases he still will be unable to determine the meaning of the unknown word. A scene description on average contains 5 predicates. A topic description only 2.75. We can therefore estimate the number of possible meanings for an unknown word as $C_5^1 + C_5^2 + C_5^3 = 25$. In practice, this number sometimes is still a lot bigger. In other words, there is a major source for homonymy. This greatly complicates the problem and significantly slows down the convergence process as was also investigated in Chapter 2, section 2.3.2.

In sum, although a purely lexical and holistic solution exists, it is far from the optimal solution both in terms of memory requirements and of the time needed to find the solution.

5.6.2 (Partially) Compositional Lexicon, Holistic Grammar

From the previous it should be clear that it is desirable to have some form of compositionality. If every predicate in a topic description maps to a different and unique word, the number of words required is $n_e + n_p + n_f$. Again, for $n_e = n_p = n_f = 5$, this amounts to 15, which is only a fraction of the number of words required in the holistic solution (of the order 10^{13}.)

On the other hand, according to assumption 5, all equalities always have to be expressed, either by being part of the meaning of a holistic word, or else

by using a grammatical construction that imposes an order among the words involved. Only the latter solution is possible here. Hence, the fact that agents also need to agree on the different orderings enlarges the search space again somewhat.

If we limit ourselves again to descriptions of length smaller than 15, and if we assume that every other combination of words of different syntactic categories is covered by a different construction (i.e. instead of for example only one recursive construction `NP -> Adj Np`, several constructions of the form `NP-> Adj NP`, `NP-> Adj Adj NP` or `NP-> Adj NP Adj` or ... and so on, are needed) then the number of possible orderings is of the order a few thousands of which some dozens need to be selected and agreed upon by the population.

What complicates things is that the agents can only successfully start to negotiate about different orderings after they more or less agree upon a basic lexicon. Indeed, if a hearer agent doesn't even know which words are contained in an utterance, let alone their meaning and semantic and syntactic categories, then he will certainly not be able to guess the grammatical constructions that were used.

Another thing that needs to be taken care of is that there has to be some mechanism that introduces compositionality in the language. As was stated in assumption 4, a speaker will associate one word with whatever parts of the meaning are still uncovered by his lexicon. Since agents start of with empty lexicons, most of the words introduced in the beginning of a simulation will be holistic. The only exception to this is when the topic contains one predicate only. Because of the way topics are calculated, this necessarily has to be a participant predicate. Only after these are covered by atomic words (a word having only one predicate as its meaning) can a speaker agent consider to introduce atomic words for the other type of predicates, and only when the topic requires it (e.g. when it consists of one participant plus one feature predicate.)

Note however that it is not because a speaker uses an atomic word that the hearer will adopt it as such, not even if the topic consists of a single participant predicate. This is because the hearer does not know the intended topic. So if he hears an unknown word, then he also needs to consider the other topic descriptions consistent with the context scene as possible meanings for it. It is clear that this makes reaching agreement about a compositional language a far from trivial task. In addition, our agents do not look for re-occurring patterns in their lexicons to introduce compositionality. For example, as was explained in the introduction, a typical agent in an iterated learning experiment would conclude that a string "a" has meaning m_a based on the two example utterances "ab" and "ac" with meanings respectively $m_a \wedge m_b$ and $m_a \wedge m_c$. Our agents do not have any such capacity.

Fortunately, there is an additional source of compositionality. To see this,

consider that a hearer agent hears the same word used on several occasions. As will be explained in the next chapter, he will use cross situational learning to figure out its meaning. Assume for example that a first situation scene is described by $(P_1\ ?a) \wedge (V_1\ ?e?a?p) \wedge (P_2?p)$. The topics contained in it are $(P_1\ ?a)$, $(P_2\ ?e)$ and the scene itself. Hence, a hearer agent presented with an unknown word in this case will associate all of the three meanings with it, each with a probability of one third. A second scene could be $(P_1\ ?a) \wedge (V_2\ ?e?a?p) \wedge (P_3?p)$, with topics now $(P_1\ ?a)$, $(P_3\ ?e)$ and the scene itself. Only $(P_1\ ?e)$ is consistent with the previous scene. So if cross-situational learning is performed on these two situations then the meaning of the unknown word will converge towards it, *even if this was not intended by the speaker.* As a second example, assume that an unknown word is observed in combination with a known word meaning $(P_1\ ?a)$. Assume further that the scene is described by $(P_1\ ?a)(A_1\ ?a) \wedge (V_2\ ?e?a?p) \wedge (P_3?p)$. Now the hearer can, knowing the set of possible topics contained in the scene, and hypothesizing that the intended topic must contain the predicate $(P_1\ ?a)$, reduce the set of possible meanings for the unknown word to one of $(A_1\ ?a) \wedge (V_2\ ?e?a?p) \wedge (P_3?p)$ and $(A_1\ ?a)$. Again, one of these would make the unknown word atomic, and in a next situation it might even converge towards it.

Put more general, using cross-situational learning for learning the meaning of words has as additional effect that any statistics present in the world will get absorbed by and reflected in the agents' lexicons. This will be illustrated more clearly in the baseline experiment in the next chapter, where a certain structure will be imposed upon the world according to a correlation-parameter. For example, if this parameter is close to one, then the probability that certain predicates occur together is increased. This in turn will be reflected in the degree of compositionality of the languages to which the agents converge. If, as a boundary case, two predicates *always* occur together, then they will never be covered by separate atomic words in the resulting language. Hence, the claim that atomic words can be used more frequently and therefore are agreed upon faster and become more efficient for communication compared to holistic words is only true to the extend that this is compatible with the statistics of the world. More correctly stated, those elements of meaning, either atomic or not, that occur more frequently as part of the topic in a communicative interaction, will also have a greater chance of becoming an elementary unit of meaning in language. This is, to my view, a desired and direct consequence of adopting a usage based view on language.

To summarize, and turning back to the original problem statement, aiming for the compositional solution might significantly reduce the search space, although it is not a-priori clear how this will be influenced by the interaction with

the search for an appropriate set of grammatical constructions.[6] Importantly, agents may not decide in advance to go compositional because then they would miss the opportunity to also use holistic constructs for frequently occurring meanings.

5.6.3 Minimal Recursive Language

A next improvement in terms of number of constructions needed is when not only the lexicon, but also the grammar becomes compositional. Better put, the only two grammatical constructions required are (variations of) the two below:

```
((?Top (sem-subunits (== ?A-unit ?V-unit ?P-unit)))
 (?A-unit (sem-cat ((participant ?a))))
 (?V-unit (sem-cat ((event ?e ?a ?p))))
 (?P-unit (sem-cat ((participant ?p))))
 ((J ?new ?Top)
          (sem-cat ((scene ?e ?a ?p)))))
<-->
((?Top (syn-subunits (== ?A-unit ?V-unit ?P-unit))
       (form (== (meets ?A-unit ?V-unit)
                 (meets ?V-unit ?P-unit))))
 (?A-unit (syn-cat ((NP ?a))))
 (?V-unit (syn-cat ((Verb ?e ?a ?p))))
 (?P-unit (syn-cat ((NP ?p))))
 ((J ?new ?Top)
          (syn-cat ((S ?e ?a ?p))))),
```

and

```
((?Top (sem-subunits (== ?A-unit ?NP-unit)))
 (?A-unit  (sem-cat ((feature ?x))))
 (?NP-unit (sem-cat ((participant ?x))))
 ((J ?new ?Top)
          (sem-cat ((participant ?x)))))
<-->
((?Top (syn-subunits (== ?A-unit ?NP-unit))
       (form (== (meets ?A-unit ?NP-unit))))
 (?A-unit  (syn-cat ((Adjective ?x))))
```

[6] A *bootstrapping procedure* might be considered here in which language learners first learn simple (lexical) elements of language and only after that start learning the more complicated (grammatical) parts. This is also what happens in infant directed speech and what was investigated briefly for example in (De Beule and Bergen, 2006).

```
(?NP-unit (syn-cat ((NP ?x))))
((J ?new ?Top)
         (syn-cat ((NP ?x))))).
```

The 'variations of' means that other orderings (other meets constraints) are also OK.

The reason for the huge reduction in the number of grammatical constructions needed (from a few dozen to only two) lies of course in the recursiveness of the last construction. By itself, it can replace all constructions of resulting type NP that select for more than 2 constituents (like `NP -> Adjective Adjective NP`.) And together with the first construction, also all S-type constructions that select for more than three constituents become obsolete (e.g. `S -> NP Adjective V NP` etc.)

This solution is the one we would like our agents to evolve but without biasing them towards it. The only thing available to the agents for deciding what set of lexical and grammatical constructions to use or propose is the expected communicative success.

In the following chapters we'll see how this can be done. Although the agents will go through a phase in which the emerging language is a mixture of all three solutions discussed, in the end they invariably develop a minimal recursive language, simply because it contains less rules (constructions) and hence is easier to conventionalize and become successful.

5.6.4 Redundant Languages

To be complete, we shortly mention that other solutions also exist. For example, instead of evolving a single word for every single predicate as is the case in the compositional and minimal solutions, agents could remember *several* words for each predicate. This solution is also known from the naming game when lateral inhibition is turned off (although things are a bit more complex in our case because of homonymy.)

Chapter 6

Agent Architecture

In the previous chapter the problem statement was formulated with which the agents in our simulation experiment will be faced. In this chapter, the details about the agents will be fleshed out. First, it will be explained how a speaker agent uses and extends his language inventory to produce an utterance. This can be divided into two parts: applying and extending the lexicon and applying and extending the grammar. Next it will be explained how a hearer uses and extends his language inventory to parse an utterance. This too will consist of two parts. Finally the learning mechanisms will be explained.

6.1 Applying the Lexicon for Production

After some interactions, the lexicon will contain a big number of lexical constructions. As explained in chapter 2, section 2.3.2, every lexical construction has a probability score associated with it. It represents the estimated probability that the other agents in the population associate the same meaning with the word. It also determines how the agent itself would interpret the word. The strategy for deciding which words to use when some meaning has to be verbalized is based on the fact that an agent may only use words that he himself would interpret as that meaning.[1]

So assume that a speaker agent has to verbalize the set of predicates $P = \{p_1, ..., p_n\}$. He first goes through his lexicon to determine the set of entries that interpret as a subset of P. For example, some entry e_1 might cover the entire set P. Another one, e_2, may cover only the predicate p_1 and so forth. From this set, the speaker constructs a set of subsets of lexical entries $\{S_1, ..., S_k\}$, such that for every set of entries S_i $(i = 1, ..., k)$, every predicate in P is covered at

[1]The idea to also let the speaker determine the meaning of a word as if he were himself a hearer resembles the *introspective obverter* mechanism of Smith (2003a).

most once. For example, e_1 and e_2 cannot occur in the same set because then the predicate p_1 would be covered twice.[2] On the other hand, we also consider those sets that leave some of the predicates uncovered. For example, one of the sets S_i is equal to $\{e_1\}$, even if this leaves all of the predicates p_2 to p_n uncovered. As an extreme example, also the empty set $\{\}$ is considered.

A specific subset S_i is called a (candidate) coverage for the set of predicates P. Now we'll explain how the score of a coverage is determined. The idea is that the speaker should try to maximize the portion of the topic description P that is conveyed successfully. If few of the predicates in P are covered, then not much will be conveyed. However, if most of the predicates in P are covered by low probability entries, then not much will be conveyed either, because the hearer will most probably misunderstand.

Evidently, the best would be to have every predicate covered with a maximum probability of 1.0, and the worst if nothing is covered or, equivalently, everything with a probability of zero. So one way to score a coverage would be to determine the average over all predicates in P of the probability with which the predicate is covered, taking a probability of zero for uncovered predicates. However, this would treat synonyms with equal probability equally, even if they have different synonymy scores. Therefore, each probability is modulated with (multiplied by) the appropriate synonymy score:

Definition 1. The score of a coverage S for a topic description P is equal to the average over all predicates $p \in P$ of the product of the probability and synonymy score of the lexical entry in S covering p.

Assumption 7. *To produce an utterance, a speaker selects that coverage with the highest score. If several coverages exist with equal scores then one is selected randomly.*

Note that more complete coverages are not a-priori preferred. Indeed, assume that the topic description P contains 2 predicates p_1 and p_2. Assume furthermore that the speaker's lexicon contains an entry e_1 covering both p_1 and p_2 with score s_1 and an entry e_2 covering only p_1 with score s_2. The scores intended here are already the products of the probability and synonymy scores. The set of coverages contains the empty set $\{\}$, the set $\{e_1\}$ and the set $\{e_2\}$. Their scores are respectively 0, $s_1/2$ and $s_2/2$. If $s_2 > s_1$ then the coverage $\{e_2\}$ will be selected, although, in contrast with the coverage $\{e_1\}$, it does not cover all predicates in P.

On the other hand, note that this scoring mechanism favors strong holistic entries because all of the predicates covered by it benefit from the high score.

[2] Unless the predicate p_1 occurs twice in P.

In sum, when a speaker has to verbalize a set of predicates P, he first performs a lexicon lookup in which he determines the set of lexical entries, called a coverage, that has the highest coverage score. The result of applying the entries in the winning coverage is a semantic and syntactic structure containing subunits for all entries applied and, on the semantic side, a top unit possibly still containing a set of uncovered predicates in its meaning feature. As already explained (see assumption 4), the speaker proposes a new lexical entry covering all uncovered predicates. Applying this additional entry results in an additional subunit and an empty top unit's meaning feature. The next step consists of applying the constructicon.

6.2 Applying the Grammar for Production

According to assumption 5), all equalities still present in the semantic structure after lexicon lookup need to be resolved before the syntactic structure is rendered into an utterance.

The first step consists of searching for a combination of already acquired constructions that can be applied. We haven't yet explained how a set of grammatical constructions should be applied. In the case of the lexicon, things were relatively simple: the set of candidate coverages is determined and that with the highest score is selected. Then, all lexical constructions in it are applied one after the other and in a random order.

In the case of the constructicon however, things are more complicated because on the one hand we haven't yet explained how constructional entries are scored and second because the recurrent application of grammatical constructions brings a problem with it that doesn't occur with lexical entries. To see the problem, consider the following semantic structure:

```
((Top (sem-subunits (F1-unit F2-unit P-unit)))
 (F1-unit (sem-cat ((feature X)))
          (meaning ((F1 X))))
 (F2-unit (sem-cat ((feature X)))
          (meaning ((F2 X))))
 (P-unit  (sem-cat ((participant X)))
          (meaning ((P1 X))))).                          (1)
```

There are equalities between all subunits in the structure. One possibility would be to apply a 'holistic construction' that takes all units together at once in a new unit, imposing a word order on them.

Another possibility would be to apply the following recursive construction twice:

```
((?Top (sem-subunits (== ?F-unit ?P-unit)))
 (?F-unit (sem-cat (== (feature ?x))))
 (?P-unit (sem-cat (== (participant ?x))))
 ((J ?new ?Top)
          (sem-cat (== (participant ?x)))))
<-->
((?Top (syn-subunits (== ?F-unit ?P-unit))
       (form (== (meets ?F-unit ?P-unit))))
 (?F-unit (syn-cat (== (Adjective ?x))))
 (?P-unit (syn-cat (== (NP ?x))))
 ((J ?new ?Top)
          (syn-cat (== (NP ?x))))).                    (2)
```

If this construction is applied once to the structure in (1), the result looks like:[3]

```
((Top (sem-subunits (F1-unit NEW)))
 (F1-unit (sem-cat ((feature X)))
          (meaning ((F1 X))))
 (NEW (sum-subunits (F2-unit P-unit))
      (sem-cat ((feature X))))
 (F2-unit (sem-cat ((feature X)))
          (meaning ((F2 X))))
 (P-unit  (sem-cat ((participant X)))
          (meaning ((P1 X))))).                        (3)
```

Now construction (2) indeed applies again to this structure, but *twice*: once taking together the `NEW` and `F1` units as intended, but also again taking together the `F2` and `P` units with the `?Top` variable bound to `NEW`. The problem is illustrated in Figure 6.1. To solve it, we state the following:

Definition 2. The *hat* of a unit structure consists of the top unit and its direct subunits only.

For example, the hat of structure (3) consists of the units `TOP, F1-unit` and `NEW` (see also Figure 6.1 where the hat of the structures always appears above the dashed lines.) Hence, since a grammatical structure pushes its constituents one level deeper into the hierarchy, it actually *pushes them out of the hat*. Therefore we state:

[3]This is one of two possibilities, the other would take the units `F1-unit` and `P-unit` together

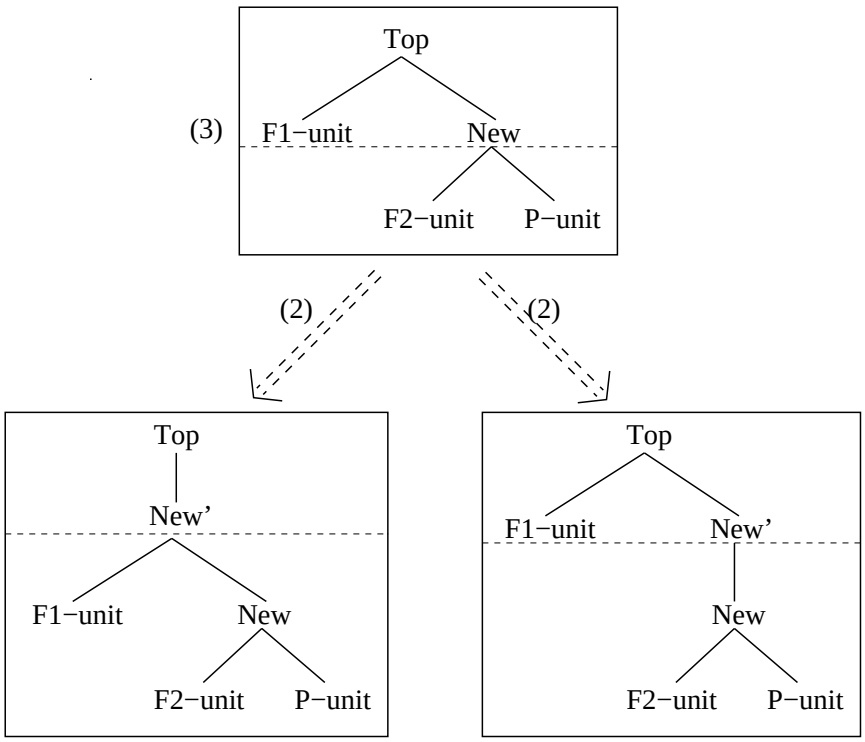


Figure 6.1: Illustration of the problem described in section 6.2 concerning the repeated application of constructions. The above tree corresponds to the structure in (3), which was obtained by applying construction (2) once to the structure shown in (1). Now construction (2) again applies, but *twice*, leading on the one hand to the desired result (on the left) but on the other also to an undesired one (right.) The hat of each structure appears above the horizontal dashed lines within the structure boxes.

Assumption 8. *The unification phase of rule application is performed on the hat of the current semantic (production) or syntactic (parsing) structure only.*

This excludes the possibility to apply construction (2) to structure (3) twice: only the intended application remains possible.

Let us now proceed with explaining the scoring mechanism. A grammatical construction also has two associated scores. The semantic pole of a grammatical construction selects for a specific combination of semantic categories involved in one ore more variable equalities. For example, the construction in (2) selects for two units of semantic category `((feature ?x))` and `((participant ?x))`. The order in which these units appear is of no importance, but the variable equalities between the categories are.

On the one hand, agents need to agree upon which syntactic pattern (word order) to use for expressing such a combination of semantic categories. For example, the semantic pole of the construction in (4) is equivalent to the one in (1), but its syntactic pole specifies a different pattern (see the `meets` constraint):

```
((?Top (sem-subunits (== ?F-unit ?P-unit)))
 (?F-unit (sem-cat (== (feature ?x))))
 (?P-unit (sem-cat (== (participant ?x))))
 ((J ?new ?Top)
          (sem-cat (== (participant ?x)))))
<-->
((?Top (syn-subunits (== ?F-unit ?P-unit))
       (form (== (meets ?P-unit ?F-unit))))
 (?F-unit (syn-cat (== (Adjective ?x))))
 (?P-unit (syn-cat (== (NP ?x))))
 ((J ?new ?Top)
          (syn-cat (== (NP ?x))))).                    (4)
```

The first score associated with a grammatical construction reflects this competition between different patterns: out of every set of grammatical constructions with equivalent semantic poles but different syntactic patterns, only one should be selected. Therefore, for every specific combination of semantic categories, the scores of all constructions that select for it but are in competition because they specify different syntactic patterns sum to 1. Each individual score reflects the (estimated) probability that the other members in the population prefer that pattern. This is very similar to the probability score used for lexical constructions, except that there the probabilities were defined over a set of semantic poles (meanings) competing for the same word instead of over a set of syntactic poles (patterns) competing for the same meaning (combination of semantic categories.)

On the other hand, the agents need to agree upon which set of combinations of semantic categories to use when building hierarchical phrases. This reflects the choice between the different solutions (e.g. several 'holistic' grammatical constructions or one recursive one, or agents could choose to use a different word order depending on the number of Adjectives etc.)

In principle, only some kind of transitive and a recursive Adjective-NP construction are needed. However, as explained, agents may propose other construction, like the one below:

```
((?Top (sem-subunits (== ?A-unit ?P-unit ?V-unit ?F-unit)))
 (?A-unit (sem-cat ((participant ?a))))
 (?P-unit (sem-cat ((participant ?p))))
 (?V-unit (sem-cat ((event ?e ?a ?p))))
 (?F-unit (sem-cat ((feature ?a))))
 ((J ?new ?Top)
          (sem-cat ((scene ?e ?a ?p)))))
<-->
((?Top (syn-subunits (== ?A-unit ?P-unit ?V-unit ?F-unit))
       (form (== (meets ?A-unit ?P-unit)
                 (meets ?P-unit ?V-unit)
                 (meets ?V-unit ?F-unit))))
 (?A-unit (syn-cat ((NP ?a))))
 (?P-unit (syn-cat ((NP ?p))))
 (?V-unit (syn-cat ((Verb ?e ?a ?p))))
 (?F-unit (syn-cat ((Adjective ?a))))
 ((J ?new ?Top)
          (syn-cat ((S ?e ?a ?p)))))                    (5)
```

Such constructions are indeed proposed or adopted by the agents because this happens on the fly as they need them. This is as intended because we don't want to bias the agents towards only using recursive constructions. Still, an agent needs a mechanism to decide which analysis to prefer: using construction (5) or the (necessarily conflicting) combination of a normal transitive and Adjective-NP one if these would also be part of his constructicon. For this reason, constructions have an additional preference score associated with them. It enables the agents to represent a preference among the different possibilities similar to the lexical synonymy scores.

Let us now proceed with explaining how agents apply their constructicon. As was the case with the lexicon, after some time the speaker will have multiple ways to apply constructions. Some of them will be equivalent (e.g. it really makes no difference whether construction (2) is first applied to units `F1-unit`

and `P-unit` or first to units `F2-unit` and `P-unit` in structure (1)), but some will not. Certain combinations of grammatical constructions might resolve more equalities than others. Again, we'll score the result of applying a specific combination of constructions.

The reasoning behind the scoring mechanism is similar to the one used for the lexicon. On the one hand, the speaker should optimize the probability that the topic description is conveyed successfully (i.e. take probability scores into account). On the other, he should also take preference scores into account. So for every construction used, the product of its probability, its preference score and the number of equalities it covers is counted and the sum of all these numbers is then divided by the number of equalities in the original structure. This way, unresolved equalities contribute a score of zero, similar to uncovered predicates in case of the lexicon. If several combinations of constructions receive the same score, one of them is chosen at random.

Note that combinations of grammatical constructions that cover all equalities are not a priori preferred over combinations that leave some equalities unresolved. Also, this scoring mechanism favors strongly holistic constructions because all of the equalities covered by it benefit from the high score.

If the winning combination of constructions leaves some of the equalities unresolved, the speaker proposes a new construction, picking a random ordering between the units involved. The result is a semantic and syntactic structure of which the hat contains the top unit and one subunit only. All leaf units are ordered and contain string information. The concatenation of all these strings according to the total order specified by the meets constraints in all but the leaf units is the utterance which is presented to the hearer for parsing.

6.3 Lexicon Lookup and Adopting Words

A hearer agent receives an utterance as one long sequence of characters, even if according to the speaker it consists of multiple words. The first challenge for the hearer is now to decompose this sequence back into words according to the entries in his own lexicon. This is done as follows.

First, all lexical entries are determined of which the string specification is contained in the utterance. For example, if the utterance is "verybigredball", and if the hearer would know the lexical entries e_1 for "big", e_2 for "red" and e_3 for "redball", then these would be selected.

Next, from this set, all combinations of entries are determined that do not overlap. So in the example case there would be five sets: the three containing one entry only ($\{e_1\}$, $\{e_2\}$ and $\{e_3\}$), the set $\{e_1,\ e_2\}$ and the set $\{e_1,\ e_3\}$. Entries e_2 and e_3 cannot be in the same set for this utterance because they would overlap

(they could be in the same set if the utterance was "verybigredredball" and another set would then contain two times the entry for "red".)

Every combination now determines a decomposition of the utterance into an ordered set of lexical entries and strings left uncovered. For example, the combination $\{e_1,\ e_2\}$ results in the decomposition ("very", e_1, e_2, "ball"). The combination $\{e_1,\ e_3\}$ into ("very", e_1, e_3).

Each of these decompositions is rated as follows. The products of the probability and synonymy scores of each lexical entry in a decomposition are summed. this number is then divided by an estimate of the number of predicates in the topic description. If the decomposition that is being rated doesn't contain any uncovered strings, then the estimate is simply the total number of predicates as specified by the lexical entries in the decomposition.

If there is at least one part of the utterance left uncovered however, the total number of predicates is estimated as the number of components in the biggest decomposition found, e.g. 4 in the example case since ("very", e_1, e_2, "ball") is the biggest decomposition found. Thus, all decompositions that contain uncovered strings are treated equally with respect to the number of predicates they are estimated to introduce.

Next, the decomposition with the highest rating is selected. If several decompositions exist with equal rating then one of them is selected randomly. The selected decomposition can now be transformed into an initial syntactic and semantic structure as follows:

- Every uncovered string contributes an empty semantic unit and a corresponding syntactic unit containing nothing else but the uncovered string in its form feature.
- Every lexical entry in the decomposition contributes a semantic unit containing the meaning predicates and semantic category as specified by the entry, and a corresponding syntactic unit containing the string and syntactic category as specified by the entry.
- The syntactic top unit's form feature is initialized with `meets` constraints between the units reflecting the order in which the corresponding strings appear in the utterance.

For example, and assuming that according to the entry e_1 the meaning of the string "big" is `((big ?x))`, and that according to e_2 the meaning of "red" is `((red ?y))`, then if the decomposition ("very", e_1, e_2, "ball") is chosen, the initial semantic structure looks like:[4]

[4]Note that care has to be taken that no unintended variable equalities are introduced. For example, if the same lexical entry appears more than once in the selected decomposition then all variables appearing in it have to be renamed.

```
((Top (sem-subunits (U1 U2 U3 U4)))
 (U1)
 (U2 (meaning ((big ?x)))
     (sem-cat ((feature ?x))))
 (U3 (meaning ((red ?y)))
     (sem-cat ((feature ?y))))
 (U4)),
```

and the syntactic structure like:

```
((Top (syn-subunits (U1 U2 U3 U4))
      (form ((meets U1 U2)
             (meets U2 U3)
             (meets U3 U4))))
 (U1 (form ((string U1 "very"))))
 (U2 (form ((string U2 "big")))
     (syn-cat ((Adjective ?x))))
 (U3 (form ((string U3 "red")))
     (syn-cat ((Adjective ?y))))
 (U4 (form ((string U4 "ball"))))).
```

6.3.1 Adopting Words

As explained, the hearer will only continue if there is at most one uncovered string in the utterance according to the selected decomposition. If there are more uncovered strings, a failure is signalled to the speaker. He will also only continue if the set of predicates in the semantic structure makes sense according to the context and signal a failure otherwise. If there is exactly one uncovered string, he will determine the set of possible meanings for it taking into account the meanings of the (presumed) known words and the set of possible topics consistent with these and the context. He will then add the appropriate set of lexical entries to his lexicon and signal the speaker that he has learned something. This ends the interaction. Only if there are no uncovered strings and if the utterance makes sense after applying the lexicon will he proceed with applying his constructicon. This will be explained in the next section.

Note that if a speaker agent uses a number of words all unknown to the hearer, then the hearer will assume that the utterance consists of one word. If anything, this mechanism favors holistic words. Note also that it is possible that a hearer analyzes a word as uncovered, even if he knows it. Indeed, assume that the utterance is analyzed as consisting of two strings s_1 and s_2. Assume

furthermore that, according to the entry e_i in the hearer's lexicon, the meaning of s_i consists of a set of n_i further unspecified predicates with a probability times synonymy score of p_i. The table below summarizes the possible analysis and their scores:

decomposition	score
$(s_1 s_2)$	0
$(e_1,\ s_2)$	$p_1/2$
$(s_1,\ e_2)$	$p_2/2$
$(e_1,\ e_2)$	$\frac{(p_1+p_2)}{n_1+n_2}$

If $p_1 > p_2$ then the second analysis (decomposition $(e_1,\ s_2)$) will be preferred over the full decomposition $(e_1,\ e_2)$ iff

$$\frac{p_1}{2} > \frac{p_1 + p_2}{n_1 + n_2}$$

or so iff (with $n_1 + n_2 > 2$)):

$$p_1 > \frac{2p_2}{n_1 + n_2 - 2}.$$

Still, if an incomplete analysis is selected, the hearer will use the knowledge he already had about the presumed unknown word and combine it with the new knowledge he will acquire by 'learning' the unknown word. This will become clear later.

To conclude, after lexicon lookup, the hearer either signals a failure, that he has learned something or he moves on to applying his grammar. A failure occurs if the selected decomposition of the utterance results in more than one unknown string or if the meaning resulting from the selected lexical constructions doesn't make sense in the context (i.e. if there is no interpretation.) He signals that he has learned something when there was exactly one unknown string. Both cases also end the interaction (apart from learning, see section 6.5.) In the third case, lexicon lookup results in a semantic and syntactic structure both constructed from a set of lexical entries that together completely cover the utterance and of which the combined meanings make sense in the context. Note however that the meanings do not yet contain any variable equalities, these will be added by the grammatical constructions.

6.4 Applying the Grammar Parsing

As was the case for a speaker while producing an utterance, a hearer will search for some best set of grammatical constructions. Ideally, he should find a set of

constructions that transforms the semantic and syntactic structures such that their hats contain two units only (remember that every construction applied introduces a new sub-unit of the top unit and pushes all units it covers out of the hat. But also if all variable equalities are expressed in a holistic fashion will the hat contain two units only.) If this is not possible, then the interpretation of the meaning predicates present in the final semantic structure should enable him to detect additional equalities between variables and learn a new construction.

The set of already acquired grammatical constructions to apply is determined in a similar way as for the speaker: all possible combinations of constructions are considered and rated and the best combination is chosen. There is a difference however: the hearer doesn't know the intended topic description and hence neither does he know which variables should be identified. What he does know is that every equality should be covered (see assumption 5.) Therefore, the heuristic we'll use is to prefer the analysis that introduces the maximum number of variable equalities. If multiple such analysis exist, the one that has the highest combined preference score is preferred, where the combined preference score is computed as the product of all preference scores of the grammatical constructions involved. This neglects the pattern (probability) scores however. Therefore, among the set of constructions that have equivalent semantic poles, we only consider the one with the biggest pattern score. If there are several such constructions then one is selected randomly (this is similar to what happens during lexicon lookup: only one meaning is considered per word.)

After applying the winning set of constructions, several things can happen. First of all, the hearer will only continue if the parsed meaning so far (but now including variable equalities) still makes sense in the context. If not then the hearer signals a failure and the interaction ends.

If the parsed meaning does make sense, still one of two things can happen depending on whether the hat of the semantic structure contains additional uncovered equalities after interpretation. For example, the hat could look as follows (unit and variable names are arbitrary):

```
((Top (sem-subunits (A-unit NP-unit)))
 (A-unit  (sem-cat ((feature ?x)))
          ...)
 (NP-unit (sem-cat ((participant ?y)))
          ...)).
```

Interpreting the meaning predicates present in the semantic structure results in a set of bindings for all variables to actual object references: $[?x/X,\ ?y/X]$. When these bindings are substituted into the semantic structure it becomes:

```
((Top (sem-subunits (F-unit P-unit)))
```

```
(F-unit (sem-cat ((feature X)))
        ...)
(P-unit (sem-cat ((participant X)))
        ...)).
```

An additional equality is introduced between the units `A-unit` and `NP-unit` which should also be covered by a construction that apparently still needs to be learned by the hearer. He should of course take care that the learned construction is in accordance with the speaker's utterance. If this is possible, then the hearer signals that he has learned something. If not then the interaction fails. (see page 90, section 5.5.2 for an example of why this could fail.)

If no additional equalities are introduced after interpretation then the interactions ends successfully.

6.5 Learning

In this section we'll discuss how the constructional scores are updated after an interaction. Let us start with recapitulating the possible interaction schemes (see figure 6.2. Since the speaker always succeeds, we only have to look at what happens at the hearer side to identify the different interaction endings.)

There are basically three different interaction endings: learned, failure or success. The updating steps that are performed by the speaker and the hearer in each case are summarized in table 6.1. We'll now explain each of the steps

Result	Speaker	Hearer
Failed	Update Failure	Update Lexicon
		Repair Constructicon
Learned	Consolidate lexicon	Consolidate Lexicon
	Consolidate Constructicon	Consolidate Constructicon
	Update Failure	Update Constructicon
Success	Consolidate lexicon	Consolidate lexicon
	Consolidate Constructicon	Consolidate Constructicon
		Update Lexicon
		Update Constructicon

Table 6.1: The different learning actions taken by the speaker and the hearer for each of the three possible interaction endings.

in more detail.

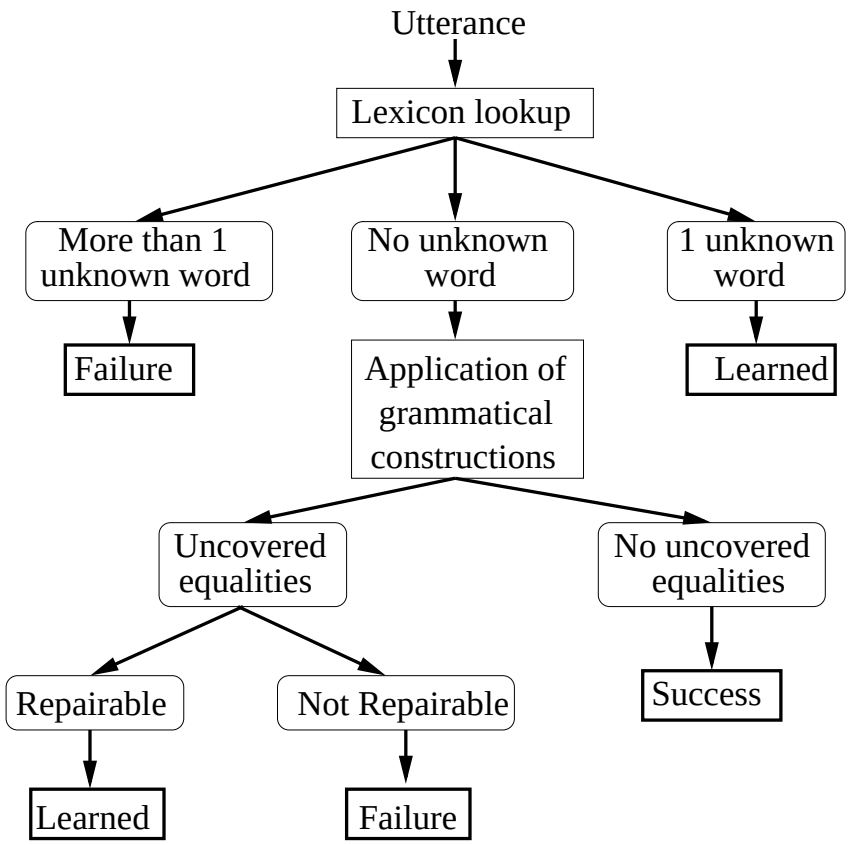


Figure 6.2: Flow chart of a hearer agent showing the possible ending scenario's of an interaction.

6.5.1 Consolidation

In every interaction, new lexical and grammatical constructions might be introduced or adopted. Unless the interaction failed, these should of course be remembered. To 'remember' something simply means that the proposed or adopted entries are actually added to the agent's language inventory with the appropriate scores. This is called consolidation.

Learning Rule 1. *Unless the interaction fails, both the speaker and hearer consolidate their lexicon and constructicon by remembering any lexical or grammatical constructions that were proposed (by the speaker) or adopted (by the hearer.)*

If the interaction fails, both agents forget again whatever constructions they might have proposed or adopted.

6.5.2 Updating Lexical Scores

The idea behind the learning mechanism is that through interacting, the agents sample the population to figure out what the preferences are of other agents in the population. However, the speaker doesn't really get to know anything about the hearer's preferences except in case of a failed and learned game: it is not because a hearer successfully *understands* the speaker's utterance that he also would himself *prefer to produce* the same utterance. Therefore, only the hearer may update his constructional scores in case of a successful game (recent theoretical results confirm this finding: letting the speaker always update his scores with the same strength as the hearer might even prevent the population from converging [private communication with Bart De Vylder]. In addition, this also seems to be compatible with parent-child interactions.)

The scores of the lexical entries used are updated using the cross-situational learning mechanism of Chapter 2, section 2.3.2. However, in that chapter only one-word utterances were considered.

The strategy adopted here consists of simply applying the learning algorithm for every word involved separately. In other words, for every word w^* in the utterance the set of compatible meanings is determined according to the context but without taking into account the fact that the other words actually cover parts of it. This means that the knowledge about the set of possible topics in the context is not used either (however, it is used when learning a new word.) Instead, all combinations of predicates in the context scene description up to length 4 are considered as possible meanings of every word (the limit of 4 was set simply for keeping simulations times reasonable.)

In principle, one could devise a more sophisticated learning scheme that narrows down the set of compatible meanings for every word by combining the information provided by all words together as well as the set of possible topics given the context. In fact, it can be shown that treating all words independently isn't sufficient to learn the meaning of words under certain circumstances (e.g. when a word never occurs on its own, but only in combination with other words.) It is possible to remedy this flaw, but since this was not needed for the current experiment this was not done.

Note that the hearer not only updates his lexicon in case of a successful interaction: also in case of a failure should the meaning of the words involved be a subset of the predicates in the context.

Learning Rule 2. *In case of a success or a failure, the hearer updates the probability and synonymy scores of all lexical entries used independently and according to the cross-situational learning scheme of Chapter 2, section 2.3.2.*

In short, this amounts to increasing all probability scores compatible with the observed words and meanings in the context, decreasing all probability scores incompatible with what is observed, increasing the synonymy scores of all observed words and laterally inhibiting the synonymy scores of all synonyms.

6.5.3 Updating Scores of Grammatical Constructions

The updating mechanism for grammatical constructions is very similar to the one used for lexical constructions. Again, only the hearer updates probability or pattern scores. remember that these scores represent an estimate of which syntactic pattern (word order) is preferred to express a certain combination of semantic categories.

If the interaction ended because the hearer learned a new word then no constructional scores are changed. (The learning of the new word ends the interaction so no grammatical constructions are considered at all.)

Also, if the interaction ends in a failure because too many words were unknown then the hearer cannot infer any information about which pattern is preferred by the speaker (this would require information about the meaning and syntactic categories of the unknown words which is unavailable or too uncertain.)

On the other hand, if the interaction failed because the hearer used an incompatible grammatical construction he still might try to repair things. This will be explained in the next section.

In all other cases, i.e. when the interaction ends successfully or when the hearer successfully learned a new construction, the hearer updates his constructional scores. In these cases, the hearer successfully used and adopted zero or

more grammatical constructions. He will therefore update the patterns scores of these constructions in the same way that observed lexical constructions are updated: all constructions used are enforced and those with incompatible patterns are demoted. The set of incompatible constructions are those with equivalent semantic poles but different syntactic poles (different word order specification.)

The preference scores are also not changed by the speaker but only by the hearer. However, in contrast with lexical synonymy scores, there is no lateral inhibition among preference scores. The updating scheme for the pattern scores already has the effect of laterally inhibiting competing patterns (that is patterns with equivalent semantic poles or constructional synonyms.)

Learning Rule 3. *When the interaction end successfully or when the hearer successfully adopted a new construction, the hearer increases the pattern scores of all grammatical constructions used and decreases the pattern scores of incompatible constructions (i.e. those with equivalent semantic poles as the ones used but with different syntactic poles.) The preference scores of the used constructions are also increased.*

6.5.4 Repairing the Constructicon

As was mentioned in the previous section, when an interaction fails even though the hearer was able to perform a complete parse, he might try to repair things and still learn something. Consider for example the case that the speaker uses an SVO pattern to construct a transitive sentence. Assume furthermore that all words used by the speaker are successfully understood by the hearer, but that he himself prefers an OVS pattern. In this case, the interaction will fail because the hearer's parse will result in the wrong set of equalities between the variables present in the semantic structure. For example, instead of having a parsed meaning of:

```
(P1 ?a) (E1 ?e ?a ?p) (P2 ?p),
```

(meaning that there is an event `E1` with agent `P1` and patient `P2`), he will end up with a meaning of:

```
(P2 ?a) (E1 ?e ?a ?p) (P1 ?p)
```

(the agent and patient variables of the event predicate are wrongly linked to the actual participant predicate variables.)

For this reason, whenever a hearer was able to perform a complete but unsuccessful parse, he will re-do the parse but without applying one of the applied

grammatical construction. If after that he is able to learn a new construction that does result in a successful interpretation, he will adopt it and update constructional scores as explained in the previous section.

Note that the newly learned construction actually might already be present in the hearer's constructicon, but with a lower score than the one initially preferred. The repair mechanism explained here actually is a kind of one-level backtracking mechanism as is used extensively in many common search and learning algorithms. It would in principle be possible to endow the agents with even more sophisticated repair strategies implementing higher level backtracking that considers parses without not only one but several previously applied constructions etc. This might speedup convergence times but is was found not to be necessary for reaching convergence in the current experiment, the power of self-organization is enough.

Learning Rule 4. *If the hearer was able to perform a complete but faulty parse, he backtracks one level by trying to use an alternative (and possibly new) construction instead of one of the initially preferred ones*

6.5.5 Failure Update

Most of the previous cases concerned the hearer only and we have seen that even in the case of a failed interaction the hearer is able to learn certain things. The reason for not letting the speaker update anything is because the fact that there was a successful interaction actually doesn't provide the speaker with much information about the hearer's preferences (there is some information but not much.) However, after a failed game, or after the hearer signalled that he had to learn something to succeed, the speaker *can* conclude that he should have done something else. Therefore:

Learning Rule 5. *Unless the game succeeds, the speaker decreases the synonymy scores of all lexical entries used, as well as the preference scores of all grammatical constructions used.*

With the learning and language processing mechanisms defined in this chapter we are now ready to perform actual simulations. In chapter 8, a simulation experiment will be presented in which populations of agents evolve a language that corresponds to the minimal recursive solution as defined in the previous chapter. This will support our claim about the emergence of compositional and recursive language. However, in order to make sure that this is not because the learning mechanisms a-priori bias the agents towards compositional language, we'll first present a number of baseline experiments in which also holistic languages emerge. This will be the topic of the next chapter.

Chapter 7

Baseline Experiments

The main claim made in this thesis about the emergence of compositionality and recursion was reformulated in section 5.6.2 as 'those elements of meaning, either atomic or not, that occur more frequently as part of the topic in a communicative interaction, will also have a greater chance of becoming an elementary unit of meaning in language.' In this chapter we want to show that our experimental setup is adequate to support this claim. We'll therefore present a number of baseline experiments in which frequency effects are visible. This means that if certain predicates are correlated, e.g. when there is an increased probability that they occur together in scene descriptions, then there is a good chance that the agents opt for using one, holistic word to express them instead of using different words for each of them separately. In other words, the emerging language will be (partially) holistic. That way, when compositionality does emerge, we'll know that this is not because it was built into the agents and their learning mechanisms.

The results of simulation experiments will be presented mainly in the form of a number of graphs. Each graph shows the evolution over time of a certain measure, like the communicative success, and for different values of a certain parameter (like the population size.) Before presenting actual results, we'll first explain how graphs were obtained and how they should be interpreted. This will be done in the next section.

7.1 Parallel and Sequential Time

The time scale of graphs will be such that at each point in time every agent has, on average, had the same number of interactions, independent of the population size n_a. We'll call this *parallel* time and use a p subscript to denote this as in t_p. Hence, at parallel time t_p, and for a population of n_a agents, there have been

(exactly) $n_a t_p$ interactions. The time scale in which only one pair of agents interact each time step is called *sequential* time, denoted with a subscript s as in t_s. Hence we have $t_p = t_s/n_a$. Using parallel time in graphs allows an easier and, in some sense, more correct way to compare results for different population sizes.

Some of the measures were recorded during a simulation as a *running average*. A running average $r(t_s)$ is calculated as follows:

$$r(t_s + 1) = \tau_s x(t_s) + (1 - \tau_s) r(t_s),$$

with $x(t_s)$ the new actual measurement at time t_s. The initial value $r(0)$ to use depends on the measure that is being averaged.

A running average value of $r(t_s)$ is an approximation of the average of the actual measurements $x(t'_s)$ over the interval $t'_s = t_s, t_s - 1, ..., t_s - 1/\tau_s$. More correctly put, it is a weighted average of all values ever encountered, with older values having exponentially smaller weights as time proceeds. A simulation runs in sequential time. Hence, running averages are also recorded in sequential time. Because we want the averages to be taken for comparable intervals for different population sizes n_a, i.e. for equal intervals in parallel time, the value of τ_s is always set according to $\tau_s = 1/n_a$. In other words, running averages can be interpreted as being calculated with a constant averaging window of $1/\tau_p = 1$ in parallel time.

All graphs (so not only those showing running averages) show averages of 10 independent runs. So every experiment was run 10 times and the recorded measures were averaged. The error bars are all calculated over the 10 runs and are two standard deviations wide (one above and one below the graph.)

To obtain smoother graphs, some measures will be averaged over time such that the point at time t_p is actually the average of all recorded values from time $t_{min} = t_p - C/2$ to time $t_{max} = t_p + C/2$. Although care was taken that identical C values were taken for different population sizes in the same graph, different values for C were used for different graphs (different measures.) The caption below each graph reports the value of C.

7.2 Baseline Experiment 1

As explained, in order to obtain frequency effects we have to manipulate the world such that there are certain correlations between the predicates as they occur in scene descriptions. There are many ways to do this, and we will consider two. In both cases, we will make the problem faced by the agents somewhat easier by having them play language games about $A(x)$ kind of scenes instead of full fledged $S(e, a, p)$ scenes. In other words, the scenes will only contain feature

and participant predicates. For clarity, the definition of $A(x)$ descriptions is repeated below:

1. $A(x) \rightarrow P(x)$ or
   $\rightarrow F(x) \wedge A(x)$, each with probability 1/2.
2. $P(x) \rightarrow (P_i\ x), i \in \{1, ..., n_p\}$, each with probability $1/n_p$.
3. $F(x) \rightarrow (F_i\ x), i \in \{1, ..., n_f\}$, each with probability $1/n_f$.

The first way in which correlations among certain predicates are introduced is by having a number of feature and participant predicates *always* occur together. Therefore, let us define $A_n(x)$ scene descriptions, with n a positive integer with $0 \leq n \leq n_p$, $0 < n_p$ and $n < n_f$ (the number of participant and feature predicates respectively), as follows:

1. $A_n(x) \rightarrow P_n(x)$ or
   $\rightarrow F_n(x) \wedge A_n(x)$, each with probability 1/2.
2. $P_n(x) \rightarrow (F_i\ x) \wedge (P_i\ x), i \in \{1, ..., n\}$ or
   $\rightarrow (P_i\ x), i \in \{n+1, ..., n_p\}$, each with probability $1/n_p$.
3. $F_n(x) \rightarrow (F_i\ x), i \in \{n+1, ..., n_f\}$, each with probability $1/(n_f - n)$.

In words, $A_n(x)$ scenes are like normal $A(x)$ scenes, except that for $i = 1, ..., n$, the predicates $(P_i\ x)$ and $(F_i\ x)$ *always* appear together, and never separately. It is as if for these values of i, the participant predicates $(P_i\ x)$ are replaced by the combination of two predicates $(P_i\ x) \wedge (F_i\ x)$, while at the same time the number of feature predicates is reduced from n_f to $n_f - n$. This is also how the definition of $A_n(x)$ is extended to the case when $n = n_p = n_f$: then $A_n(x) \equiv P_n(x)$. In the following, n_p and n_f will always be equal to 5. These are arbitrary numbers chosen by the author for no special reason, choosing smaller or bigger values doesn't change fundamental results, only convergence times and the like.

Figure 7.1 shows the evolution of the communicative success over time for a population of $n_a = 5$ agents and for different values of the correlation parameter n. Each interaction involves one hearer and one speaker agent. It is ended by the hearer signalling either a success, which is counted as +1, a failure, which is counted as -1, or the fact that he was able to learn something (a word or a grammatical construction), which is counted as 0. The graph shows a running average of this signal, averaged over 10 independent runs and with an initial value of 0.

As can be seen, about 100% communicative success is reached for all values of the correlation parameter n. If $n = 5$ then this happens significantly faster then for smaller values of n. One reason for this is that no grammatical constructions are needed in this case. Another reason is that, in general, increasing n actually

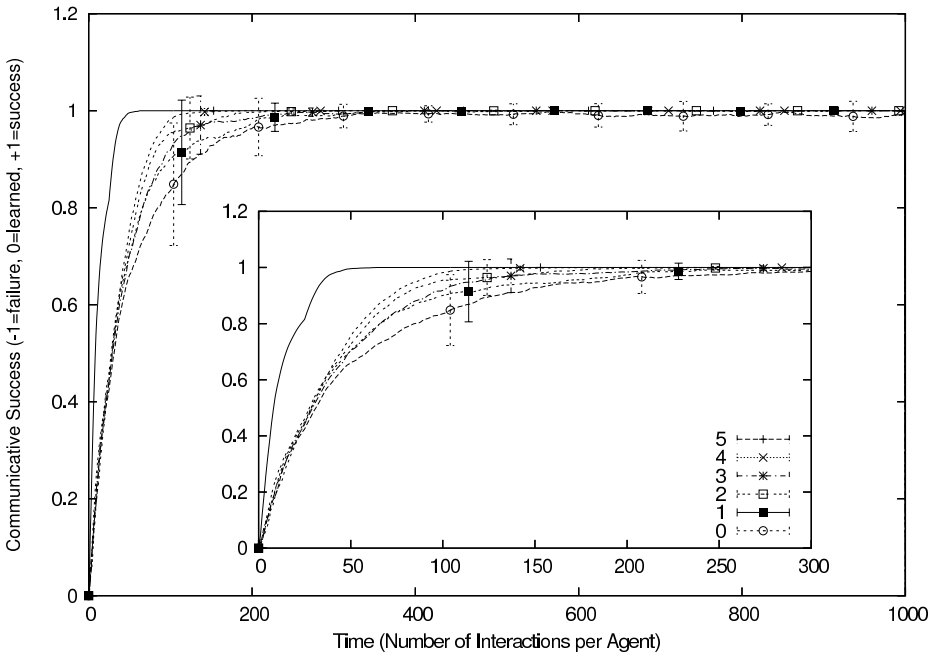


Figure 7.1: Evolution of the communicative success in a population of 5 agents and for different values of the correlation parameter n, see text for further details. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

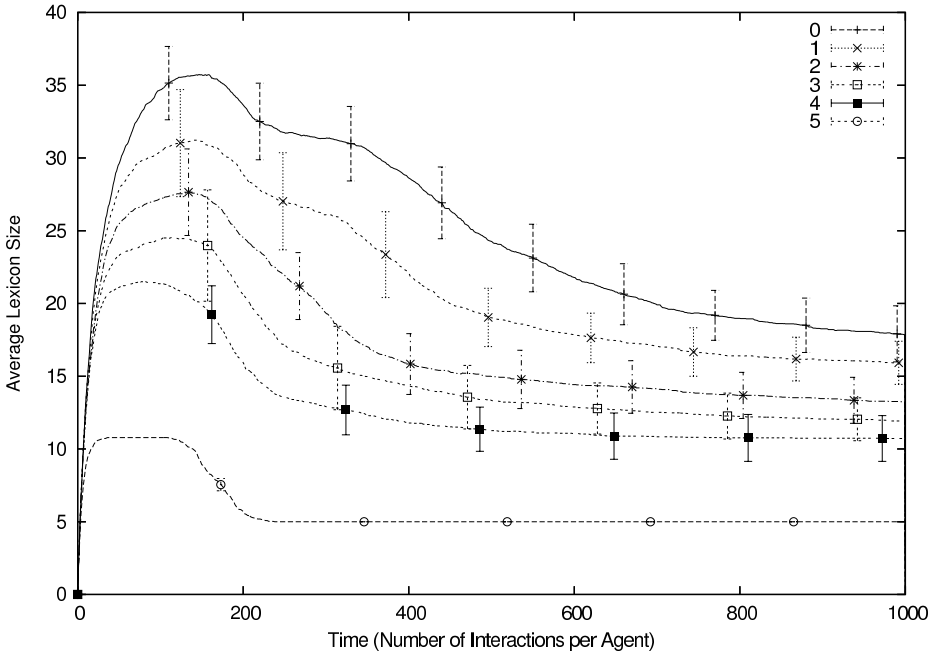


Figure 7.2: Evolution of the number of words known by an agent for different values of the correlation parameter n and measured by inspecting the agent's lexicons and by averaging over all agents in the population. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

decreases the size of the world in terms of the number of elementary units of meaning that occur on their own across different scenes. This means that the agents in principle only have to negotiate and reach agreement about fewer word/meaning pairings. Hence, when n goes up then convergence should be faster and the resulting language should contain fewer words. Figure 7.2 indeed shows that lexicon size decreases for increasing values of n and hence for 'smaller' or 'more correlated' worlds.

If we define the compositionality of a language as the average number of predicates covered per word, then increasing n should also result in less compositional languages. This is indeed the case as can be seen in Figure 7.3. This graph is obtained as follows. Every lexical entry associates a word with a number of predicates in its meaning according to a certain probability. For atomic entries, the meaning pole contains only a single predicate. The meaning poles of more holistic entries contain several predicates. Hence, we would like to define the 'degree of compositionality' of a lexical entry as the reciprocal of the number of predicates in it's meaning.

Words are only associated with a meaning according to a certain probability. Therefore, the compositionality of a word is defined as the reciprocal of the

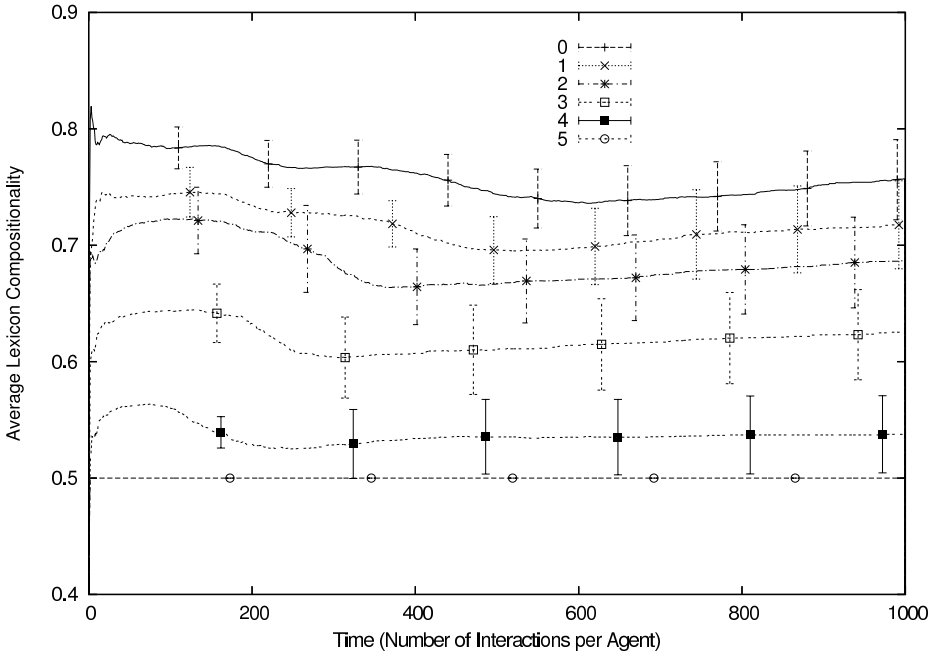


Figure 7.3: Average Lexicon Compositionality ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 0$.)

weighted sum of the number of predicates in all it's associated meanings, with weights the probability of the word having the particular meaning. For example, if the word w^* is associated with the meanings m_1, m_2 and m_3 according to the probabilities p_1, p_2 and p_3 respectively (with $p_1 + p_2 + p_3 = 1$), and if the number of predicates in meaning m_i is given by $|m_i|$ ($i \in \{1, 2, 3\}$), then the compositionality $comp(w^*)$ is given by:

$$comp(w^*) = \frac{1}{p_1|m_1| + p_2|m_2| + p_3|m_3|}.$$

Next, we define the compositionality of an agent's lexicon as the average compositionality of all the words it contains.[1] Finally, the compositionality of the emerging language is defined as the average of the lexicon-compositionality of all agents in the population. Figure 7.3 shows the evolution of this measure.

The careful reader might have noticed that, except for $n = 5$, the agents do not converge to the optimal lexicon size. They also do not converge to the expected degree of compositionality (again except for $n = 5$.) Although the

[1]Note that we don't take synonymy scores into account here. If we would, and if for example all words would have low synonymy scores (which is possible because these scores are also lowered after an unsuccessful interaction), then compositionality would appear low but possibly only for this reason.

number of different predicates remains the same independent of n, the actual number of 'elementary' units of meaning is equal to $n_p + (n_f - n)$. This means that, in principle, if agents succeed in converging to the optimal language in terms of the number of words, the average lexicon size should also be equal to $n_p + (n_f - n)$. And if this would be the case, then the expected degree of compositionality would be equal to[2]

$$\frac{n_p + n_f - 2n + \frac{1}{2}n}{n_p + n_f - n}. \tag{7.1}$$

The table below lists this value for different values of n and for $n_p = n_f = 5$.

n	0	1	2	3	4	5
(7.1)	1	0.94	0.88	0.79	0.67	0.5

Figures 7.2 and 7.3 on the other hand show that the population converges to larger lexicons and smaller compositionality values except for $n = 5$. If the simulation would have been run longer, one could see that all lexicon size curves (except for the $n = 5$ one) converge towards a value between 10 (for $n = 4$) and 15 (for $n = 0$, this can be seen in Figure 7.6, the graph labeled "$p_5 = 0$" corresponds to the $n = 0$ case.) This is because these graphs were obtained by considering all words *known* by the agents, independent of whether these words are correct or actually still *used*. In other words, lexicon size and compositionality, as shown in these figures, are not *observable measures*, but require inspection of the agents' internal states and therefore do not necessarily reveal what is going on during actual games.

This is because the set of words known is not necessarily equal to the set of words used. For example, recall that, as in the naming game, the population first goes through an initial phase in which many new words are proposed and adopted. Only after a second phase of negotiation, and only if lateral inhibition is turned on, will all agents eventually converge to using the same word. However, long after agents have agreed to only use the preferred word, they will still remember the others. These will still be contained in their lexicon's. Hence, taking them into account when computing the lexicon size or the degree of compositionality will give different results from when only the used words are considered.[3]

Moreover, if both holistic and compositional constructs are proposed, then even after full convergence will there be entries in the lexicon for words that are never used anymore. This is because the learning and score-update mechanisms

[2]There would then be $n_p + n_f - 2n$ atomic words and n words covering two predicates.

[3]As was already mentioned in the introduction, iterated learning might remove these redundancies. This will be investigated in section 7.4.

like lateral inhibition or the punishment of unsuccessful words only act on those words that are either used or else are in direct competition with other words that are used, like synonyms. There is no default 'forgetting' mechanism that causes unused words to disappear. So if agents at some point decide to go fully compositional for example, then no mechanism remains to kick out the holistic words that, although they are never used anymore, still occur in the lexicons.

This explains why the graphs in Figures 7.2 and 7.3 deviate from what is expected, even after full convergence. Indeed, Recall from section 6.5.2 that hearer agents consider all possible combinations of predicates up to length 4 as the meaning of words while updating the lexical probability scores in case of a successful or failed game. Hence, although certain predicates never occur separately for $n > 0$, they will occasionally be considered as the meaning of words on their own. This may give rise to atomic words for these predicates after all. At the same time, many holistic words involving more than 2 predicates will be proposed and adopted as well, even some involving predicates that do occur separately across context descriptions. Some of these will remain in the lexicon because they are not used anymore and neither are in competition with any used word.

What this means is that if we want to check the degree of compositionality of the resulting languages for different values of the correlation parameter n, we should only take those words into account that are actually being used in language games. Figure 7.4 shows the evolution of the number of predicates per word actually used in a game at the corresponding time, measured as a running average with initial value 2 (the average number of predicates in a context description.) The table below also lists the expected value of this measure for different values of the correlation parameter n (these values were obtained numerically):

n	0	1	2	3	4	5
expected value	1.0	1.15	1.30	1.45	1.60	2.0

As can be seen, these values correspond almost perfectly with the corresponding curves in Figure 7.4, meaning that the statistics of the world are, at least in this simple case, perfectly adopted by the agents in their lexicons. In other words if, in any experiment presented in this thesis, compositionality emerges, then this it is *not* because our agents are in any way biased towards it but because (1) they are capable of adopting the relevant statistics present in the world and (2) do so because this increases the (expected) communicative success while evolving a language.

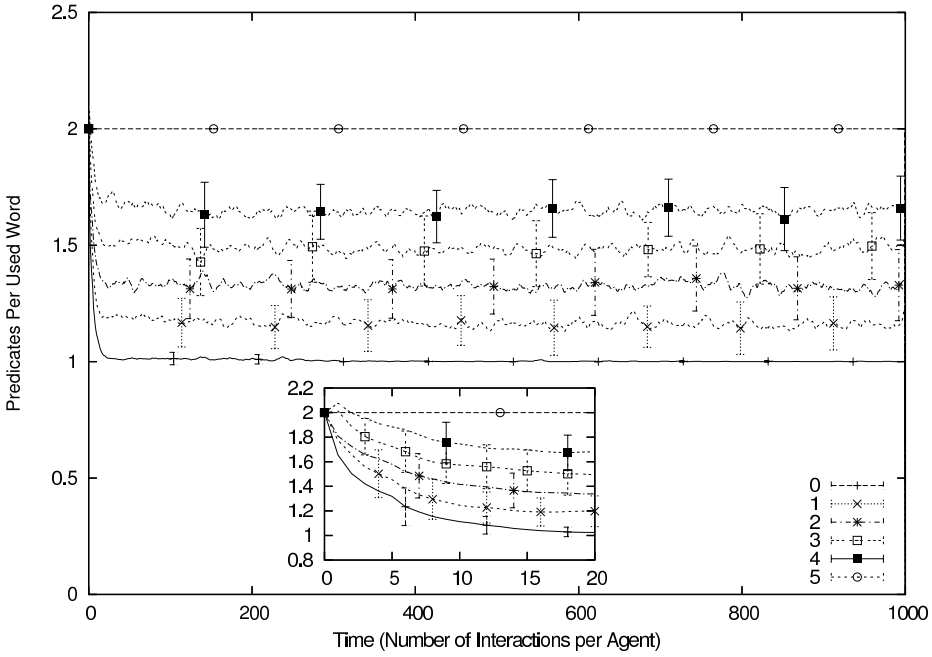


Figure 7.4: Evolution of the number of predicates per used word recorded as a running average. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

7.3 Baseline Experiment 2

The way in which correlations among predicates were introduced in the first baseline experiment was rather straightforward: two predicates were either completely correlated or they were not at all. This allowed us to obtain results that are in complete correspondence with what could be expected, at least for so called observable measures.

In this section we want to investigate what happens if predicates are only correlated up to a certain degree. For that, we'll let the agents talk about $A_0(x)$ scenes (i.e. correlation free scenes) with probability p_0 and about $A_5(x)$ scenes (i.e. scenes with all P_i and F_i predicates perfectly correlated) with probability $p_5 = 1 - p_0$.

Figure 7.5 shows the evolution of the communicative success measured as in the previous experiment and for different values of the correlation parameter p_5. Although it is difficult to see, and while large values of p_5 seem to correspond with in general an initial advantage, the graph shows that agents have more difficulty reaching 100% communicative success unless the correlation among the predicates is perfectly clear, i.e. when p_5 is either 0 or 1 (and of course much faster when $p_5 = 1$.). Still, in all cases perfect communication is achieved.

The size of the language to which the agents converge can be predicted

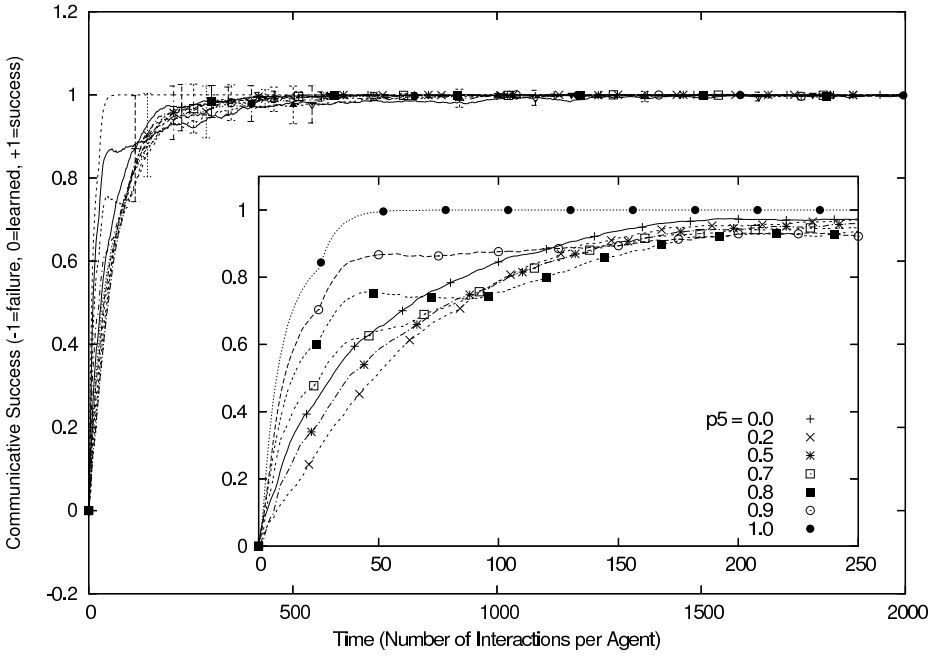


Figure 7.5: Evolution of the communicative success in a population of 5 agents and for different values of the correlation parameter p_5, see text for further details. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

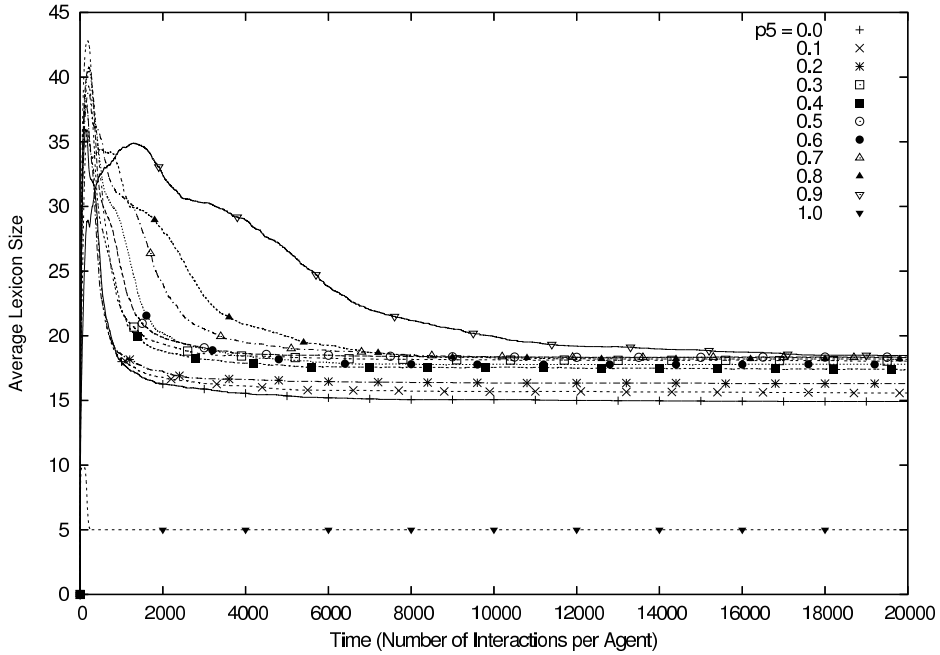


Figure 7.6: Evolution of the number of words known by an agent for different values of the correlation parameter p_5 and measured by inspecting the agent's lexicons and by averaging over all agents in the population. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

somehow this time. One would expect that, except if p_5 equals zero or one, and in first approximation, the agents would need both the set of words needed in the pure A_0 and A_5 cases, i.e. $10 + 5 = 15$ words in total. Interestingly, this seems to be exact for $p_5 = 0$, as can be seen in Figure 7.6. In second approximation, one would expect that the larger p_5, the harder it is for the agents, because there will be less occasions to learn the A_0 part and the world will seem more structured than it actually is. Apparently this confusion causes the agents to try out more combinations of predicates, leading to larger lexicon sizes and less compositional languages (see Figures 7.7 and 7.8.)

7.4 Impact of Population Turnover

So far, we have been able to show that our agent model and experimental setup is adequate to produce some interesting frequency effects. In the introduction we already mentioned that frequency effects could be an ideal candidate for an iterated learning mechanism to have an observable effect on language. If agents are 'confused' like in the previous section, meaning that they try out several

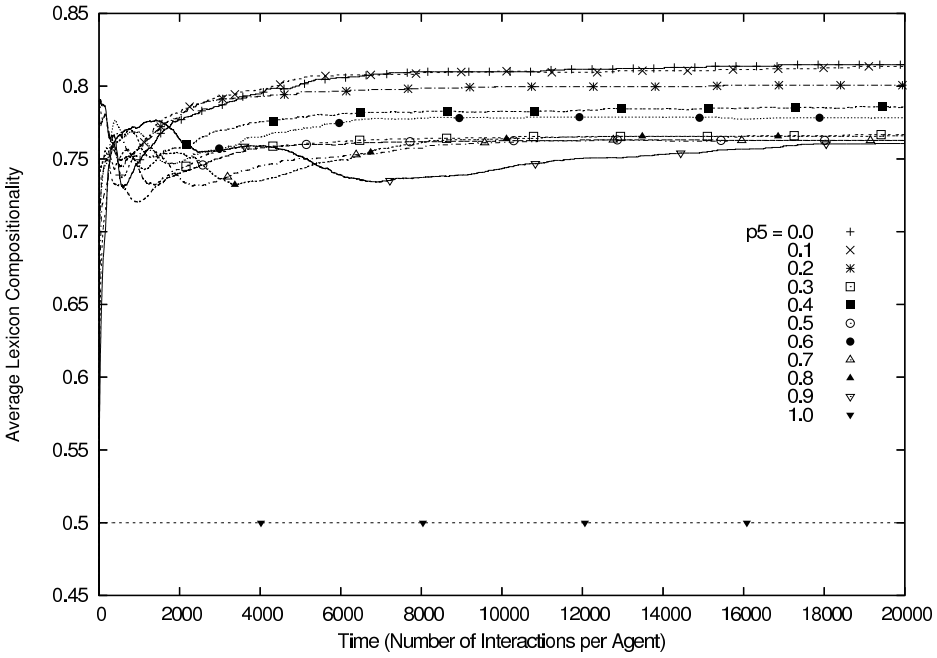


Figure 7.7: Average Lexicon Compositionality ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 0$.)

solutions before reaching convergence but without forgetting all the bad ones, then maybe these idiomatic phrases that survived it by chance would get filtered out after all when new agents enter the population after the language has more or less stabilized.

To test this hypothesis, we carried out a simulation that includes a population turnover as follows. First, 5 agents were allowed to play 600 games, starting from scratch and in a similar setting as in the previous chapter. Then, one of the agents is replaced by a newborn agent. This agent gets to play 200 games as a hearer. Each time, the speaker is selected randomly from the remaining 4 players. However, these do not update their internal state, only the new agent does so. After this the process is repeated: all 5 remaining agents together play 600 normal games. Then one of the oldest agents is replaced by a new agent which is allowed to learn for 200 games and so on. This iteration was performed 100 times, meaning that in the end the population has been turned over completely for 20 times.

Every learning period, communicative success initially obviously drops, but after some time always starts increasing again (see Figure 7.9.) Although this might not seem very interesting, we mention that this would not be the case if the speaker agents in the learning phase (i.e. the teachers) would also update their internal state after every interaction with the learner (the hearer agent.)

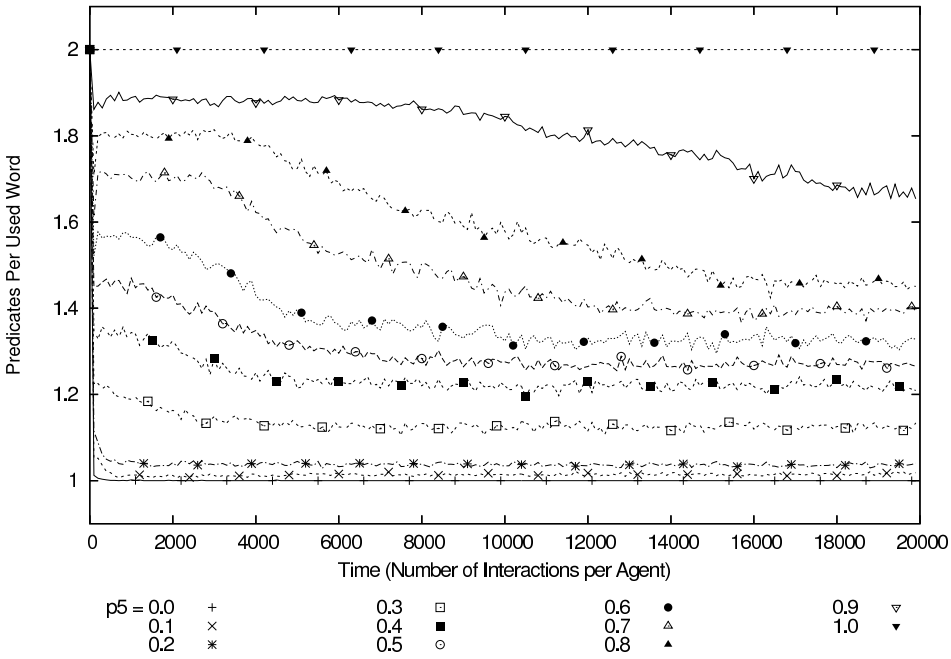


Figure 7.8: Evolution of the number of predicates per used word recorded as a running average for different values of the correlation parameter p_5, see text for further details. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

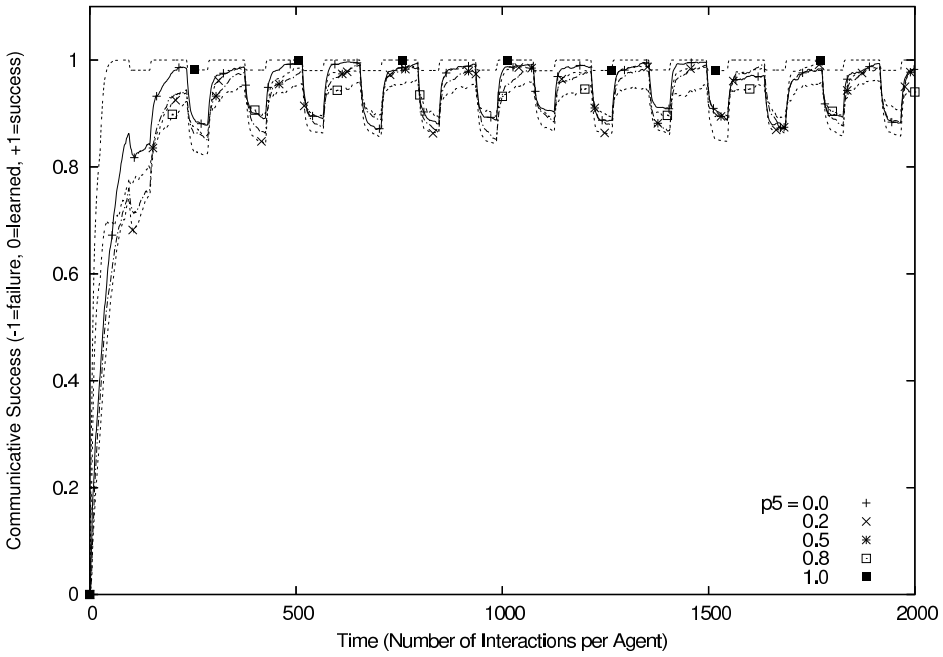


Figure 7.9: Evolution of the communicative success in a population of 5 agents and for different values of the correlation parameter p_5, see section 7.3 for further details. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

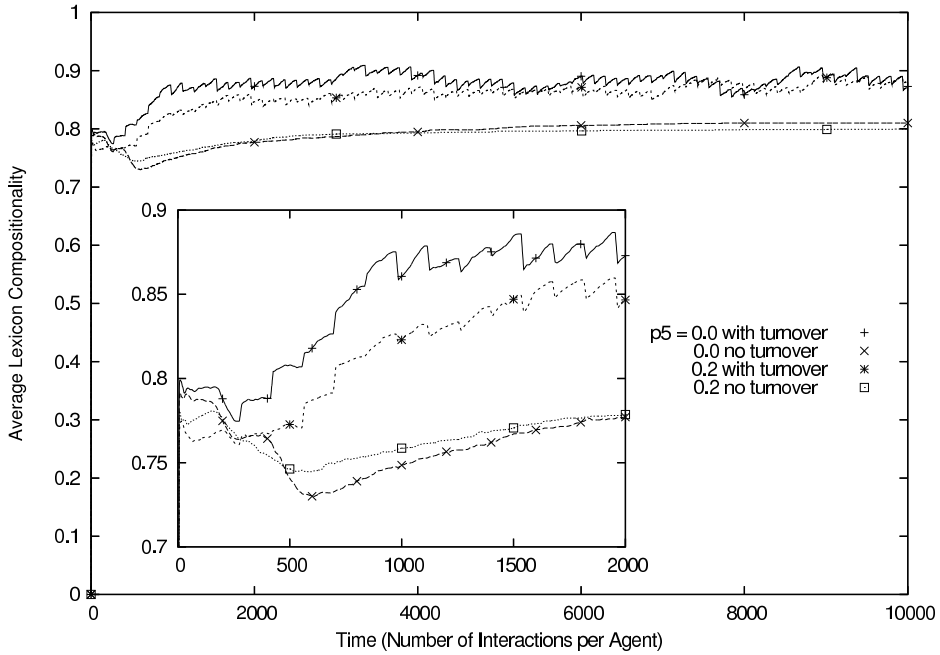


Figure 7.10: Average Lexicon Compositionality ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 0$.)

If they would, then the language would de-stabilize.

What is interesting, and what confirms our intuition, is that compositionality increases every other learning period for $p_5 = 0$ and $p_5 = 0.2$ (and hence probably for all values in between.) Recall that according to our definition, compositionality is not an observable measure because it also takes words into account that are not used anymore. But if they are not used, then maybe a new agent will not adopt them either, which might have an effect on compositionality. This can indeed be seen in Figure 7.10, showing the evolution of the compositionality for $p_5 = 0$ and 0.2, together with the corresponding compositionality curves from the previous chapter in which the same experiment was carried out but without a population turnover. What also might be interesting, but for which I have no explanation, is that this effect seems to be stronger for smaller values of p_5, although further investigation is required. For values of $p_5 = 0.5$ and above no significant effects were found.

For the sake of completeness, let me mention that another thing still unexplained is that now the number of predicates per used word remains at its initial value (see Figure 7.11.) The challenge here is not to explain why it remains the same, but why it *doesn't* in Figure 7.8.

In any case, we have shown that iterated learning indeed is a shaping force of

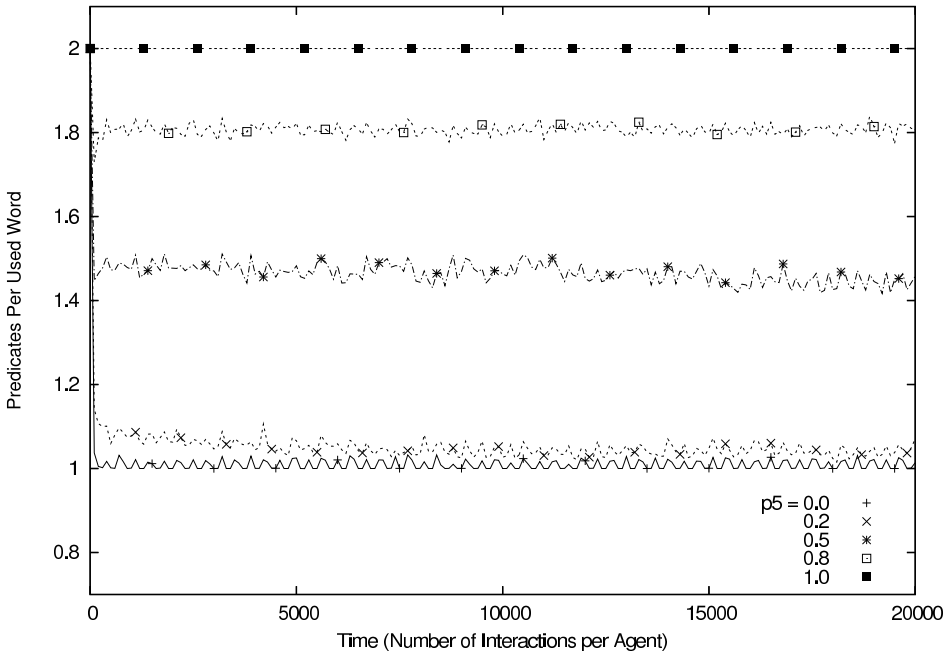


Figure 7.11: Evolution of the number of predicates per used word recorded as a running average for different values of the correlation parameter p_5, see text for further details. ($n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

language, although not with respect to the emergence of compositionality, but rather at the level of frequency effects and certain changes in the environment: while a fixed population would be insensitive to them, a changing one gradually might adopt them. Still this force only works at much larger time scales than is the case for other shaping forces like the urge for successful communication.

Chapter 8

Main Experiment

In this chapter we'll turn our attention back to the initial problem setting. Hence, scene descriptions are full fledged $S(e, a, p)$ descriptions, containing an event predicate, a participant expression for the agent of the event and a participant expression for the patient of the event. There are no correlations among any predicates except for the ones entailed in the rules to generate $s(e, a, p)$ descriptions, nor is there a population turnover.

Therefore, from what we learned from the previous chapter, we can suspect that a fully compositional language will emerge (at least if measured in an observable way, i.e. by counting the number of predicates per used word.) As will be shown shortly, this will indeed be the case.

However, until now we haven't really looked at any grammar related aspects. We don't know yet whether the agents are able to find the minimally recursive solution. This will be investigated in the following. Recall that every simulation, a number of agents needs to bootstrap a communication system from scratch to communicate about a number of scenes containing n_e events in which pairs of in total n_p event participants are involved. Each of the participants can have one or more of n_f features, and every feature can in principle occur an unlimited number of times. The results presented in the following all have $n_e = n_p = n_f = 5$. The learning parameters (α and θ) are always set to 0.2.

8.1 Communicative success

Figure 8.1 shows the evolution of the communicative success over time for population sizes $n_a = 5, 10, 15$ and 20. As can be seen, all populations reach 100% communicative success. As expected, and similar to the naming game, the larger the population the slower the dynamics. At first, all populations go trough a period of mainly failing games, after which they gradually start to learn and

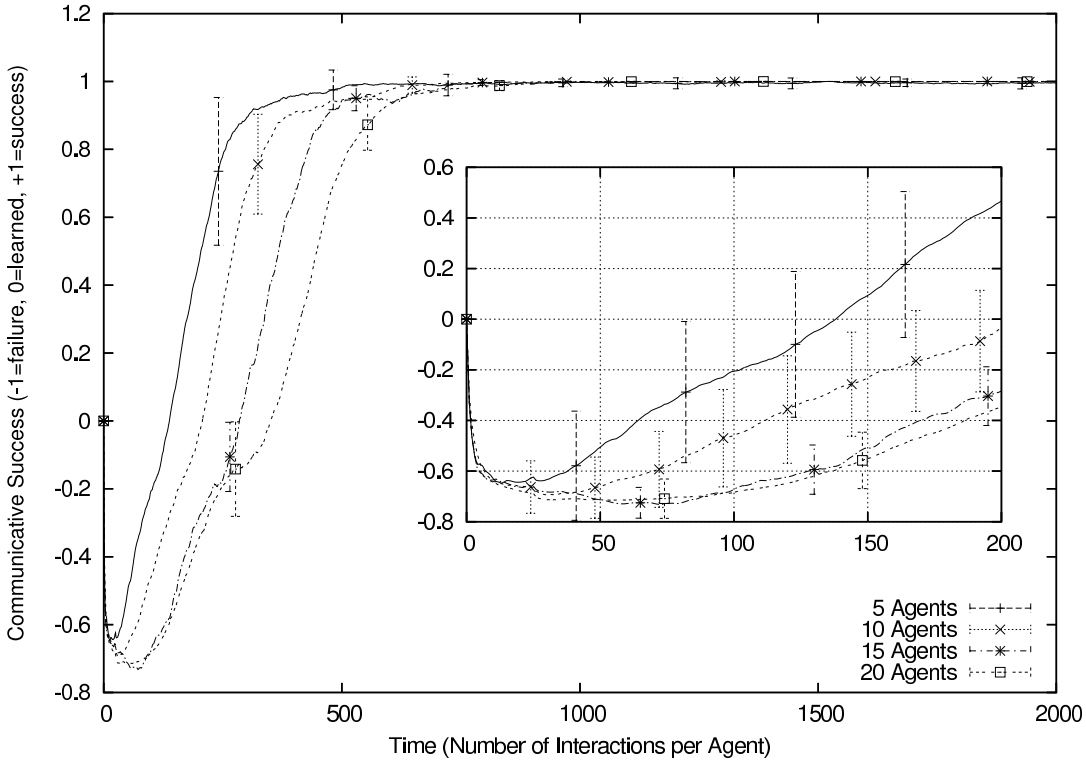


Figure 8.1: Evolution of the communicative success for different population sizes recorded as a running average (see text.) ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 50$.)

finally converge to a successful communication system.

Although a successful interaction does not necessarily imply that the hearer was able to arrive at the same meaning as intended by the speaker, the fact that after some time *all* interactions are successful, and this while scenes and topic descriptions are generated randomly, *does* imply that the return is 1 and hence that communication is successful.

Conclusion 1. *The Learning mechanisms proposed in this thesis are sufficient for a population of agents to bootstrap a successful language (at least up to population sizes of* $n_a = 20$ *agents.)*

All learning mechanisms also seem to be necessary: if either one of them is turned of then it was found that no convergence is reached anymore.

8.2 The Lexicon

8.2.1 Size and Synonymy Score

Let us continue with investigating what happens to the lexicons of the agents. Figures 8.2 and 8.3 show the evolution of the average number of words and the average associated synonymy score in an agent's lexicon. Recall that synonymy scores are preference scores. It could be for example that some agents prefer one word while others prefer another to express the same meaning, i.e. the two words are synonyms. Lateral inhibition is then used on the synonymy scores of both words to let all agents converge to preferring the same word. Note that if all synonymy scores would be 1 (and hence also the average synonymy score), then there would be no conflicts and synonymy would actually be zero.

Note that the number of words does not coincide with the number of lexical entries: every word can be present in the form pole of *several* lexical constructions, namely homonyms.

As was the case with the naming and guessing games (see sections 2.3.1 and 2.3.2), the number of lexical entries first increases because speakers propose new words which are then adopted by hearers and spread through the population.

After some time, words start disappearing again. There are basically two mechanisms by which this can happen. First, whenever a word was used unsuccessfully by a speaker, its synonymy score is decreased (see learning rule 5 in section 6.5.5.) In addition, when a word's synonymy score goes below a certain threshold value (which was set to $1.0e^{-3}$) then the word and all its associated lexical entries are forgotten by the agent. However, note that this mechanism requires that the words that are updated are *used* and even are *used unsuccessfully.*. Looking again at figure 8.1, this mechanism can only contribute to the

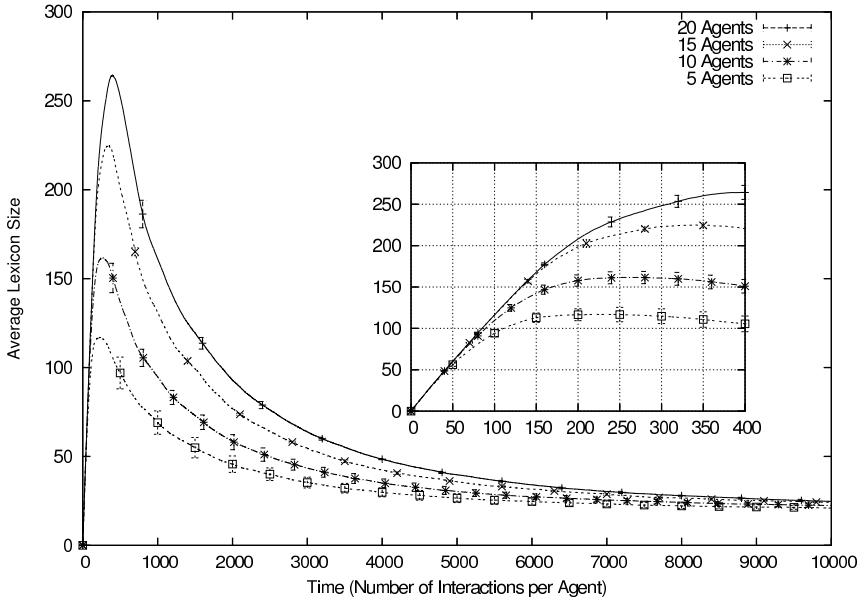


Figure 8.2: Evolution of the number of words known by an agent for different population sizes. ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 1$.)

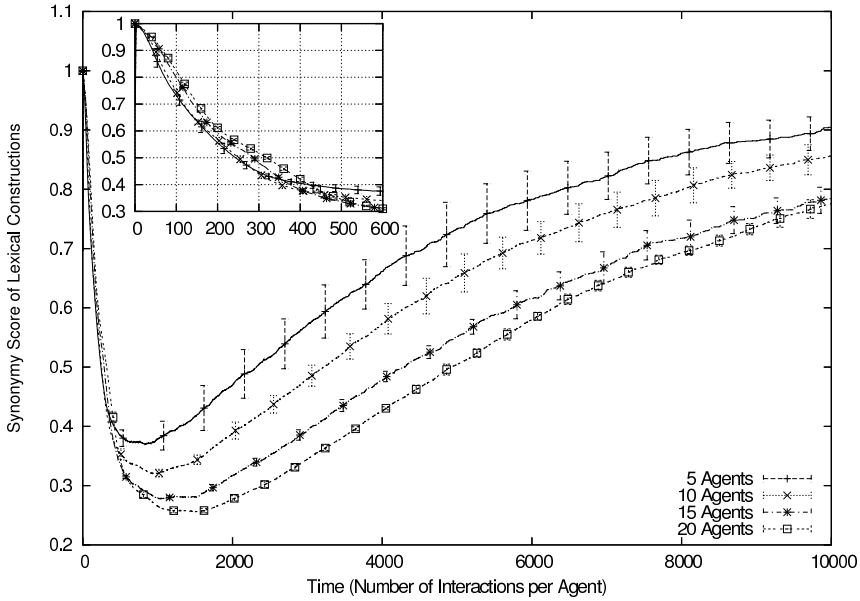


Figure 8.3: Evolution of the synonymy scores for different population sizes. ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 1$.)

lowering of synonymy scores and hence to the forgetting of words until time about $t_p \simeq 500$.

Second, whenever a word is observed by a hearer, its synonymy score is increased and the scores of all its competing synonyms are lowered. Again, this can eventually lead to the forgetting of a number of lexical entries. Also, whenever a word with low to zero synonymy score is removed from the lexicon, the average synonymy score will evidently increase. This is the mechanism responsible for the increase of the synonymy scores and the reduction of the lexicon sizes after $t_p \simeq 500$.

Note that there is a third mechanism reducing the number of lexical entries (but not words): whenever a word is observed by a hearer, the probabilities of its associated lexical entries are updated so that they are in accordance with the context. This may lead to certain probabilities becoming zero (or below a certain threshold), and hence to the forgetting of the associated entries. Although this is not directly visible in figures 8.2 and 8.3 (only the number of words is counted in them), this mechanism does help explain why so many words *do* disappear because of the lateral inhibition dynamics: because the meaning of most words converges towards the same set of meanings which therefore makes them synonyms and puts them in competition. We'll see that this is indeed what happens in section 8.3 on compositionality.

In any case, all populations eventually reach a situation where the lexicons contain about 25 words. Still, the average synonymy score remains below 1. Because all interactions are successful in this regime, and hence all words that are still used eventually should get a maximum synonymy score of 1.0, this means that there is a number of words that on the one hand are never used anymore and on the other have less than maximal synonymy score. Later on we will indeed see that after time $t_p \simeq 100$ only words that have only one predicate in their meaning are still used (see figure 8.6), which accounts for at least 15 out of the 25 words that should get a maximum synonymy score ($n_e + n_p + n_f = 15$).

Furthermore, these observations are also in accordance with the fact that the number of predicates in the meaning of words in the lexicon (i.e. also taking words into account that are not used anymore) steadily decreases but never reaches a value of 1 (see section 8.3.) This is because the lexicons will still contain a number of words that on the one hand are never used anymore and on the other are holistic.

Conclusion 2. *Although internally the agents remember a number of holistic words, these words are never used anymore and hence are not observable from the outside. A flux in the population could hence have the effect of these words really disappearing (because they are never observed), but such a flux is not necessary for explaining the emergence of compositionality.*

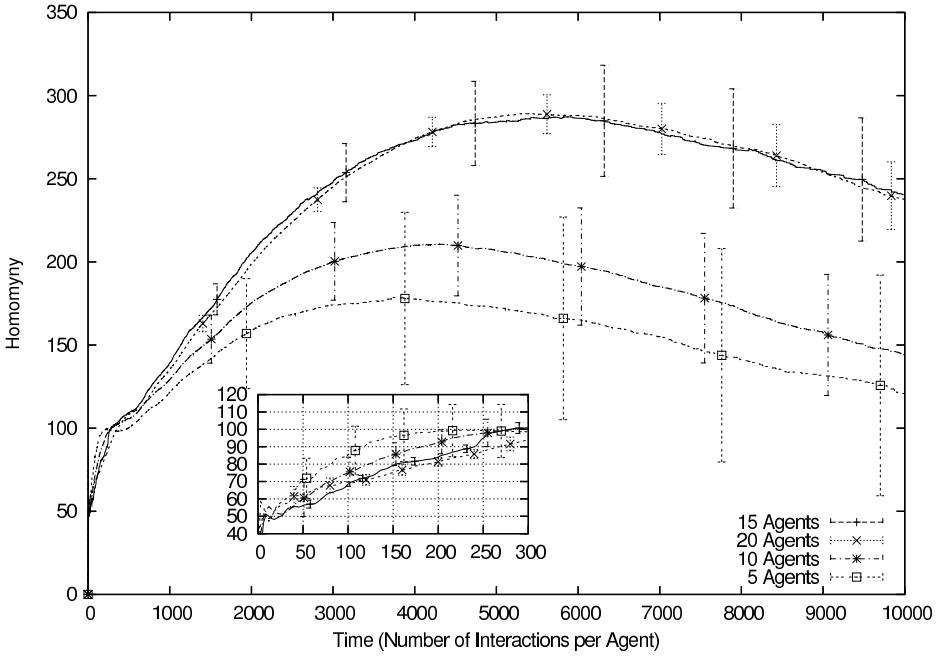


Figure 8.4: Evolution of the (agent internal) homonymy for different population sizes. ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 1$.)

Later on we will have a closer look at compositionality, but first we will continue with our analysis of the lexicon by taking a look at the evolution of homonymy and synonymy.

8.2.2 Homonymy

Figures 8.4 and 8.5 show the evolution of the agent internal homonymy and synonymy. Recall that a homonymy value of h is equivalent to a word on average being associated with $h + 1$ different meanings. Similarly, a synonymy value of s means that on average the same meaning is covered by $s + 1$ different words.

The reason why the average homonymy value is so high is because hearers almost never know the intended meaning of a new word. Hence, if the scene contains l predicates, then the number of possible meanings for an unknown word is equal to[1] $|m| = C_l^1 + C_l^2 + C_l^3 + C_l^4$. The table below lists this value for a number of scene lengths l.

[1]Remember that meanings longer than 4 predicates are not considered by the hearer.

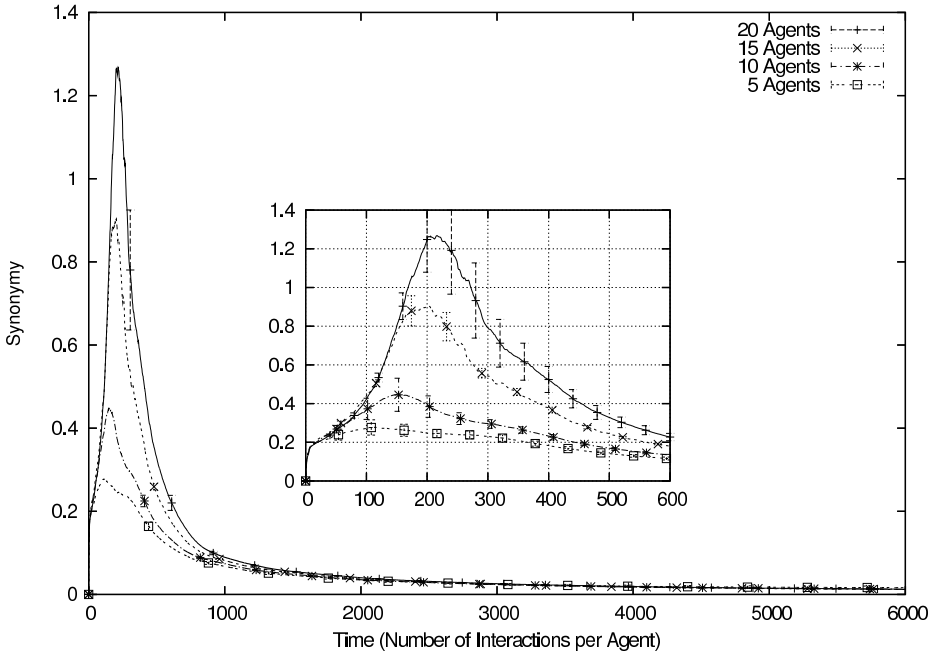


Figure 8.5: Evolution of the (agent internal) synonymy for different population sizes. ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 10$.)

l	1	2	3	4	5	6	7	8	9	10	11	12
$\|m\|$	1	3	7	15	30	56	98	162	255	385	561	793

Homonymy increases with population size although not unlimited: there already seems to be no significant difference between the curves for population sizes $n_a = 15$ and $n_a = 20$. The reason for this is that in smaller populations the homonymy simply isn't allowed to reach its 'natural' maximal value because agents start converging to a shared language earlier.

To understand this, consider that homonymy increases whenever a new word is encountered or when an existing word is used in an unexpected way. On the other hand, the total number of possible meanings for a word is clearly bounded by the fact that only a limited number of predicates and meanings are considered. This puts a limit on the maximum average homonymy value, independent of the population size. At the same time, every interaction can be an opportunity to decrease homonymy because the hearer might be able to tighten the meaning of some words. This latter mechanism clearly works faster in smaller populations, just as they also reach convergence faster: although at a certain point in parallel time every agent has on average had the same number of interactions independent of the population size, still, in smaller populations the chance of encountering the same agent twice is bigger and so is the chance of

encountering an opportunity to tighten the meaning of an already know word. Hence, although this mechanism works too slow for big populations to keep the homonymy below its maximal value, for small enough populations it works fine.

The same mechanism does not apply for synonymy. This is because compared to homonymy there is an additional source of synonymy (see also chapter 2, section 2.2.2): the fact that more speakers will propose more words for the same meaning. For similar reasons that agents in smaller populations have a bigger chance of being able to tighten the meaning of words, there is a bigger chance that agents in bigger populations will have to propose words for meanings that were already covered by other agents. This would predict that the synonymy evolves in tight accordance with the average lexicon size and has a peak value proportional to the size of the population which indeed is more or less the case.

In sum, the agents seem to reach a language that is almost synonym-free. The homonymy remains at a high level, meaning that there are still a number of words for which the meaning is not clear at all. Taking into account the conclusions from the previous sections, namely that the agents seem to end up with 15 words that are used successfully and then about 10 words that are not used anymore and hence also not negotiated about, we can conclude that the high remaining homonymy level must be due to the 10 words that are not used. Everything thus seems to point towards a minimal language in the sense that only words of which the meaning contains only one predicate are used. In other words, everything seems to point towards the emergence of compositionality. In the next section we'll see that this is indeed the case.

8.3 Compositionality

The evolution of the compositionality of the emerging language was measured in several ways.

8.3.1 Utterance Level, Words

First, figure 8.6 shows the evolution of the number of predicates in the topic description divided by the number of words in the utterance. Again, this was recorded as a running average. As before, these graphs were obtained by looking at the topic description and utterance alone. Hence, in contrast with homonymy and synonymy graphs from the previous section, this is an observable measure.

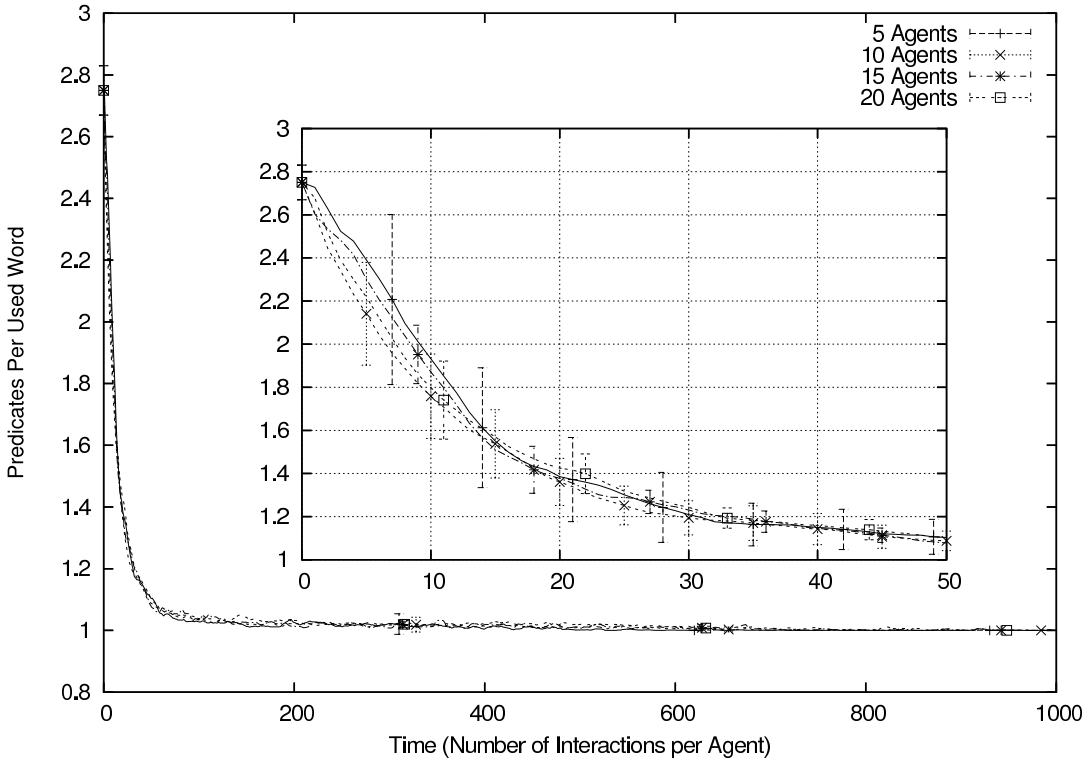


Figure 8.6: The number of predicates in the topic description divided by the number of words in the utterance. For a completely holistic language, this number would be 2.75 (the average number of predicates in a topic description divided by 1). This is also the number at which all graphs start. For a completely compositional language this number would be 1, which is also the value to which all graphs converge. ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 10$.)

For a completely holistic language, the average fraction of the number of predicates per used word would be 2.75 (the average number of predicates per topic description). This is also the number at which all graphs start. For a completely compositional language this number would be 1, which is also the value to which all graphs converge.

The size of the population is of no importance: all populations more or less go compositional equally fast and there are no significant differences at all.

The mechanism that can explain these results is the following. Each interaction the speaker receives feedback about whether the hearer agrees with the utterance or not. If the speaker uses a word the hearer has never encountered before, he will slightly lower the score of that word.

At the same time, the hearer learns the word, so the next time the word is used the interaction might be successful and its score can increase again.

This mechanism is the same for holistic and compositional words. However, there are far fewer occasions in which holistic words can be used. This has as effect that compositional words will become successful and preferred before holistic words can, driving the agents towards compositionality.

The above explanation is somewhat simplified and doesn't take things into account like the fact that compositional utterances also require a set of successful grammatical constructions. Still, we believe that we can state the following:

Conclusion 3. *Compositionality wins from holism in language because compositional language is productive, re-uses already established bits of language and can be used more often (and hence is more* useful*) compared to holistic language.*

Conclusion 4. *Population size has no influence on the rate at which compositionality emerges.*

To understand even better what is going on, Figure 8.7 shows what happens if a single agent would simply build up a lexicon by subsequently decomposing randomly generated topic descriptions according to his lexicon and adding words to it for uncovered parts of meaning while always preferring the most compositional decomposition of the topic description (see 'base' curve in the graph, for comparison the curve for 5 agents was included as well.) Note that there is no communication involved to produce the base curve. It simply represents the fastest possible way that the agents could evolve compositional language as prescribed by the structure of the world (presented to them in the form of subsequent topic descriptions.)

From the graph we can conclude that when communicative success *does* matter, initially holistic words are also used and convergence towards a completely compositional language is slowed down. However, after about 100 interactions

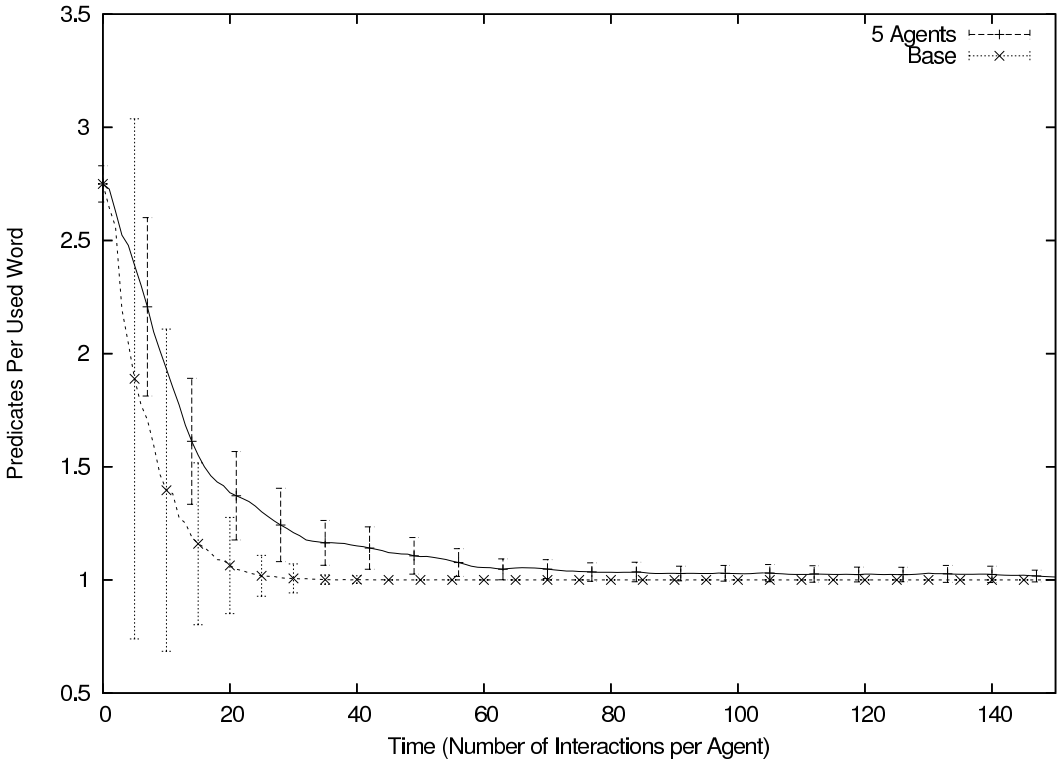


Figure 8.7: The number of predicates in the topic description divided by the number of words in the utterance if one agent would simply always use the maximum number of words possible, not taking communicative success or constructional scores into account. The curve for 5 agents from figure 8.6 is also shown for comparison.

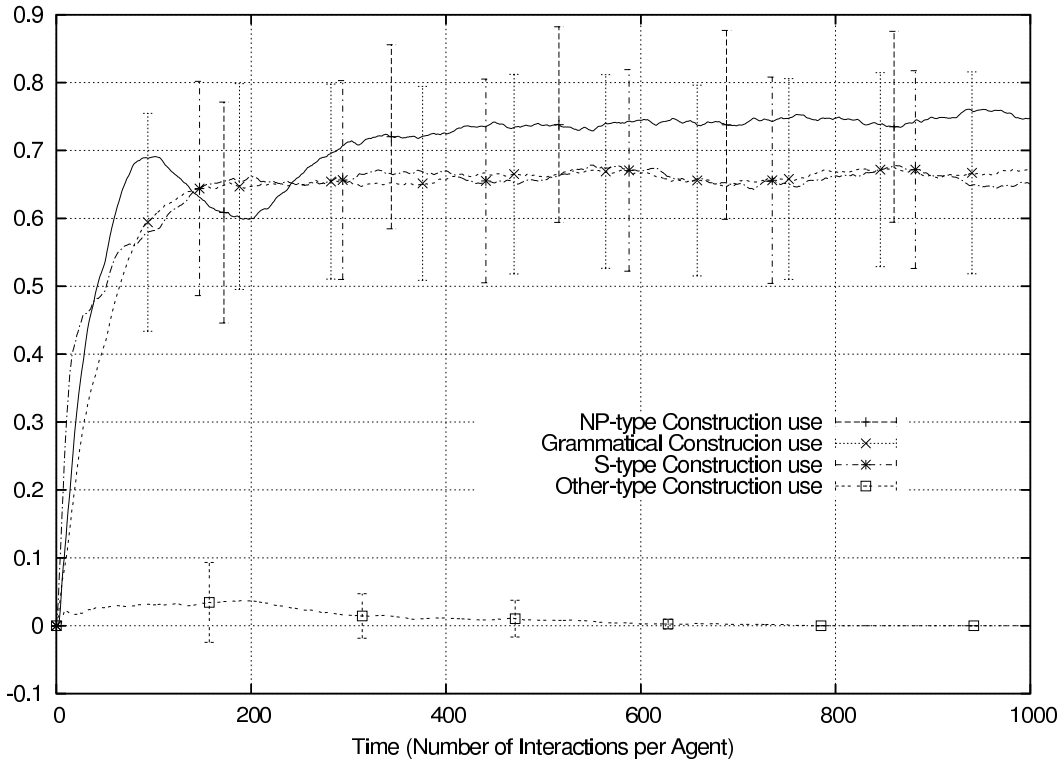


Figure 8.8: The evolution over time of the use of different types of grammatical constructions in a population of 5 agents. See text for more details. ($n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 100$.)

per agent, the language becomes almost 100% compositional, despite the fact that this also involves the evolution of a set of grammatical constructions.

8.3.2 Utterance Level, Constructions

Albeit indirectly, a second way in which one can see that the agents go compositional is by measuring when grammatical constructions are used. Recall that whenever equal predicate arguments are expressed by different words (i.e. whenever a compositional utterance is constructed) the equality should be covered by a grammatical construction. This also goes in the other direction: whenever a grammatical construction is used the utterance must be compositional.

Figures 8.8 and 8.9 show when different types of grammatical constructions are used for different population sizes. The figures show 4 curves. First, the number of times a grammatical construction was used by the speaker in the first place is shown. From section 5.1.1 we know that the average number of times the topic description contains at least one equality is 66.6%, which is indeed

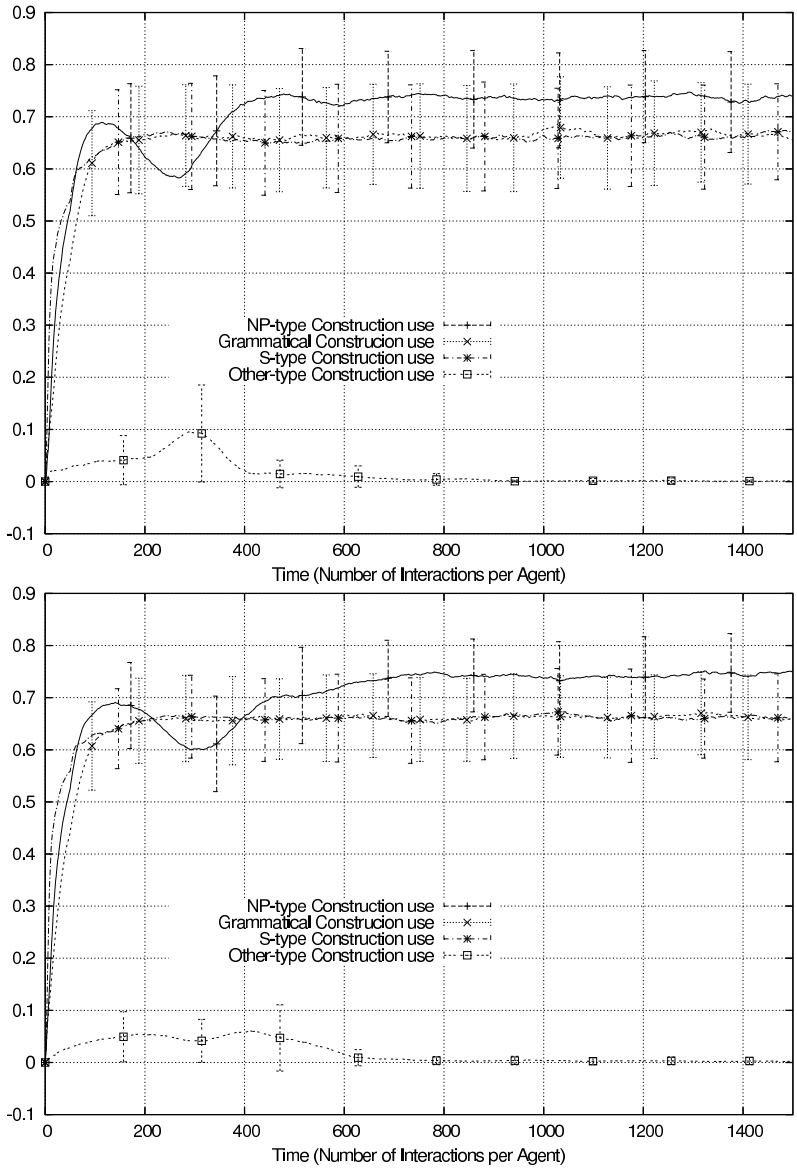


Figure 8.9: Same as Figure 8.8 but for 10 agents (top) and 15 (bottom) agents.

the value around which the curve fluctuates from $t_p \simeq 200$ on, meaning that all equalities are covered by a grammatical construction and hence that the agents prefer compositional over holistic constructs.

The other three curves show the fraction of the times that a construction of a specific type is used *given that a construction is used.* The 'NP-type construction' curve shows the number of times that a construction with resulting syntactic type NP is used. Examples of these include constructions like '(NP) → (Adjective) (NP)', '(NP) → (Adjective) (NP) (Adjective)' etc. These are all observable measures.

The S-type curve shows the number of times a construction with resulting type S is used given that a construction is used. Examples of these include '(S) → (V) (NP) (NP)', '(S) → (V) (NP) (Adjective) (NP)', etc.

The O-type curve shows the use of all other constructions, again given that a construction is used. An example could be '(Adjective Adjective S) → (S) (Adjective Adjective)'.

One can see that after some time only the NP- and S-type constructions are used. We can calculate how many times we expect this to be the case by counting the fraction of times that at least one feature or one event type predicate is contained in the topic description respectively given that either of this should be the case. This is 74.8 and 67.3 percent, which are indeed the values around which the curves fluctuate.

In any case, for now the main point is that again we see that the agents evolve and use compositional language.

8.3.3 Agent Level

Finally, and as in the previous chapter, the evolution of the compositionality can also be measured by inspecting the agents' lexicons. Figure 8.10 shows the evolution of this measure. Again one can see that, over time, compositionality invariably increases (except for a small time interval in the very beginning of each experiment.) This is in complete accordance with the findings so far. Recall that there are basically two mechanisms by which new entries are added. First, a new word is proposed by the speaker in the case of uncovered meaning. This gives rise to one new lexical entry. The expected compositionality of such an entry is about 1/2.[2] Second, the hearer will associated the new word with all meanings compatible with the current context. Again taking into account that the average scene length is 5, this gives rise to about 30 entries for the word

[2]This is calculated starting from the fact that the average scene length is 5, and the assumptions that on average a scene contains one agent and 1 patient feature and all 8 topic descriptions have equal probability 1/8 of occurring.

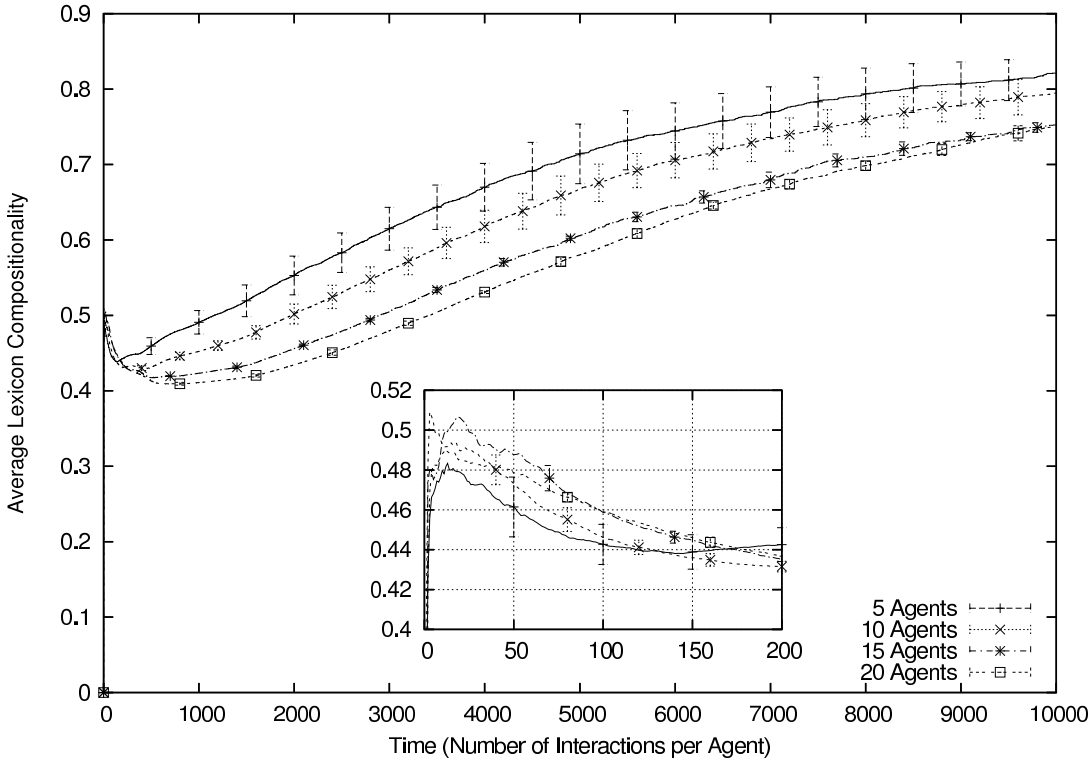


Figure 8.10: Evolution of the compositionality defined as the average lexicon compositionality over all agents (see text.)

and an associated compositionality of 0.4. Hence, on average, and after a very crude calculation, we expect an initial compositionality of about 0.45. This is indeed compatible with figure 8.10.

After the initial phase, words start disappearing again. As was explained in section 8.2.1, the main mechanism that is responsible for this is the reduction of synonymy (lateral inhibition.) This however does not necessarily mean that compositionality increases, only when two holistic words with the same meaning are in competition and one of them looses will the overall compositionality decrease.

There is however another mechanism by which compositionality can increase: whenever a word is observed by a hearer, the probabilities of its associated lexical entries are updated so that they are in accordance with the context. In section 8.2.1 it was suggested that this mechanism was responsible for letting most of the remaining words converge to the same set of meanings and hence put them under the influence of the lateral inhibition dynamics. We can now state with more confidence that this indeed happens and that most words simply become more and more compositional, meaning that they more and more converge to one of a small set of meanings.

This is also in accordance with the previous chapter, where it was found that the cross-situational learning algorithm leads the agents towards absorbing the statistics of the world. In sum, every time a hearer learns a new word, he associates many possible meanings with it. Every next time he hears the word, probability scores are updated such that they are more in accordance with the current context. However, the probability that a holistic meaning is in accordance with the current context is smaller than is the case for compositional meanings. Hence, the meaning of words has a tendency to become compositional because of the interplay between on the one hand the inherent uncertainty about the meaning of a word with which a hearer is confronted and on the other the structure present in the world in the form of single predicates occurring more frequently than specific combinations of predicates.

Conclusion 5. *Language may become compositional when the world is itself compositional and when language learners are confronted with uncertainty about the meaning of words but use the structure in the world to resolve this uncertainty with some sort of cross-situational learning mechanism.*

Furthermore, we can now state with high confidence:

Conclusion 6. *The agents succeed in evolving a successful language that is purely compositional and minimal in the sense that only words of which the meaning contains only one predicate are used. Internally the agents do remember a number of other words but these don't have a clear meaning attached to them.*

These words do not disappear because they are never used and because their meanings do not overlap with the meaning of any of the words that are used, i.e. they must be holistic.

8.4 The Grammar

So far we have been able to answer our first question, namely whether compositionality emerges as a side effect of optimizing communicative success: it does. Because of the assumptions made in the experiment we can also conclude that the agents succeed in evolving a grammar to support this. The question remains however whether the grammar that evolves is minimal.

8.4.1 Constructicon Size and Score

Figure 8.11 shows the evolution of the average number of semantic patterns covered by a grammatical construction. (The actual number of constructions is bigger because one semantic pattern can be associated with several syntactic patterns.) As can be seen, the constructicon reaches its maximum size at about the same time that communication is successful ($t_p \simeq 500$.) Successful communication does indeed require a stable set of grammatical constructions because it is compositional.

Recall that a minimal grammar only requires two constructions (see section 5.6.3): one transitive (e.g. S $\rightarrow$ NP V NP) and one to combine Adjectives and Noun Phrases into bigger Noun Phrases. In contrast, the agents at least *remember* more than two constructions, and more in bigger populations.

Looking at figure 8.12, one can see that all constructions remembered also have almost maximum preference score. Almost, but not all. Because of the learning rules, and because after $t_p \simeq 500$ communication is successful, all constructions that are used get a maximum preference score. This means that some of the constructions still in the constructicon are not used anymore. Still, there is the possibility that a certain number of 'holistic' constructions (instead of only a minimal set of) is used as well.

The main difference with the lexical case is that it seems as if once a semantic pattern is expressed by a construction it remains in the constructicon. This is because there is no competition or lateral inhibition between constructions with different semantic patterns (constructional homonyms), only between so called constructional synonyms (constructions with the same semantic but different syntactic patterns.) In the next section we'll see that constructional synonyms do disappear.

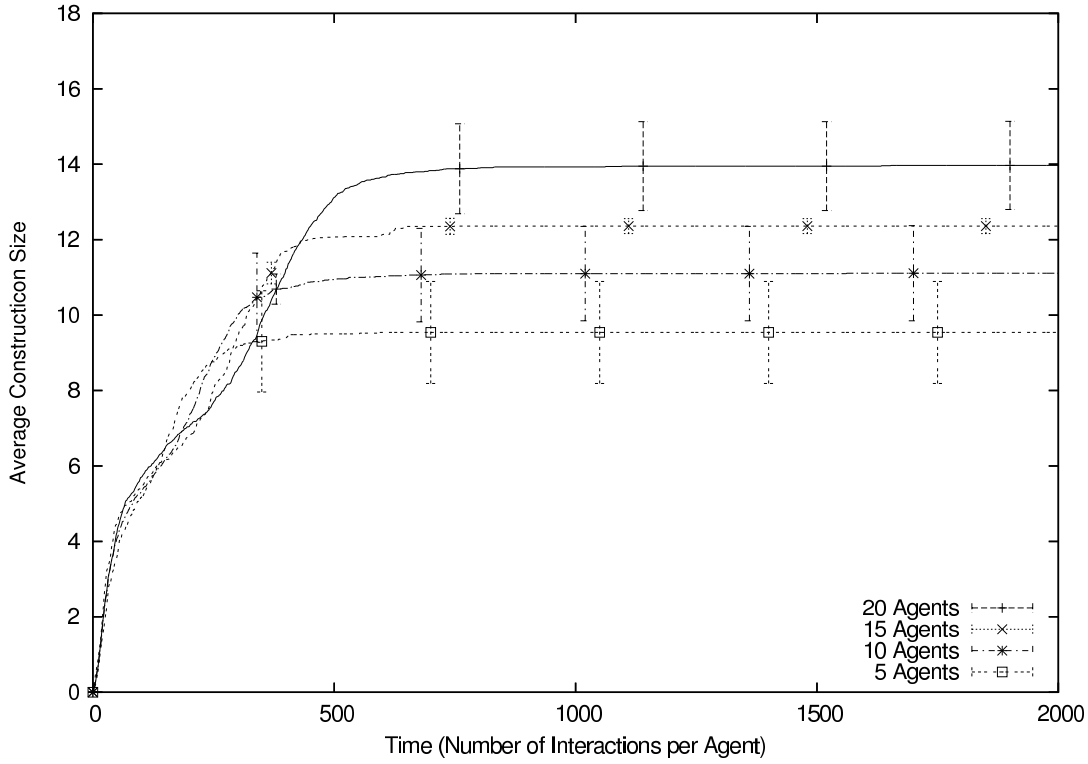


Figure 8.11: Evolution of the constructicon size for different population sizes. The size represents the *actual* size, i.e. also those constructions are counted that are not used anymore but still reside in the individual agents' constructicons. (All graphs are averages of 10 independent runs, $n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 1$.)

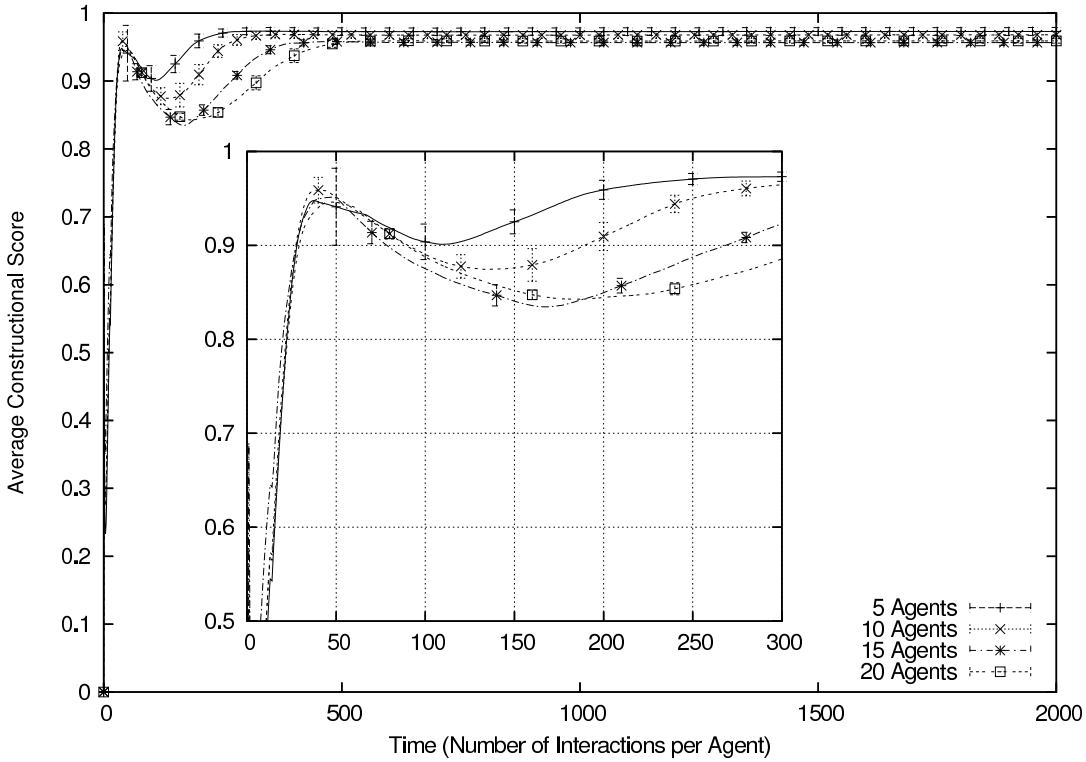


Figure 8.12: Evolution of the average constructional preference score for different population sizes. (All graphs are averages of 10 independent runs, $n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 25$.)

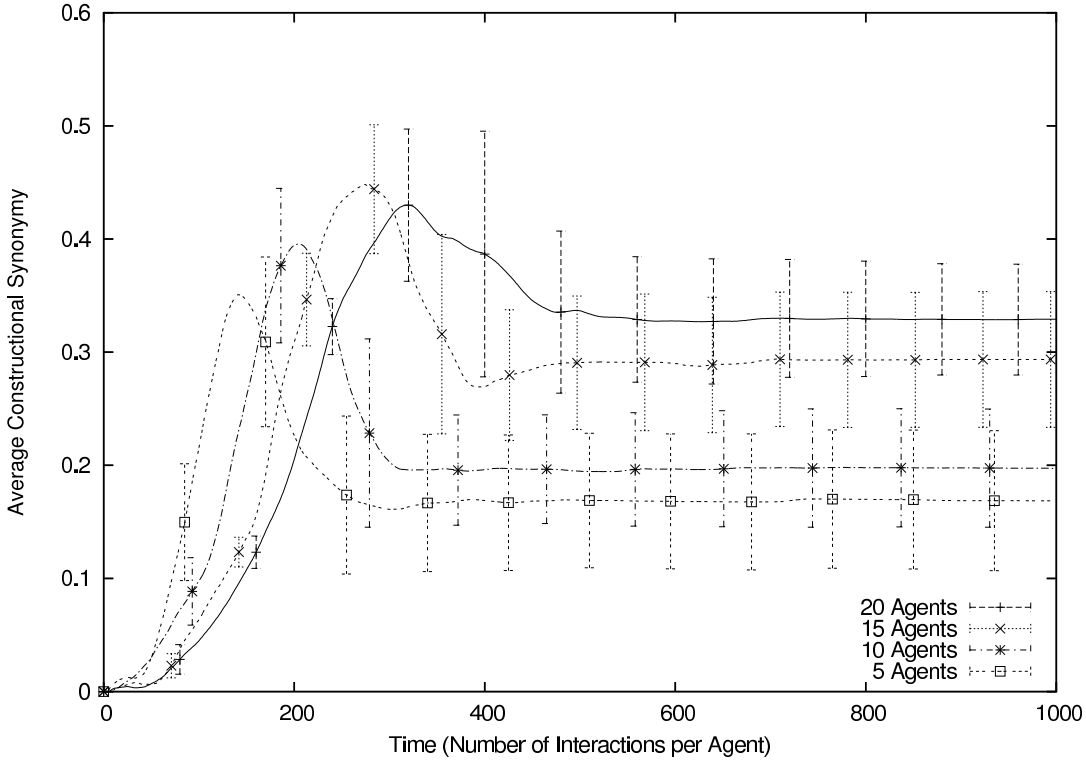


Figure 8.13: Evolution of the constructional synonymy for different population sizes. (All graphs are averages of 10 independent runs, $n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 10$.)

8.4.2 Constructional Synonymy

Figure 8.13 shows the evolution over time of what we have called the constructional synonymy. It shows the average number of syntactic patterns (like 'Adjective NP' or 'NP Adjective') associated with different semantic patterns (like '`(feature ?x) (participant ?x)`'), calculated in the same way as synonymy was calculated for lexical constructions but using pattern scores instead of probability scores.[3]

As for lexical constructions, the agents first go trough a phase of exploration in which many different syntactic patterns are proposed. After reaching a peak value that seems to be independent of the population size, synonymy decreases again, meaning that the agents are starting to reach a consensus about which syntactic patterns to use. Interestingly, and just before communicative success

[3]Note that there is no sense in calculating the homonymy for grammatical constructions since it is not possible that two different semantic patterns are linked to the same syntactic pattern and hence the constructional homonymy is always 0.

becomes optimal, synonymy rises again somewhat before it finally stabilizes.

This is also the main difference with the lexical case: instead of decreasing until it vanishes, synonymy converges to a finite positive value which is bigger for bigger populations. This means that the agents, even after full convergence, are left with a number of semantic patterns for which no single syntactic pattern is chosen. In accordance with the findings in the previous section, this can only mean that some of the semantic patterns are never used anymore (If they were used then their pattern scores would continue changing.)

8.4.3 Number of Constituents

From the previous we still were not able to check whether the resulting grammars are recursive. One way to test this is to count the number of constituents in the grammatical constructions actually used. Again, we'll make a distinction between three different types of constructions: S-type, NP-type and O(ther)-type constructions (see section 8.3.)

Let's define the 'length' of a grammatical construction as the number of syntactic types the construction selects for in its syntactic pole. For example, all of the poles shown below have length 3, and all were taken from an actual simulation with 5 agents:

```
((?top (syn-subunits (== ?unit-1 ?unit-2))
       (form (== (meets ?unit-1 ?unit-2))))
 (?unit-1 (syn-cat (== (Adjective ?x) (Adjective ?x))))
 (?unit-2 (syn-cat (== (NP ?x))))
 ((J ?new ?top)
          (syn-cat ((NP ?x)))))                              (1)

((?top (syn-subunits (== ?unit-1 ?unit-2))
       (form (== (meets ?unit-1 ?unit-2))))
 (?unit-1 (syn-cat (== (S ?x ?y ?z))))
 (?unit-2 (syn-cat (== (Adjective ?z) (Adjective ?z))))
 ((J ?new ?top)
          (syn-cat ((S ?x ?y ?z)
                    (Adjective ?z) (Adjective ?z)))))        (2)

((?top (syn-subunits (== ?unit-1 ?unit-2))
       (form (== (meets ?unit-1 ?unit-2))))
 (?unit-1 (syn-cat (== (Verb ?x ?y ?z) (NP ?z))))
 (?unit-2 (syn-cat (== (NP ?y))))
 ((J ?new ?top)
          (syn-cat ((S ?x ?y ?z)))))                         (3)
```

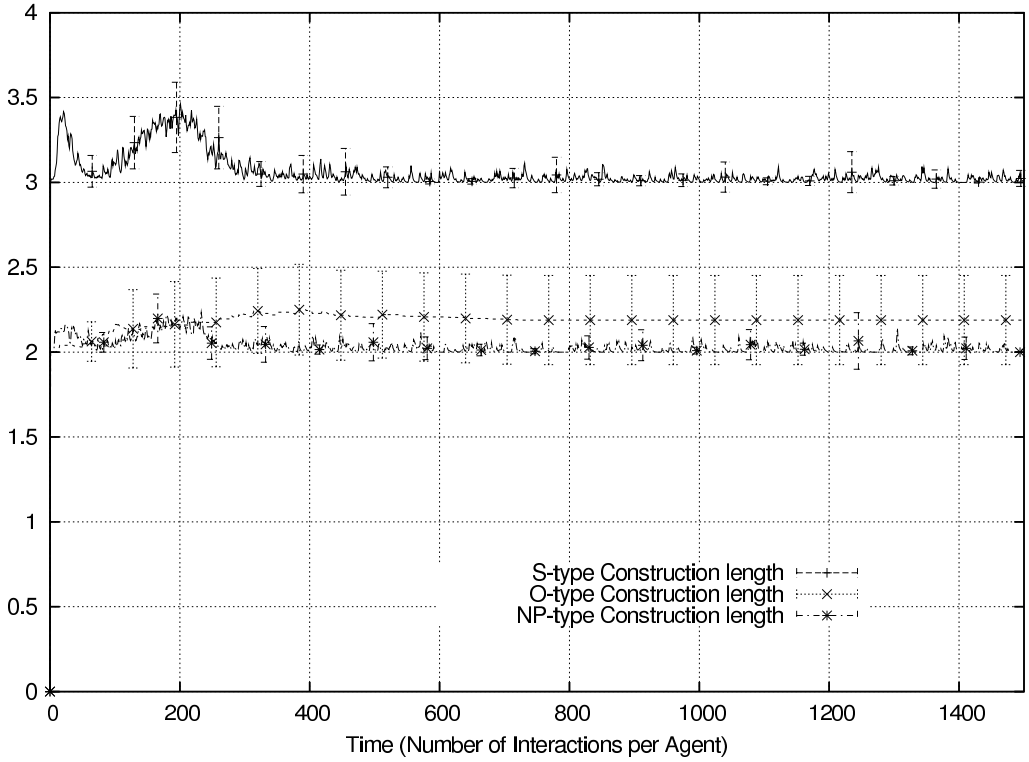


Figure 8.14: Evolution of the number of constituents of grammatical constructions of different result type in the case of 5 Agents. (All graphs are averages of 10 independent runs, $n_e = n_p = n_f = 5$, $\theta = \alpha = 0.2$, $C = 1$.)

```
((?top (syn-subunits (== ?unit-1 ?unit-2 ?unit-3))
       (form (== (meets ?unit-1 ?unit-2)
                 (meets ?unit-2 ?unit-3))))
 (?unit-1 (syn-cat (== (Verb ?x ?y ?z))))
 (?unit-2 (syn-cat (== (NP ?y))))
 (?unit-3 (syn-cat (== (NP ?z))))
 ((J ?new ?top)
         (syn-cat ((S ?x ?y ?z)))))                              (4)
```

Pole (1) is of NP-type, poles (3) and (4) of S-type and pole (2) is of O-type.

Only for the minimal recursive solution do we expect the length of S-type constructions to be 3 and that of NP-type constructions to be 2. Figures 8.14 and 8.15 show the evolution of these lengths in the case of 5, 10 and 15 agents and measured as a running average with $\tau_p = 1$ (see beginning of section 8 for an explanation on how running averages are measured.) The O-type curve is not of much interest since O-type constructions are not used after $t_p \simeq 600$

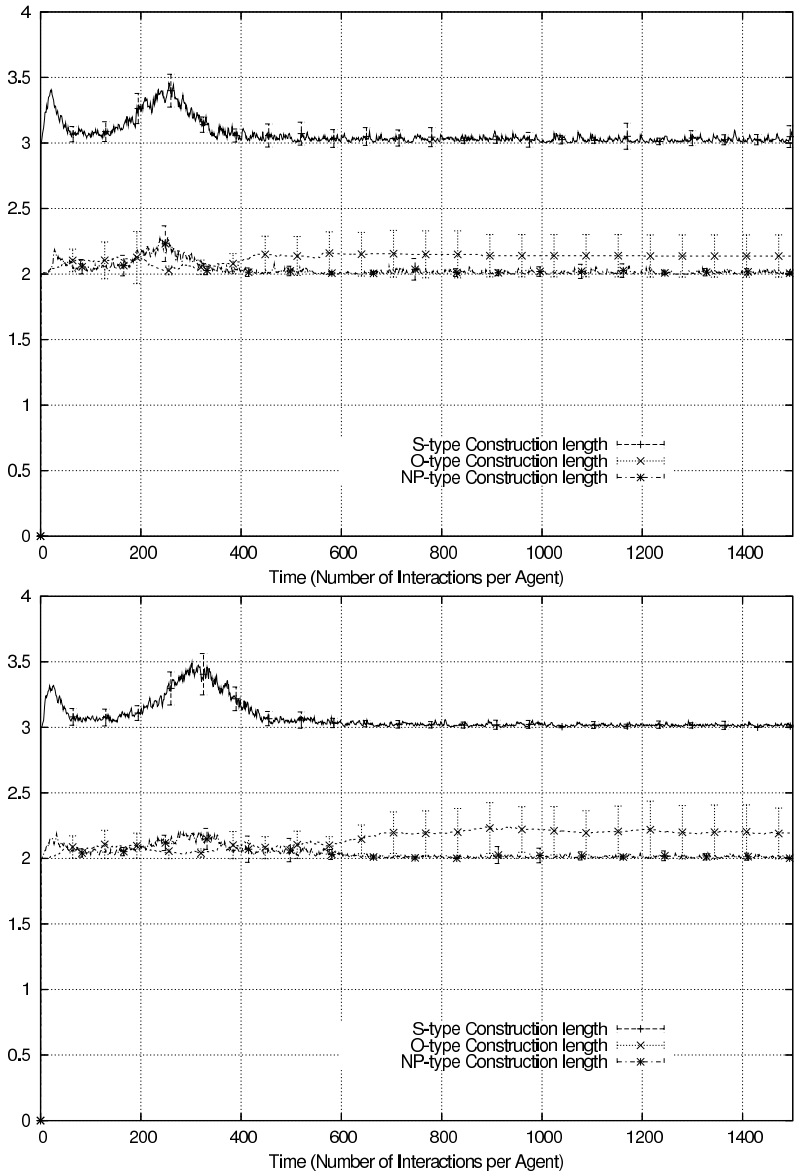


Figure 8.15: Same as Figure 8.14 but for 10 (top) and 15 (bottom) agents.

(see figures 8.8 and 8.9.) The other type of constructions *are* used whenever an equality of the appropriate type needs to be expressed.

As a first approximation, we can conclude that almost all speakers use S-type constructions of length 3 and NP-type constructions of length 2 almost all of the time. Otherwise the curves wouldn't stay so close to those values.

The small fluctuations can be explained by the fact that a minority of the agents probably uses more holistic constructions some of the time. However, because communicative success is maximal, these have to be compatible with the minimal set of recursive constructions used by the other agents. Hence:

Conclusion 7. *Internally, a minority of the agents continues to use holistic constructions. However, these are not distinguishable from the constructions needed in the minimal recursive solution used by almost all of the agents almost all the time. Hence, it is safe to say that the solution found by the agents is the minimal recursive one.*

Chapter 9

Discussion and Conclusions

Compositionality, hierarchy and recursion are universal features of language, meaning that every language in the world exhibits them in one or another way. This observation naturally rises the question of how and why this came to be the case.

The question of why is probably most easily answered. Because they are productive in some sense, and because they introduce regularities in a language, they make a language easier to learn and increase its *usability* (expressive power) and *effectiveness*. Hence, they may increase a language's fitness as well as that of individual language users. The question remains then at what level and by what mechanism this increased fitness is selected for.

At least three levels of selection are possible, each acting on a different time scale and leading to three different explanations: Nativism, iterated learning and functional emergentism.

Nativism focuses on the universality property. They argue that if all languages share these features, then this is because all languages are the product of humans. Hence, these features must be encoded in the human genome and selected for at a biological level.

It was already explained in the introduction however that it is not a trivial task to explain how language, being primarily a *social* phenomenon, could be the result of natural selection. And the few attempts that have been performed so far give rather negative results (see e.g. Steels and Belpaeme (2005).) In addition, no nativist theory was as yet presented that satisfactory fills in the details of what exactly it is that is encoded in our genes or implemented in our brains and could explain why all languages are compositional, hierarchical and recursive. Functional emergentism does not solve this problem. It too presupposes e.g. the capacity for recursion. One of the differences is however that this capacity needn't be language specific. Therefore the problem of how it could have been evolved specifically for language is avoided. More importantly,

functional emergentism also does not require that this capacity *dictates* modern languages to be compositional. For example, as was shown in Chapter 7, holistic languages (which are necessarily not recursive because they require no grammar) can also emerge. This is also in line with what can be observed in natural languages. What *is* assumed is that this capacity can be *recruited* for language if this turns out to be useful (see also Steels (2006) on the recruitment theory of language origins.)

Iterated learning instead sees learnability as the core property, and regards universality and expressive power as side effects. All languages in the world are the product of an iterated learning process. This process evidently favors more learnable languages over less learnable ones, and therefore can effectively explain the universality of a productive feature of language like compositionality. However, by basically ignoring the function of language and the usability aspect of e.g. compositionality, iterated learning is mistakenly seen as the primary reason for why it emerges. Functional emergentism does not claim that multi-generational mechanisms like iterated learning have no effect on language. For example, in section 7.4 it was shown that a population turnover does have an influence on e.g. the degree of compositionality of a language. However, the effect is only of second order, merely acting *on top of* the first order dynamics governed by usability considerations.

Finally, the answer advocated in this thesis and which was dubbed *functional emergentism* focusses on the usability aspect. If a feature of language is productive and hence allows for more effective communication, than individual language users will prefer to use it over less effective means of communication.

The effectiveness of a language feature can of course not be isolated from its learnability and the fact that the entire language community should agree to use it. However, this is precisely one of the points made by functional emergentism: Language should be studied as a complex adaptive system involving multiple and interacting language users, each using the language for a purpose. As explained, other differences lie in the time scale at which selection happens (already in individual interactions), the mechanisms by which selection is performed (in grounded and situated interactions) and, most importantly, the primary reason why selection is performed (communicative effectiveness instead of learnability or e.g. mating advantage.)

Hence, the main claim made in this thesis is that the universality of compositionality, hierarchy and recursion should primarily be explained as being an emergent property of the complex dynamics governing the establishment and evolution of a language in a population of language users on an intra-generational time scale, rather than being the result of a genetic selection process leading to a specialized language faculty that imposes those features upon language or than being a multi-generational cultural phenomenon focussing on

learnability. Productive features of language become prominent because they allow language users and learners to re-use already established bits of language and hence to increase their (expected) communicative success.

We have supported this claim with a computational language game experiment in the spirit of functional emergentism. It was shown that recursive (and hence compositional and hierarchical language) indeed emerges. We have also confirmed the intuition that multi-generational mechanisms like iterated learning play a role as a kind of second order effect. this means that, while being important to investigate e.g. frequency effects, their role in the emergence of compositionality is minor.

To set up the experiments, we could build upon a great deal of experience and insights gained in previous experiments as well as recent theoretical results. Still, most of the solid results so far have been obtained for the emergence of simple lexicons and for single-word utterances. Although there have been a number of experiments on compositionality and the role of syntax and grammar, there are as yet only very few demonstrations where non-trivial grammars arise.

Part of the reason is of course that the problem of compositionality and grammar emergence is much more encompassing than that of simple lexicons and many fundamental questions remain unanswered. But another reason, we believe, has to do with the nature of the computational apparatus that is required to do serious systematic experiments. Whereas lexicon emergence for single word utterances can be studied relatively easy, compositionality and grammar requires much more powerful techniques from machine learning and symbolic processing.

Therefore a cross-situational learning mechanism was developed that would enable the agents to solve the difficult task with which they are faced. We have shown the effectiveness of the algorithm by running a number of simulations of different kinds. It was already shown in (De Beule et al., 2006) that the algorithm enables a language learner to learn the meaning of words using only a very limited amount of information and even if the learning target changes over time. In this thesis we have shown how it can also be used effectively when utterances contain more than one word. In addition, we have shown in Chapter 7 that, when used in a multi-agent setting, it enables a population of agents bootstrapping a language to make use of and absorb any structure that is found in the world, e.g. in the form of correlations between the parts of meaning that need to be expressed, something we consider as an important property.

Next, Fluid Construction Grammar (FCG), a full-fledged grammar formalism, was developed to support and enable the research on the emergence and evolution of grammatical language in a systematic way. It's importance and adequacy is only now becoming visible as there is a growing body of publications reporting on experiments that were performed using FCG (see the experiments

in this thesis and e.g. De Beule and Bergen (2006); Steels and Wellens (2006); Steels and Loetzsch (2007); Van Trijp (2008); Bleys (2008); De Beule (2008))

The main contributions to FCG lie in the implementation and formalization of it. In terms of scientific contributions, a number of important and now widely used special operators were added to FCG. These include extensions to the includes operator (==) and for example the TAG directive introduced in section 4.3.1 enabling a form of inheritance in FCG. The most important operator introduced was the J-operator (De Beule and Steels, 2005). It extends the power of FCG significantly by introducing the possibility to cope with hierarchy and recursion. On top of that, it was shown how it also can be used to solve a number of other imperfections of FCG in an elegant way.

The development of and the contributions made to FCG are by themselves important contributions to the field. But in addition they allowed us to do the experiments required for this thesis. It should be kept in mind however that these make a number of assumptions not generally made in FCG, including the choice of world model, the existence of a universal and shared ontology, a predefined set of semantic and syntactic categories and the specific choice of lexical and grammatical constructions allowed. An important assumption and one that is not made in most language game experiments is the un-awareness of intended word boundaries by the hearer.

These assumptions were made for two reasons. First, and as is done in every experiment, by making the assumptions explicit, the problem that is being investigated is precisely defined and it should become clear what results are, or are not, due to them. At the same time, some of the assumptions were also made to make the problem more tractable. Care was taken however that no bias was introduced that could undermine the point being made. For example, from the baseline experiments in Chapter 7 we can conclude that the results concerning the emergence of compositionality in general and those of the main experiment in particular are valid. To conclude, these results are summarized below:

1. The Learning mechanisms proposed in this thesis are sufficient for a population of agents to bootstrap a successful language (at least up to population sizes of $n_a = 20$ agents.)

2. Although internally the agents remember a number of holistic words, these words are never used and hence are not observable from the outside. A flux in the population hence has the effect of these words really disappearing (because they are never observed), but such a flux is not necessary for explaining the emergence of compositionality.

3. Population size has no influence on the rate at which compositionality

emerges.

4. A minority of the agents continues to use holistic constructions. However, these are not distinguishable from the constructions needed in the minimal recursive solution used by almost all of the agents almost all the time. Hence, it is safe to say that the solution found by the agents is the minimal recursive one.

Most importantly, we have shown that compositional, recursive language can emerge without having to fall back upon genetic arguments or multi-generational mechanisms. The main reasons for this are that:

- Without taking frequency effects into account, holistic words and constructions can only be used on relatively few occasions. This makes it harder for a population of interlocutors to agree upon them.

- Moreover, holistic, idiosyncratic constructs are like one-shot, all-or-nothing messages: one either understands them or not. This makes them potentially very inefficient when it comes to conveying a structured message, especially when language is seen as a complex adaptive system, shaped by all who participate and engage in peer-to-peer dialogues without any central authority or language faculty dictating how to express things. In contrast, compositional constructs are structured themselves. This opens up the possibility of *partially* conveying a structured message instead of all or nothing.

- Language learners are confronted with a fundamental problem, informally called the 'gavagai problem' (Quine, 1960). They have to infer the meaning of unknown words, for example by refining it and eliminating incompatible meanings across situations. When the world is itself compositional in the sense that distinct situations can be conceptualized as being different configurations of the same more basic set of concepts, then language learners will tend towards compositional language because the meaning of words will gradually converge to the set of basic concepts.

The main conclusion is that language should be seen as a complex adaptive system and that the urge for successful communication is a strong shaping force acting on language. We have shown that within this view, it is possible to setup experiments that can explain why the languages of the world share a number of universal features without having to fall back upon genetic arguments. In particular, we have shown that compositionality wins from holism in language because it is productive, re-uses already established bits of language and can be used more often and hence is more *useful* compared to holistic language.

Appendix A

FCG more Formally

A.1 Preliminaries

This appendix provides the more formal discussion of FCG and assumes some familiarity with it (e.g. from Steels (2004), Steels et al. (2005), De Beule and Steels (2005) or the FCG website at `http://arti.vub.ac.be/FCG`). We will adopt the standard terminology of logic, which is unfortunately different from the terminology often used in other unification-based approaches to natural language. In logic, the term 'unification' does not involve the merging of two structures or a change to a structure, so unification is a kind of 'matching' in which it is tested whether a source structure matches a target or pattern structure, but no new resulting structure is constructed.

Example A.1.0.1. *Consider the following syntactic* FCG *unit (*FCG *units will be defined formally later on):*

```
(John-unit (form ((string John-unit "John"))))
```

And the following template unit which could be part of the right-pole of a morphological FCG *template:*

```
(?john-unit (form ((string ?john-unit "John"))))
```

Symbols starting with a question mark, like `?john-unit`*, are variables. The unification of the template unit with the syntactic unit results in the binding [*`?john-unit/John-unit`*]. In contrast, the template unit would not unify with unit below because they do not 'match' (the string value "walks" differs from "John"):*

```
(walks-unit (form ((string walks-unit "walks"))))
```

In contrast with simpler forms of matching (as used in many production rule systems), variables can be present both in the source and in the target. This is the meaning of 'unify' as used in this thesis.

Because we also want a way to extend or build up structure based on templates, we have an additional operator called 'merge' which takes a source structure and a target structure and combines them into a new structure.

Example A.1.0.2. *Consider again the John-unit from the previous example:*

```
(John-unit (form ((string John-unit "John"))))
```

and now also the template unit shown below, which could be the left*-pole of the morphological template from the previous example:*

```
(?john-unit (syn-cat ((lex-cat Proper-Noun))))
```

Given the binding [`?john-unit/John-unit`*], this template unit merges with the John-unit resulting in the following new unit:*

```
(John-unit (form ((string John-unit "John")))
           (syn-cat ((lex-cat Proper-Noun))))
```

We now introduce some definitions which are common in logic (see e.g. Sterling and Shapiro (1986); Agostino et al. (2001)) and which will allow us to formally define the notion of structures, templates, unification and merging. The atomic elements of our formalism are called **Symbols**. For example the template-unit in example A.1.0.1 contains the symbols `?john-unit`, `form`, `string` and `"John"`. Symbols starting with a question mark are **variables**, like `?john-unit`. $\mathcal{A}$ denotes the set of all symbols, $\mathcal{V} \subset \mathcal{A}$ the set of all variables. Symbols that are not variables are **constants.** A **simple expression** is defined as a symbol or a list of one or more simple expressions. More complex expressions can be created with operators which take zero or more argument expressions to produce a new expression. More formally, let (.|.) be the list-creator operator (similar to CONS in LISP Steele (1990b)) and let () be an operator of arity 0 (similar to NIL in LISP).

Definition 3. Let $\mathcal{E}_s$ denote the set of **simple expressions**:

- All elements of $\mathcal{A}$ as well as () are elements of $\mathcal{E}_s$.

- If $e_1 \in \mathcal{E}_s$ and $e_2 \in \mathcal{E}_s$ and if e_2 is not in $\mathcal{A}$ then $(e_1|e_2) \in \mathcal{E}_s$.

Hence, all symbols and the syntactic and template units in examples A.1.0.1 and A.1.0.2 are simple expressions. In addition to simple expressions, we will consider later non-simple expressions which feature special operators and will be called general FCG-expressions or simply expressions.

Let us denote the set of variables in an expression e by **vars(e)**. Often an expression of the form $(e_1|(e_2|...(e_k|())...))$ is represented as $(e_1e_2...e_k)$ and is called a **list of length k**. The operator () is called a list of length 0.

A **binding** $[?x/X]$ specifies the value X (a simple expression) of a variable $?x$. If X is itself a variable then such a binding is called an **equality**. A set of bindings $B = \{[?x_1/X_1], ..., [?x_n/X_n]\}$ defines a function $\sigma_B : \mathcal{E}_f \rightarrow \mathcal{E}_s$ such that $\sigma_B(e) = e$ except for the set of variables $?x_1$ to $?x_n$ in B. This set is called the **domain** of σ_B and denoted $\text{dom}(\sigma_B)$. For $?x_i \in \text{dom}(\sigma_B)$ it holds that $\sigma_B(?x_i) = X_i$, $1 \leq i \leq n$. The set $\{X_1, ..., X_n\}$ is called the **range** of σ_B and denoted $\text{ran}(\sigma_B)$. Such a function σ_B is called a **substitution** and the set of bindings B is sometimes called the **graph** of σ_B and is written simply as $[?x_1/X_1, ..., ?x_n/X_n]$. Often a substitution σ_B is represented by its graph B. The empty substitution (the identity function) is denoted by ϵ. Furthermore define *fail* to be a special symbol which will be used to specify the failure to find a substitution. The extension of a substitution to the domain $\mathcal{E}_s$ can be defined using structural induction. If σ_B is the substitution constructed from the set of bindings B then the application of σ_B to an expression e is written either as $\sigma_B(e)$ or as $[e]_B$.

Example A.1.0.3. *The binding* $B =$*[*`?john-unit/John-unit`*] from example A.1.0.1 defines a substitution* σ_B *that leaves all expressions unchanged except for the variable* `?john-unit`*, which is mapped onto the constant* `John-unit`*. Hence:*

σ_B(`?john-unit`)=`John-unit`

and

σ_B(((?john-unit (form ((string ?john-unit "John"))))))
$= \sigma_B$(((John-unit (form ((string John-unit "John"))))))
= ((John-unit (form ((string John-unit "John")))))

The domain of σ_B *is* `{?john-unit}` *and its range is* `{John-unit}`*.*

Definition 4. Two simple expressions x and y are said to be **equal** (written as $x = y$) if and only if either:

1. x and y are both the same atom
2. both $x = (x_1...x_n)$ and $y = (y_1...y_m)$ are lists of the same length $(m = n)$ and for every $i = 1..n : x_i = y_i$.

Substitutions can be ordered according to the pre-order $\preceq_{\mathcal{V}}$ which is defined as follows: If σ_1 and σ_2 are two substitutions then $\sigma_1 \preceq_{\mathcal{V}} \sigma_2$ if and only if there exists a substitution λ such that $\sigma_1(v) = (\sigma_2 \circ \lambda)(v) = \sigma_2(\lambda(v))$ for each $v \in \mathcal{V}$. When clear from the context, $\sigma_1 \preceq_{\mathcal{V}} \sigma_2$ is written as $\sigma_1 \preceq \sigma_2$.

If $\sigma_1 \preceq_{\mathcal{V}} \sigma_2$ and $\sigma_2 \preceq_{\mathcal{V}} \sigma_1$ then we say that $\sigma_1 =_{\mathcal{V}} \sigma_2$.

Definition 5. Two simple expressions e_1 and e_2 are said to **unify** if and only if there exists a substitution σ such that $\sigma(e_1) = \sigma(e_2)$. In this case the substitution σ is called a **unifier** of the two expressions and the set of all unifiers of e_1 and e_2 is written as $U(e_1, e_2)$.

Example A.1.0.4. *From example A.1.0.3 it can be seen that the expressions*

`((?john-unit (form ((string ?john-unit "John")))))`

and

`((John-unit (form ((string John-unit "John")))))`

indeed unify with unifier σ_B*;* $B =$*[*`?john-unit/John-unit`*].*

It is easy to see that the set of unifiers of two unifiable expressions e_1 and e_2 is infinite (if $\mathcal{V}$ is infinite.) Indeed, a unifier can always be extended with an additional binding for a variable that is not an element of either $vars(e_1)$ or $vars(e_2)$. In order to exclude these unifiers we only assume unifiers that satisfy the **protectiveness** condition. A unifier σ of two expressions e_1 and e_2 satisfies this condition if and only if $dom(\sigma) \subseteq vars(e_1) \cup vars(e_2)$ and $dom(\sigma) \cap vars(ran(\sigma)) = \emptyset$

Example A.1.0.5. *Both*

$B =$*[*`?john-unit/John-unit`*]*

and

$B' =$*[*`?john-unit/John-unit,?walks-unit/Walks-unit`*]*

define unifiers of

$e_1 =$`((?john-unit (form ((string ?john-unit "John")))))`

and

$e_2 =$`((John-unit (form ((string John-unit "John")))))`*.*

However, only B satisfies the protectiveness condition. Indeed:

$$\begin{aligned} dom(\sigma_{B'}) &= \{\textit{?john-unit, ?walks-unit}\} \\ &\not\subseteq vars(e_1) \cup vars(e_2) = \{\textit{?john-unit}\} \end{aligned}$$

A **complete set** of unifiers $c(e_1, e_2)$ is a subset of $U(e_1, e_2)$ that satisfies the additional condition that for any unifier σ of e_1 and e_2 there exists a $\theta \in c(e_1, e_2)$ such that $\theta \preceq \sigma$.

The **set of most general unifiers** $\mu(e_1, e_2)$ is a complete set of unifiers that additionally satisfies the **minimality condition**: for any pair $\mu_1, \mu_2 \in \mu(e_1, e_2)$, if $\mu_1 \preceq \mu_2$ then $\mu_1 = \mu_2$.

It is well known that if two simple expressions x and y unify, then there always is only one most general unifier (see e.g. Sterling and Shapiro (1986)). In example A.1.0.5 the bindings B define the unique most general unifier of e_1 and e_2.

Let $f_\mu(x, y, \epsilon)$ be a function that computes the most general unifier of two simple expressions x and y if one exists and returns *fail* otherwise:

$$f_\mu(x, y, \epsilon) = \begin{cases} \mu(x, y) & \text{if } x \text{ and } y \text{ unify} \\ \mathit{fail} & \text{otherwise} \end{cases}$$

We will now define $f_\mu(x, y, B)$, with B a non-empty substitution, which will allow us to show how $f_\mu(x, y, \epsilon)$ can be computed. For this we need the notion of valid extension. Let B be a set of bindings and let $X \in \mathcal{E}_s$ be a simple expression. If it exists, then the valid extension $\Xi(B, [?x/X])$ of B with $[?x/X]$ is a substitution that has $?x$ in its domain: $?x \in \mathrm{dom}(\sigma_{\Xi(B,[?x/X])})$. For example, if $X \in \mathcal{C}$ and $?x \notin \mathrm{dom}(\sigma_B)$, that is if X is a constant and $?x$ is not yet bound in B, then B is simply extended to include the binding $[?x/X]$:

$$X \in \mathcal{C} \text{ and } ?x \notin \mathrm{dom}(\sigma_B): \quad \Xi(B, [?x/X]) \equiv B \cup [?x/X]$$

However, if B already specifies a value for $?x$ then this might conflict with the new value X for $?x$ as specified by $[?x/X]$. The following two definitions takes care of this (and of other possible conflicts). Note that both definitions depend on each other, but in such a way that both functions are always computable.

Definition 6. The **valid extension** $\Xi(B, b)$ of a set of bindings $B = [?x_1/X_1, ..., ?x_n/X_n]$ with a binding $b = [?x/X]$ (with $X, X_i \in \mathcal{E}_s$) is defined as follows:

$$\Xi(B, b) = \begin{cases} f_\mu(X, X_i, B) & \text{if } ?x = ?x_i \text{ for some } i \in \{1, ..., n\} \\ f_\mu(?x, X_j, B) & \text{if } X = ?x_j \text{ for some } j \in \{1, ..., n\} \\ \mathit{fail} & \text{if } ?x \text{ occurs anywhere in } X \text{ or in } \sigma_B(X) \\ B \cup [?x/X] & \text{otherwise.} \end{cases}$$

Definition 7. Let $x, y \in \mathcal{E}_s$ and B be a set of bindings or *fail*. Then

$$f_\mu(x, y, B) = \begin{cases} \mathit{fail} & \text{if } B = \text{fail} \\ B & \text{if } x = y \\ \Xi(x, y, B) & \text{if } x \in \mathcal{V} \\ \Xi(y, x, B) & \text{if } y \in \mathcal{V} \\ f_\mu(x_r, y_r, f_\mu(x_1, y_1, B)) & \text{if } x = (x_1|x_r) \text{ and } y = (y_1|y_r) \\ \mathit{fail} & \text{otherwise} \end{cases}$$

With these definition the standard unification function $f_\mu(x, y, \epsilon)$ on simple expression x and y is also defined. As will be shown in the next section, FCG unification $f_{\mathrm{FCG}}(x, y, \{\epsilon\}) = f_\mu(x, y, \epsilon)$ if both x and y are simple expressions.

A.2 Unifying

General FCG expressions are more extensive than simple expressions because they may include special operators such as the includes-operator '==' which specifies which elements should be included in a feature's value, or the J-operator which plays a key role in defining hierarchy.[1] Each of these operators has a dedicated function for defining how unification should be carried out.

Example A.2.0.6. *Consider the following source syntactic unit*

```
(walks-unit (form ((stem walks-unit "walk")
                   (affix walks-unit "-s"))))
```

Now assume we want to build a template unit that is capable of 'recognizing' all present tense forms of the verb "to walk", independent of person (i.e. indendent of the affix "-s"). As a first attempt we could write:

```
(?walk-unit (form ((stem ?walk-unit "walk"))))
```

Unfortunately, this template unit does not unify with (or 'recognize') the source unit because it does not contain the affix part. The FCG *solution to this is to use the includes special operator, written as* `==`*, in the template unit as follows:*

```
(?walk-unit (form (== (stem ?walk-unit "walk"))))
```

An includes list (like the `(== (stem ?walk-unit "walk")` *part above) unifies with all lists that* at least *contain the specified elements, but more elements may be present in the source. As a result the above modified template unit does unify with (or 'recognizes') the source unit as was desired.*

We will now formalize this. Let $\mathcal{O}$ be the set of special operator symbols. It includes for example the symbol '=='.

Definition 8. Let $\mathcal{E}$ denote the set of all FCG **expressions**. Then:

1. All elements of $\mathcal{A}$ and $\mathcal{O}$ as well as () are elements of $\mathcal{E}$.
2. if $e_1 \in \mathcal{E}$ and $e_2 \in \mathcal{E}$ and if e_2 is not in $\mathcal{A} \cup \mathcal{O}$ then $(e_1|e_2) \in \mathcal{E}$.

[1] The J-operator is treated in De Beule and Steels (2005).

In words, the set of all FCG expressions consists of all (nested) lists of which the leaf elements may be symbols but also special operators. The modified template unit in example A.2.0.6 is thus an FCG expression. It is however not a simple expression because it contains the includes special operator. In the following we will refer to an FCG expression simply as an expression.

To extend the notion of unification to the set of all expressions we must specify how special operators are handled. Therefore, for every operator o in $\mathcal{O}$ and for all expressions $e_1 = (o|e_1')$ and e_2 and sets of bindings $\mathcal{B}$ a designated unification function $f_o(e_1, e_2, \mathcal{B})$ must be defined returning a set of unifiers for e_1 and e_2. FCG unification can then be defined as follows:

Definition 9. FCG **unification** $f_{\text{FCG}}(x, y, \mathcal{B})$ of two expressions x and y, with $\mathcal{B}$ a set of sets of bindings or *fail* is defined as follows:

1. $f_{\text{FCG}}(x, y, \text{fail}) = \text{fail}$
2. if $x = (o|x')$ with $o \in \mathcal{O}$ then $f_{\text{FCG}}(x, y, \mathcal{B}) = f_o(x, y, \mathcal{B})$
3. else if $y = (o|y')$ with $o \in \mathcal{O}$ then $f_{\text{FCG}}(x, y, \mathcal{B}) = f_o(y, x, \mathcal{B})$
4. else if $x = (x_1|x_r)$ and $y = (y_1|y_r)$ then $f_{\text{FCG}}(x, y, \mathcal{B}) = f_{\text{FCG}}(x_r, y_r, f_{\text{FCG}}(x_1, y_1, \mathcal{B}))$
5. else $f_{\text{FCG}}(x, y, \mathcal{B}) = \{B' | B' = f_\mu(x, y, B); B' \neq \text{fail}; B \in \mathcal{B}\}$

It is easy to see that if x and y are simple expressions (i.e. do not contain special operators) then $f_{\text{FCG}}(x, y, \{\epsilon\}) = \{f_\mu(x, y, \epsilon)\}$. In this sense FCG-unification is equivalent to standard unification in the space of simple expressions. For general FCG-expressions the properties of f_{FCG} will depend on the properties of the dedicated unification functions f_o. We now define some of these special operators so that FCG unification of FCG feature structures can be defined. The list of special operators can in principle be extended by defining the relevant unification functions.

A.2.1 The includes operator ==

Let us first define the notion of containment.

Definition 10. An expression x_i is **contained** in a list if and only if it FCG-unifies with an element in this list.

A list starting with the **includes operator** (== $x_1 ... x_n$) unifies with any source list that at least contains the elements x_1 to x_n. The order in which the elements occur in the source list is irrelevant, however every x_i should unify with a *different* element in the source as in the following examples:

Example A.2.1.1.

$$
\begin{aligned}
&f_{==}((==\ a\ a\ b), (a\ b), \{\epsilon\}) = \text{fail}\\
&f_{==}((==\ a\ a\ b), (a\ a\ b), \{\epsilon\}) = \{\epsilon\}\\
&f_{==}((==\ a\ a\ b), (b\ a\ a), \{\epsilon\}) = \{\epsilon\}\\
&f_{==}((==\ a\ ?x), (a\ b\ c), \{\epsilon\}) = \{[?x/b], [?x/c]\} \qquad \text{(A.1)}
\end{aligned}
$$

This is formalized as follows:

Definition 11. Let $p_n((e_1...e_m))$ with $n \leq m$ be the set of expressions $(e_{i_1}...e_{i_n})$ for every variation $(i_1, ..., i_n)$ of n elements out of $(1, ..., m)$.[2] Then

$$
\begin{aligned}
&f_{==}((== x_1...x_n), (a_1...a_m), \mathcal{B})\\
&\qquad \equiv \{B | B = f_{\text{FCG}}((x_1...x_n), a, \mathcal{B}); B \neq \mathit{fail}; a \in p_n((a_1...a_m))\}
\end{aligned}
$$

Example A.2.1.2. *Consider again the last example in (A.1). We have to consider all variations of two elements out of* $(a\ b\ c)$*, i.e.* $(a\ b)$*,* $(b\ a)$*,* $(a\ c)$*,* $(c\ a)$*,* $(b\ c)$ *and* $(c\ b)$*. Unifying these with* $(a\ ?x)$ *results in* $\{[?x/b]\}$*,* fail*,* $\{[?x/c]\}$*,* fail*,* fail *and* fail *respectively. Keeping only the successful ones indeed leads to* $\{[?x/b], [?x/c]\}$*.*

A.2.2 Special cases of $f_{==}$

First of all, the unification of two include-lists $x = (==\ x_1...x_n)$ and $y = (== y_1...y_m)$ is not well defined. One possibility is to state that two such lists always unify. However, it is not clear what the resulting set of bindings should be. For simplicity we define $f_{\text{FCG}}((o_1|x), (o_2|y), \mathcal{B})$ with $o_1, o_2 \in \mathcal{O}$ to *always be equal to fail.*

Second, the unification of a pattern $(x_1...x_k == y_1...y_l)$ with a source $(z_1...z_m)$, $m \geq k + l$, is well defined. More formally, the pattern should be written as

$$(x_1...x_k == y_1...y_l) \equiv (x_1|(...|(x_k|(== |(y_1|(...|(y_l|())...).$$

The f_{FCG} function will progressively unify the elements x_1 to x_k with the first k elements in the source. At this point f_{FCG} is recursively applied to the pattern $(== |(y_1|(...|(y_l|())...)$, which can be re-written as $(== y_1...y_l)$, and the source $(z_{k+1}...z_m)$.

Third, operators are treated as ordinary symbols if they are not the first element of a list:

$$f_{==}((==\ a\ ?x), (a\ ==\ b), \{\epsilon\}) = \{[?x/ ==], [?x/b]\}$$

[2] We intend here the set-theoretic notion of variation, i.e. all subsets of n elements from $(1, ..., m)$ where the order of the elements matters.

A.2.3 Minimality, Completeness and Complexity of $f_{==}$

Theorem A.2.1 (Completeness of $f_{==}$)**.** *The function* $f_{==}((== |X), Y, \{\epsilon\})$ *computes a complete set* $c((== |X), Y)$ *of unifiers*

Proof. From the definition of $f_{==}$ it follows that, if a substitution τ is a unifier of $(== |X) = (== x_1...x_n)$ and $Y = (y_1...y_m)$, then it must be that $\tau(X)$ is a variation of n elements from Y. Let $\sigma \in f_{==}((== |X), Y, \{\epsilon\})$ be the substitution that computes this variation, so that $\sigma(X) = \tau(X)$. If X and Y do not contain any special operators then, from the completeness of F_{FCG} for simple expressions, it follows that $\sigma \preceq \tau$ and thus that $f_{==}(X, Y, \{\epsilon\})$ computes a complete set of unifiers. If some element x_i (y_i) of X (Y) is of the form $(== |z)$, with z not containing a special operator, then the same argument can be used recursively. Continuing in this way, it follows that $f_{==}$ computes a complete set of unifiers. □ □

Theorem A.2.2 (Non-minimality of $f_{==}$)**.** *The function* $f_{==}((== |X), Y, \{\epsilon\})$ *does not necessarily compute the most general set of unifiers* $\mu((== |X), Y)$.

Proof. It suffices to show that $f_{==}$ does not satisfy the minimality condition. Consider the unification of $(== ?x\ ?y)$ with $(?x\ ?y)$ which results in $\{\epsilon, [?x/?y]\}$. This is different from the minimal set of unifiers which consists of the empty substitution ϵ only (this example was taken from nez (2004).) □ □

Theorem A.2.3 (Complexity of $f_{==}$)**.** $f_{==}((== x_1...x_n), (y_1...y_{m \geq n}))$ *is of exponential complexity in* n.

Proof. Basically, the exponential complexity arises from the need to calculate the variations. Indeed, when $x_i \neq y_j$ for all possible combinations of i and j then $f_{==}((== x_1...x_n), (y_1...y_m))$ is equivalent with the subset unification of $\{x_i\}$ and $\{y_j\}$. General subset unification is exponential and the subset-unifiability problem is NP-complete Kapur and Narendran (1986); Degyarev and Voronkov (1994). □ □

The implementation of $f_{==}((== x_1...x_n), (y_1...y_{m \geq n})$ can be made more efficient by calculating in advance the set of candidate expressions $C_i = \{y_j | y_j$ unifies with $x_i\}$ and by only considering combinations of n distinct elements out of each C_i. However the inherent exponential complexity cannot be improved upon in general.

In the following we introduce two additional special operators which are defined as special cases of the includes operator. This ensures that the completeness of f_{FCG} is maintained. However they can be computed more efficiently. The need for introducing these operators will become more clear in sections A.3 and A.5.

A.2.4 The permutation operator $==_p$

The permutation operator is like the includes operator except that the source should contain *exactly* the elements specified in the pattern. Thus we have:

Definition 12.

$$f_{==_p}((==_p x_1...x_n),(a_1...a_m),\mathcal{B}) \equiv \begin{cases} f_{==}((== x_1...x_n),(a_1...a_m),\mathcal{B}) & \text{if } n=m, \\ \mathit{fail} & \text{otherwise.} \end{cases} \quad \text{(A.2)}$$

A.2.5 The includes-uniquely operator $==_1$

The function of this operator will become more important in merging (see later.) However its behavior in unification must be specified because FCG-templates may contain this operator.

Definition 13. Let $s=(y_1...y_m)$. Then $f_{==_1}((==_1 x_1...x_n),s)$ is the set $\{B\} \subset f_{==}((== x_1...x_n),s)$ of substitutions B that satisfy the following conditions

1. No two symbols $\sigma_B(y_i)$ and $\sigma_B(y_j)$ of $\sigma_B((y_1...y_n))$ with $i \neq j$ are allowed to unify: $f_{\mathrm{FCG}}(y_i,y_j,\{B\}) = \mathit{fail}$ and
2. if $\sigma_B(y_i) = \sigma_B((y_{i1}|y_{i2}))$ and $\sigma_B(y_j) = \sigma_B((y_{j1}|y_{j2}))$ are two non-atomic elements of $\sigma_B((y_1...y_n))$ with $i \neq j$ then their first elements are not allowed to unify: $f_{\mathrm{FCG}}(y_{i1},y_{j1},\{B\}) = \mathit{fail}$

The above definition ensures that every element in $\sigma_B((y_1...y_n)), B \in \mathcal{B}$ is distinct. It also implies that no element $\sigma_B(y_i)$ can be a variable or start with a variable if $n \geq 1$.

Example A.2.5.1.

$$\begin{aligned} &f_{==_1}((==_1\ ?x_1\ a),(?y_1\ (?y_2)\ b)) = \{[?y_1/a, ?x_1/(?y_2)], [?y_1/a, ?x_1/b]\} \\ &f_{==_1}((==_1\ ?x_1\ a),(?y_1\ ?y_2\ b)) = \text{fail} \\ &f_{==_1}((==_1\ ?x_1),(?y_1\ b)) = \text{fail.} \end{aligned} \quad \text{(A.3)}$$

In contrast with the last of these examples, consider that:

Example A.2.5.2.

$$f_{==}((==\ ?x_1),(?y_1\ b)) = \{[?x_1/b], [?x_1/?y_1]\}. \quad \text{(A.4)}$$

Note that some includes-uniquely patterns cannot be satisfied (e.g. $(==_1$ a $a)$.)

A.3 Unifying Feature Structures

We are now ready to define the matching of two FCG structures. We begin by defining FCG feature structures which are more constrained than general FCG expressions. They are also more constrained than feature structures in other unification grammars Pollard and Sag (1994), in the sense that they are not hierarchical. Hierarchy is represented instead by using the name of a unit as the definition of the syn-subunits or sem-subunits slots. This has many advantages, including that a unit can be the subunit of more than one other unit. It also simplifies computation enormously.

A.3.1 Feature structures in FCG

A syntactic or semantic structure in FCG consists of a set of units which each consist of a unique name and a set of feature-value pairs. A unit typically corresponds to a lexical item or to constituents like noun phrases or relative clauses. The name can be used to identify or refer to a unit and unit-names can be bound to variables. Feature-values cannot themselves be feature structures. Instead, we always introduce separate units with their own names and associate feature-value pairs with this new unit.

Example A.3.1.1. *The following expression could be a syntactic structure in* FCG. *The structure contains three units named* `sentence-unit`, `subject-unit` *and* `predicate-unit`.

The `sentence-unit` *has two features named* `syn-cat` *and* `syn-subunits`, *with values respectively* `(SV-sentence)` *and* `(subject-unit predicate-unit)`, *which are both lists.*

```
((sentence-unit (syn-subunits (subject-unit predicate-unit))
                (syn-cat (SV-sentence)))
 (subject-unit (syn-cat (proper-noun (number singular)))
               (form John))
 (predicate-unit (syn-cat (verb (number singular)))
                 (form walks))).
```

Without separate units or unit names (as in other formalisms) this would look like:

```
(sentence-unit
   (syn-subunits
      ((syn-cat (proper-noun)
                (number singular))
```

```
 (form John))
((syn-cat (verb)
          (number singular))
 (form walks))))).
```

A template's pole has the same form as a feature structures but typically contains variables as well as special operators (like '==').

Example A.3.1.2. *The following expression could be the syntactic pole of a template. Note how agreement in number between subject and verb is handled through the variable* `?number` *which will be bound to a specific number value.*

```
((?sentence-unit
    (syn-cat (SV-sentence))
    (syn-subunits (?subject-unit ?predicate-unit))
    (form (== (precedes ?subject-unit ?predicate-unit))))
 (?subject-unit
    (syn-cat (== proper-noun (number ?number))))
 (?predicate-unit
    (syn-cat (== verb (number ?number))))).
```

The features that may occur are restricted to a limited set of symbols: {sem-subunits, referent, meaning, sem-cat} for semantic structures and {syn-subunits, utterance, form, syn-cat} for syntactic structures. The syntactic or semantic categories are completely open-ended, and so the example categories used here (like number, proper-noun, etc.) are just intended as illustration. Syntactic and semantic structures always come in pairs, and units in a syntactic structure are paired with those in the semantic structure through common unit names. More formally, we have the following:

Definition 14. A **feature-value pair** is an expression of the form $(e_n\ e_v)$. The expression e_n is called the feature name and e_v is the feature value. A **unit** is any expression of the form $(e_n\ f_1...f_k)$ with the expression e_n the unit's name and $f_i, i = 1...k$ its features. Unit names are usually but not necessarily symbols. Finally a **unit structure** (or feature structure) is any expression of the form $(u_1...u_l)$ with all of the u_i units.

Thus, a unit structure can be represented by an expression of the form

$$\begin{aligned}&((u_1\ (f_{11}\ v_{11})...(f_{1n_1}\ v_{1n_1}))\\&\quad ...\\&\quad (u_m\ (f_{m1}\ v_{m1})...(f_{mn_m}\ v_{mn_m}))).\end{aligned} \tag{A.5}$$

Here is another (simplistic) example of a syntactic structure for the utterance "red ball":

```
((np-unit (syn-subunits (adjective-unit noun-unit)))
 (adjective-unit (syn-cat (adjective))
                 (form ((stem adjective-unit "red"))))
 (noun-unit (syn-cat (noun))
            (form ((stem noun-unit "ball")))))
```

which may be associated with the following semantic structure:

```
((np-unit (sem-subunits (adjective-unit noun-unit)))
 (adjective-unit (meaning ((color obj-1 red))))
 (noun-unit (referent obj-1)
            (meaning ((sphere obj-1)(used-for obj-1 play)
                      (mentioned-in-discourse obj-1))))).
```

The names of the units allow cross-referencing between the two structures.

Unification in FCG determines the applicability of templates. An example of an FCG template that should be triggered by the above semantic structure is shown below (the template's left (semantic) and right (syntactic) poles are separated by a double-arrow):[3]

```
((?unit (referent ?obj)
        (meaning (== (sphere ?obj) (used-for ?obj play)))))
<-->
((?unit (form (== (stem ?unit "ball")))))
```

However, it can be seen that the left pole of this template does *not* unify with the semantic structure above: only (part of) the `noun-unit` is specified by the pole but specifications for the other units are missing. Therefore no substitution can make the source structure equal to the pole or vice versa. If however the pole is changed to:

```
(==1 (?unit (referent ?obj)
            (meaning (== (sphere ?obj)
                         (used-for ?obj play))))),
```

then this indeed unifies with the source structure to yield the bindings

```
[?obj/obj-1,?unit/noun-unit].
```

[3]These examples have all been simplified for didactic reasons.

A.3.2 The unification of feature structures

Definition 15. The function **unify-structures(P,S,B)** takes a pattern structure P, a source structure S and a set of bindings B and can be computed as follows. If P is as represented in (A.5) then the pattern is first transformed to the pattern P':

$$\begin{aligned}(==_1(u_1 \ ==_1 (f_{11} \ v'_{11})...(f_{1n_1} \ v'_{1n_1}))\\ ...\\ (u_m \ ==_1 (f_{m1} \ v'_{m1})...(f_{mn_m} \ v'_{mn_m}))),\end{aligned} \tag{A.6}$$

in which the new feature values v'_{ij} are determined as follows: Every non-atomic feature value $v_{ij} = (v_1|v_2)$ in the pattern for which v_1 is not a special operator is replaced by $v'_{ij} = (==_p |v_{ij})$. Atomic feature values remain unchanged: $v'_{ij} = v_{ij}$ if v_{ij} is atomic. Unify-structures(P,S,B) is then defined as $f_{\mathrm{FCG}}(P', S, \{B\})$.

A.4 Merging

Informally, merging a source expression s and a pattern expression p means changing the source expression such that it unifies with the pattern expression. Merging two general fcg-expressions is undefined, we only consider the case where at least the source is a simple expression (i.e. does not contain special operators.) We first examine the case where also the pattern is a simple expression.

A.4.1 Merging of Simple Expressions

Definition 16. Let $g(p, s, B)$ denote the merge function that computes a set of tuples $(s', \mathcal{B}')$ of new source patterns s' and bindings sets $\mathcal{B}'$ such that $f_{\mathrm{FCG}}(p, s', \{B\}) = \mathcal{B}'$. $g(p, s, B)$ on simple expressions p and s is defined as follows:

1. If $\mathcal{B} = f_{\mathrm{FCG}}(p, s, \{B\}) \neq \mathit{fail}$ then $g(p, s, B) = (s, \mathcal{B})$.
2. Else if $p = (p_1|p_2)$ and $s = (s_1|s_2)$ then let $G'_1 = g(p_1, s_1, B)$.
   (a) If $G'_1 \neq \emptyset$ then

$$g(p, s, B) = \bigcup_{(s'_1, B_1) \in G'_1} \left(\bigcup_{g'_2 \in g(p_2, s_2, B_1)} \left(\bigcup_{(s'_2, \mathcal{B}') \in g'_2} \{((s'_1|s'_2), \mathcal{B}')\} \right) \right)$$

(b) Else, if $length(p) > length(s)$ then let $G'_2 = g(p_2, s, B)$ and let

$$S'_1 = \bigcup_{(s'_2,\mathcal{B}')\in G'_2} \left(\bigcup_{B'\in\mathcal{B}'} \{\sigma_{B'}(p_1)\} \right).$$

Then

$$g(p, s, B) = \bigcup_{s'_1\in S'_1} \left(\bigcup_{(s'_2,\mathcal{B}')\in G'_2} \{((s'_1|s'_2), \mathcal{B}')\} \right)$$

3. Else if $p = (p_1|p_2)$ and $s = ()$ then $g(p, s, B) = \{(\sigma_B(p), \{B\})\}$

4. Else $g(p, s, B) = \emptyset$

Let us clarify these steps. The first step is obvious and ensures that no unnecessary modifications are done: the merging of a pattern and a source that unify is equivalent to leaving the source unchanged and unifying them.

The second step consists of two possibilities. If the first element of the pattern merges with the first element of the source (case (a)) then the result is further completely determined by the results $g'_2 = (s'_2, \mathcal{B}')$ of merging the remaining elements of the source and the pattern.

Otherwise (case (b), the first elements do not merge), if the pattern is longer then the source we can consider extending the source with the first element of the pattern. The result is then further completely determined by the result G'_2 of merging the remaining elements of the pattern with the entire source. And because this might involve a set of bindings which could potentially lead to different expressions for the first element of the pattern p_1, the combinations of such distinct expressions and bindings need to be computed.

Theorem A.4.1 (Termination of $g(p, s, B)$)**.** *The definition above can be viewed as an algorithm to compute the value of $g(p, s, B)$. It is obvious that this algorithm will always terminate when called on a pattern of finite length: although it is called recursively in steps 2(a) and (b), it is always called on a pattern of smaller length. This can only continue until the pattern is of length 0 (i.e. is equal to ()) in which case the algorithm always returns from steps 1 or 4.* □

Example A.4.1.1. *Let a and b be constants. Then:*

$$
\begin{aligned}
&g(a, a, \{\epsilon\}) = \{(a, \{\epsilon\})\} \\
&g((a\ b), (a), \{\epsilon\}) = \{((a\ b), \{\epsilon\})\} \\
&g((a\ b), (b), \{\epsilon\}) = \{((a\ b), \{\epsilon\})\} \\
&g((a\ ?y), (a), \{\epsilon\}) = \{((a?y), \{\epsilon\})\} \\
&g((?x\ b), (a), \{\epsilon\}) = \{((a\ b), \{[?x/a]\})\} \\
&g((?x\ ?y), (a), \{\epsilon\}) = \{((a\ ?y), \{[?x/a]\})\}. \qquad \text{(A.7)}
\end{aligned}
$$

Example A.4.1.2. *The merging of*

```
(?unit (form ?form)
       (syn-cat ((lex-cat Verb))))
```

and

```
(Unit (form ((stem Unit "walk"))))
```

gives the expanded unit

```
(Unit (form ((stem Unit "walk")))
      (syn-cat ((lex-cat Verb))))
```

and bindings

```
[?unit/Unit, ?form/((stem ?unit "walk"))].
```

A.4.2 Merging a general pattern

We now turn to the case where the pattern p can be any FCG-expression. As with unification, the merge function g is extended with specialized merge functions whenever the pattern is of the form $p = (o...)$ with $o \in \mathcal{O}$.

The includes operator

Let us first look at the case where $p = (== e_1...e_n)$. The main differences with the simple case is that now neither the order nor the number of elements in the source matters:

Example A.4.2.1.

$$
\begin{aligned}
&g_{==}((== b\ a), (a\ b), \{\epsilon\}) = \{((a\ b), \{\epsilon\})\} \\
&g_{==}((== b\ a), (a), \{\epsilon\}) = \{((a\ b), \{\epsilon\})\} \qquad \text{(A.8)}
\end{aligned}
$$

Example A.4.2.2. *(Compare with example A.4.1.2.) The merging of*

```
(?unit == (syn-cat ((lex-cat Verb))))
```

and

```
(Unit (form ((stem Unit "walk"))))
```

gives the expanded unit

```
(Unit (form ((stem Unit "walk")))
      (syn-cat ((lex-cat Verb))))
```

and bindings

```
[?unit/Unit].
```

The algorithm presented above can be used to compute the merge of an includes list with only a minimal amount of changes. Let $p = (== |(p_1|p_2))$. First, in step 2, instead of trying to merge p_1 only to the first element of the source, all source elements must be considered. Every source element that merges with p_1 now leads to a case similar to 2(a). The computation of the union of the results for these cases is somewhat more complicated and requires some additional bookkeeping.

If no source element merges with the first pattern element then this leads to a case similar to 2(b). G'_2 is now computed as

$$G'_2 = g((== |p_2), s, B)$$

i.e. the includes operator must be propagated.

Merging an includes list also always terminates for the same reasons as why the merging of simple expressions terminates.

The permutation operator

Merging a permutation pattern $p = (==_p e_1...e_n)$ is similar to simple merging except that the order of elements in the source is arbitrary. As in the case of the includes operator, this requires that in step 2 all elements in the source are considered instead of only the first. A more easy but possibly less efficient implementation would be to merge the pattern as if it is an includes pattern and only keep those results that are of the same length as the original pattern (without the permutation operator.)

The includes uniquely operator

The includes uniquely operator can be used to block merging. Consider for example the patterns

p_1 = `((?unit (form (== (string ?unit "car"))`
`(syn-cat (== (number singular))))))`

and

p_2 = `((?unit (form (==`$_1$ `(string ?unit "car"))`
`(syn-cat (==`$_1$ `(number singular))))))`

and the source

s = `((unit (form ((string unit "cars")))`
`(syn-cat ((number plural))))).`

The source represents (part of) a syntactic structure. The patterns represent template-poles that are tried to merge with the source to obtain a new syntactic structure. In this case both patterns are intended to fail because a unit cannot be both singular (as specified by the patterns) and plural (as specified in the source.) However, merging p_1 and s results in

$g(p_1, s, \epsilon)$ = {(((unit (form ((string unit "car")
(string unit "cars")))
(syn-cat ((number singular)
(number plural))))),
{[?unit/unit]})}

whereas p_2 and s do not merge: the merging is blocked by the includes uniquely operator.

An includes uniquely pattern can be merged with a source by first treating the pattern as a normal includes pattern and then filtering the result on the conditions of section A.2.5. This can be made more efficient by checking whether it is allowed to add a new element to the source in step 2(b) of the merging algorithm.

A.5 Merging Feature Structures

As with unification, the merging of a pattern feature structure P with a source structure S will be defined as merging a transformed pattern P' with the source. The transformation consists of adding special operators to the pattern. However, the set of special operators defined so far does not suffice. Consider the merging of the pattern:

`((?unit (sem-cat (== (agent ?e ?a) (human ?a))))),`

with the following source:

```
((unit (sem-cat ((agent e a) (motion-event e))))).
```

The intended result with bindings $[?e/e, ?a/a]$ is clearly:

```
((unit (sem-cat ((agent e a) (motion-event e) (human a))))).
```

This solution requires that the first includes element is unified with the first source element and that the `human` part is added. However, the first includes element also merges with the second source element by adding `agent` to it, leading to the solution:

```
((unit (sem-cat ((agent e a) (agent motion-event e)
                 (human e))))),
```

with bindings $[?e/\text{motion-event}, ?a/e]$.

In this particular case the spurious solution can be ruled out by changing the includes operator == to an includes uniquely operator $==_1$. However, this is not always possible, and some more general mechanism is needed that allows to specify that feature values like `(motion-event e)` may not be modified during merging.

Therefore, for every special operator $o \in \mathcal{O}$ a non-destructive version $o!$ is defined which behaves the same in unification (i.e. $f_o = f_{o!}$) but which differs in merging such that the modification of candidate source elements for an element of a non-destructive pattern is prohibited. In terms of the merge algorithm g in section A.4.1 this means that the recursive call to g in step 2 to determine G'_1 is replaced by a call to f_{FCG} and that steps 2(b) and step 3 are not allowed because they modify the source.

By using non-destructive special operators the modification of already present feature value elements can be prohibited. However, there is another problem. Consider the merging of the pattern

```
(==1 (unit1 ==1 (F1 V1))
     (unit2 ==1 (F2 V2))),
```

with the source

```
((unit1)
 (unit2)))
```

One expected result is

```
((unit1 (F1 V1))
 (unit2 (F2 V2))).
```

However, the following is also a valid merge:

```
((unit1 unit2 (F1 V1))
 (unit2 unit1 (F2 V2))).
```

Prohibiting this solution requires the introduction of a final special operator $==_{1l}$ which is equivalent to the includes uniquely operator except that it only allows its elements to be lists.

Definition 17. The function **expand-structure(P,S,B)** which takes a pattern structure P, a source structure S and a set of bindings B is defined as follows. If P is as represented in (A.5) then the pattern is first transformed to the pattern P':

$$\begin{aligned}(==_{1l}\ (u_1\ \ ==_{1l}\ (f_{11}\ v'_{11})...(f_{1n_1}\ v'_{1n_1}))\\ \ldots\\ (u_m\ \ ==_{1l}\ (f_{m1}\ v'_{m1})...(f_{mn_m}\ v'_{mn_m}))),\end{aligned} \tag{A.9}$$

with the new feature values determined as follows: Every non-atomic feature value $v_{ij} = (v_1|v_2)$ in the pattern for which v_1 is not a special operator is replaced by $v'_{ij} = (==!_p|v_{ij})$. If v_1 is a special operator then it is replaced by its non-destructive version. Atomic feature values are left unchanged: $v'_{ij} = v_{ij}$ if v_{ij} is atomic. Expand-structures(P,S,B) is then equal to $g(P', S, \{B\})$.

A.6 Examples

The examples presented in this section are simplified to focus on the unification and merging aspects of FCG-template application and do not take the J-operator into account.

A.6.1 Example of syntactic categorisation in parsing

Assume the following syntactic structure, which could be built based on the utterance "Mary walks":

```
Syn=((sentence-unit (syn-subunits (Mary-unit walks-unit)))
     (Mary-unit (form ((string Mary-unit "Mary"))))
     (walks-unit (form ((string walks-unit "walks")))))
```

The structure contains three units: one for both words ('strings') in the sentence, and one to keep these together in a sentence unit. The initial corresponding semantic structure might look like:

```
Sem=((sentence-unit (sem-subunits (Mary-unit walks-unit)))
     (Mary-unit)
     (walks-unit))
```

It does not yet contain any meanings because we are in the beginning of the parsing process before application of the lexical templates.

As explained elsewhere, the first type of template that is applied during parsing in FCG is concerned with morpho-syntactic transformations and syntactic and semantic categorisations. In parsing, this phase is comparable to more traditional part-of-speech tagging. However, in FCG these templates can be applied both during production and in parsing and the set of form-constraints and syntactic categories (like parts of speech) is open-ended.

The following template categorises the string "walks" as the third-person singular form of the verb-stem "walk":

```
((?unit (form (== (stem ?unit "walk")))
        (syn-cat (==1 (number singular)
                      (person third)))))
<-->
((?unit (form (== (string ?unit "walks")))))
```

While producing, the same rule would be applied to establish the third-person singular form "walks" for the stem "walk".

To test the applicability of the above template while parsing, the right pole must be unified with the syntactic structure. As explained earlier, this requires first the transformation of the pole to the pattern R' (see equation A.6):

```
R'=(==1 (?unit ==1 (form (== (string ?unit "walks"))))),
```

followed by the unification of this new pattern with the syntactic structure:

$$\mathcal{B} = f_{\text{FCG}}(R', Syn, \{\epsilon\}) = \{[?unit/\text{walks-unit}]\}$$

Because this yields a valid set of bindings, the template's left pole can be applied to compute a new, extended syntactic structure `Syn'` (syntactic categorisation rules always work only on syntactic structures). This requires that the template's left pole is merged with the syntactic structure. Therefore, it is first transformed to the pattern L' (see equation A.9):

```
L'= (==1l (?unit ==1l (form (==! (stem ?unit "walk") ))
                      (syn-cat (==1! (number singular)
                                     (person third))))),
```

which then is merged with the syntactic structure: $g(L', Syn, \mathcal{B}) = \{(Syn', \mathcal{B})\}$, yielding:

```
Syn'=((sentence-unit (syn-subunits (Mary-unit walks-unit)))
      (Mary-unit (form ((string Mary-unit "Mary"))))
      (walks-unit (form ((string walks-unit "walks")
                         (stem walks-unit "walk")))
                  (syn-cat ((number singular)
                            (person third))))).
```

A.6.2 Example of lexicon lookup in parsing

Here is next a lexical template associating a predicate-argument structure with the stem "walk":

```
((?unit (referent ?event)
        (meaning (== (walk ?event) (walker ?event ?person)))))
<-->
((?unit (form (== (stem ?unit "walk")))))
```

In parsing, this template is triggered by a successful unification of its right pole with the syntactic structure. Therefore, the pole is first transformed to the pattern R'':

```
R''=(==1 (?unit ==1 (form (== (stem ?unit "walk"))))).
```

It is easy to see that R'' indeed unifies with the syntactic structure `Syn'` from the previous example with unifier B'=[?unit/`walks-unit`].

Given successful unification, the left pole can be merged with the semantic structure, yielding the new semantic structure `Sem'`, with $g(L'', Sem, \{B'\}) = \{(Sem', \{B'\})\}$,

```
L''=(==1l (?unit ==1l (meaning
                       (==! (walk ?event)
                            (walker ?event ?person)))))
```

and thus

```
Sem'=((sentence-unit (sem-subunits (Mary-unit walks-unit)))
      (Mary-unit)
      (walks-unit (referent ?event)
                  (meaning ((walk ?event)
                            (walker ?event ?person))))).
```

A.6.3 Example of construction application in production

Assume that conceptualization, lexicalisation and categorisation resulted in the following semantic and syntactic structures:

```
Sem=((sentence-unit (sem-subunits (Mary-unit walk-unit)))
     (Mary-unit (referent person-1)
                (meaning ((Mary person-1))))
     (walk-unit (referent ev-1)
                (meaning (walk ev-1)
                         (walker ev-1 person-1))
                (sem-cat (motion-event ev-1)
                         (agent ev-1 person-1))))
```

and

```
Syn=((sentence-unit (syn-subunits (Mary-unit walk-unit)))
     (Mary-unit (form ((stem Mary-unit "Mary")))
                (syn-cat ((person third)
                          (number singular))))
     (walk-unit (form ((strem walk-unit "walk"))))).
```

The above syntactic structure specifies that there are two lexical items involved (the stems "Mary" and "walk"), reflecting the fact that the meaning to express involves some person `person-1` (Mary) and some walk event `ev-1`. However it is not yet specified that it is Mary who fulfills the role of walker (agent) in the walk event. The following simple SV-construction template can be used for this and uses word order and agreement as would be the case in English:

```
((?SV-unit (sem-subunits (?subject-unit ?predicate-unit)))
 (?subject-unit (referent ?s))
 (?predicate-unit (referent ?p)
                  (sem-cat (==1 (agent ?p ?s)))))
<-->
((?SV-unit (syn-subunits (?subject-unit ?predicate-unit))
           (form (== (precedes ?subject-unit
                               ?predicate-unit))))
 (?subject-unit (syn-cat (==1 NP
                              (number ?n)
                              (person ?p))))
 (?predicate-unit (syn-cat (==1 verb
                              (number ?n)
                              (person ?p))))).
```

Many other syntactic constraints can easily be incorporated into this kind of template. The above template's left pole unifies with the semantic structure `Sem` with unifier

```
B=[?SV-unit/sentence-unit, ?subject-unit/Mary-unit,
   ?predicate-unit/walk-unit, ?s/person-1, ?p/event-1].
```

Thus, a new syntactic structure `Syn'` can be computed by merging the template's right pole with the structure `Syn`: $g(R', Syn, \{B\}) = \{(Syn', \{B'\})\}$, with

```
R'=(==1l
    (?SV-unit ==1l
              (syn-subunits (==p! ?subject-unit
                                  ?predicate-unit))
              (form (==! (precedes ?subject-unit
                                   ?predicate-unit))))
    (?subject-unit ==1l
              (syn-cat (==1! NP
                             (number ?n)
                             (person ?p))))
    (?predicate-unit ==1l
              (syn-cat (==1! verb
                             (number ?n)
                             (person ?p)))))),
```

,

```
B'=[?SV-unit/sentence-unit, ?subject-unit/Mary-unit,
    ?predicate-unit/walk-unit, ?s/person-1, ?p/event-1,
    ?n/singular, ?p/third]
```

and

```
Syn'=((sentence-unit
         (syn-subunits (Mary-unit walk-unit))
         (form ((precedes Mary-unit walk-unit))))
       (Mary-unit (form ((stem Mary-unit "Mary")))
                  (syn-cat (NP
                            (person third)
                            (number singular))))
       (walk-unit (form ((strem walk-unit "walk")))
                  (syn-cat (verb
                            (person third)
                            (number singular)))))
```

A.7 Conclusion

Experiments in the emergence of grammatical languages require powerful formalisms that support the kind of features that are typically found in human natural languages. Linguists have been making various proposals about the nature of these formalisms. Even though a clear consensus is lacking, most formalisms today use a kind of feature structure representation for syntactic and semantic information and templates with variables and syntactic and semantic categories. There are also several proposals on how templates are to be assembled, centering around concepts like match, unify, merge, etc. although the proposals are often too vague to be operationalised computationally. We argued that computer simulations of the emergence of grammar have some additional technically very challenging requirements: the set of linguistic categories must be open-ended, templates can have various degrees of entrenchment, and inventories and processing must be distributable in a multi-agent population with potentially very diverse inventories.

Fluid Construction Grammar has been designed to satisfy these various requirements and the system is now fully operational and has already been used in a number of experiments.

In this appendix the unification and merging algorithms used in FCG were formally defined as they form the core of the system. It was shown that FCG unification is a special type of multi-subset-unification, which is inherently of exponential complexity in the length of the expressions that are unified. FCG unification always returns a complete but not necessarily minimal set of unifiers. FCG merging was properly defined and it was shown that it always terminates.

The unification of a source with an includes list ($==$) was formally defined and the unification of a permutation list ($==_p$) and of an includes-uniquely list ($==_1$) were shown to be special cases hereof. These made it possible to define the matching of structures, needed for FCG template application in terms of the general unification function. Non-destructive versions of these operators were introduced to enable the definition of FCG structure merging in terms of the general merging function.

FCG can be used without considering all the technicalities discussed in the present appendix, but these details are nevertheless of great importance when constructing new implementations.

Appendix B

Example (regular verbs)

B.1 Lexical constructions

```
(def-lex-stem-rule Mary
    ((?Top (meaning (== (Mary ?m))))
     ((J ?new ?Top) (referent ?m)))
   <-->
   ((?Top (form (== (string ?new "Mary"))))
    ((J ?new ?Top)
     (form (== (stem "Mary")))
     (syn-cat ((constituent NP)
       (lex-cat Proper-Noun)
       (number sing)
       (person 3rd))))))

(def-lex-stem-rule John
    ((?Top (meaning (== (John ?m))))
     ((J ?new ?Top) (referent ?m)))
   <-->
   ((?Top (form (== (string ?new "John"))))
    ((J ?new ?Top)
     (form (== (stem "John")))
     (syn-cat ((constituent NP)
       (lex-cat Proper-Noun)
       (number sing)
       (person 3rd))))))

(def-lex-stem-rule I
    ((?Top (meaning (== (speaker ?i))))
```

```
     ((J ?new ?Top) (referent ?i)))
   <-->
   ((?Top (form (== (string ?new "I"))))
    ((J ?new ?Top)
     (form (== (stem "I")))
     (syn-cat ((constituent NP)
       (lex-cat Proper-Noun)
       (number sing)
       (person 1st))))))

(def-lex-stem-rule kiss
    ((?Top (meaning (== (kiss ?k)
                        (kisser ?k ?kisser)
(kissee ?k ?kissee))))
     ((J ?new ?Top)
      (referent ?k)
      (sem-cat ((agent ?k ?kisser)
        (patient ?k ?kissee)))))
   <-->
   ((?Top (form (== (string ?new "kiss"))))
    ((J ?new ?Top)
     (form (== (stem "kiss")))
     (syn-cat ((constituent V)
       (lex-cat verb regular)
               (number ?number)
               (person ?person))))))
```

B.2 Transitive Construction

```
(def-con-rule transitive
    ((?top (sem-subunits (== ?subject ?predicate ?object)))
     (?subject (referent ?s))
     (?predicate
           (sem-cat (Tag ?sem-cat (== (agent ?p ?s)
                                      (patient ?p ?o)))))
     (?object (referent ?o))
     ((J ?new ?top)
      (sem-cat ?sem-cat)
      (head ?predicate)))
  <-->
```

```
  ((?top (syn-subunits (== ?subject ?predicate ?object))
 (form (== (meets ?subject ?predicate)
   (meets ?predicate ?object))))
   (?subject (syn-cat (== (constituent NP)
  (number ?num)
  (person ?per))))
   (?predicate (syn-cat (TAG ?syn-cat
     (== (constituent V)
 (number ?num)
 (person ?per)))))
   (?object (syn-cat (== (constituent NP))))
   ((J ?new ?top) (syn-cat ?syn-cat))))
```

B.3 Regular Verb suffix Construction

```
(def-con-rule es
    ((?top (sem-subunits (== ?verb-unit))
   (head ?verb-unit))
     ((J ?verb-unit NIL)
      (sem-cat . ?sem-cat))
     ((J ?es ?top)
      (sem-cat . ?sem-cat)))
  <-->
  ((?top (syn-subunits (== ?verb-unit))
 (form (== (meets ?verb-unit ?es)
   (string ?es "es"))))
   (?verb-unit
 (syn-cat (==1 (lex-cat verb regular)
       (number sing)
       (person 3rd))))
   ((J ?es ?top)
    (syn-cat ((constituent V)
      (number sing)
      (person 3rd))))))
```

B.4 Parsing of “Mary kiss es John”

```
;;;; (parse '("Mary" "kiss" "es" "John" ))  ...
```

```
================== PARSING ======================
initial utterance: (Mary kiss es John)
Syntactic structure:
((TOP
    (FORM
     ((STRING John John) (STRING es es)
      (STRING kiss kiss)
      (STRING Mary Mary) (MEETS Mary kiss)
      (MEETS kiss es) (MEETS es John)))))

---------------- Morph rules -------------------
No rules could be applied.

-------------- Lex-stem rules ------------------
Rules applied:
   KISS, JOHN, MARY
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (Mary John kiss)))
    (John
       (MEANING ((JOHN ?john)))
       (REFERENT ?john))
    (Mary
       (MEANING ((MARY ?X-3164)))
       (REFERENT ?X-3164))
    (kiss
       (MEANING
        ((KISS ?X-3154)
         (KISSER ?X-3154 ?X-3155) (KISSEE ?X-3154 ?X-3156)))
       (REFERENT ?X-3154)
       (SEM-CAT ((AGENT ?X-3154 ?X-3155)
                 (PATIENT ?X-3154 ?X-3156)))))
Syntactic structure:
   ((TOP
       (FORM
        ((STRING es es) (MEETS Mary kiss)
         (MEETS kiss es) (MEETS es John)))
       (SYN-SUBUNITS (Mary John kiss)))
    (John
       (FORM ((STEM John) (STRING John John)))
       (SYN-CAT
```

```
         ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
          (NUMBER SING) (PERSON 3RD))))
    (Mary
       (FORM ((STEM Mary) (STRING Mary Mary)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD))))
    (kiss
       (FORM ((STEM kiss) (STRING kiss kiss)))
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR) (NUMBER ?X-3158)
         (PERSON ?X-3159)))))

---------------- Con rules ---------------------
Rules applied:
   ES
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (es Mary John)))
    (John
       (MEANING ((JOHN ?john)))
       (REFERENT ?john))
    (Mary
       (MEANING ((MARY ?X-3164)))
       (REFERENT ?X-3164))
    (es
       (HEAD kiss)
       (SEM-CAT ((AGENT ?X-3154 ?X-3155)
                 (PATIENT ?X-3154 ?X-3156)))
       (SEM-SUBUNITS (kiss))))
Syntactic structure:
   ((TOP
       (FORM ((MEETS Mary es)
              (MEETS es John)))
       (SYN-SUBUNITS (es Mary John)))
    (John
       (FORM ((STEM John) (STRING John John)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD))))
    (Mary
```

```
       (FORM ((STEM Mary) (STRING Mary Mary)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD))))
    (es
       (FORM ((MEETS kiss es) (STRING es es)))
       (SYN-CAT ((CONSTITUENT V) (NUMBER SING) (PERSON 3RD)))
       (SYN-SUBUNITS (kiss))))

Try more constructions? [y/n] y

No rules could be applied.

Final structures:
Semantic structure:
((TOP
    (SEM-SUBUNITS (es Mary John)))
 (John
    (MEANING ((JOHN ?john)))
    (REFERENT ?john))
 (Mary
    (MEANING ((MARY ?mary)))
    (REFERENT ?mary))
 (es
    (HEAD kiss)
    (SEM-CAT ((AGENT ?kiss ?agent)
              (PATIENT ?kiss ?patient)))
    (SEM-SUBUNITS (kiss)))
 (kiss
    (MEANING ((KISS ?kiss)
              (KISSER ?kiss ?agent)
              (KISSEE ?kiss ?patient)))
    (REFERENT ?kiss)
    (SEM-CAT ((AGENT ?kiss ?agent)
              (PATIENT ?kiss ?patient)))))
Syntactic structure:
((TOP
    (FORM ((MEETS Mary es) (MEETS es John)))
    (SYN-SUBUNITS (es Mary John)))
 (John
    (FORM ((STEM John) (STRING John John)))
```

```
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 3RD))))
 (Mary
    (FORM ((STEM Mary) (STRING Mary Mary)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 3RD))))
 (es
    (FORM ((MEETS kiss es) (STRING es es)))
    (SYN-CAT ((CONSTITUENT V) (NUMBER SING) (PERSON 3RD)))
    (SYN-SUBUNITS (kiss)))
 (kiss
    (FORM ((STEM kiss) (STRING kiss kiss)))
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 3RD)))))

Parsed meaning: ((KISS ?kiss) (KISSER ?kiss ?agent)
                 (KISSEE ?kiss ?patient)
                 (JOHN ?john) (MARY ?mary))
```

B.5 Generation of "Mary kiss es John"

```
================== PRODUCING ====================
Semantic structure:
((TOP
    (MEANING ((KISS K) (KISSER K M) (KISSEE K J)
              (JOHN J) (MARY M)))))

-------------- Lex-stem rules -----------------
Rules applied:
   KISS, JOHN, MARY
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (Mary John kiss)))
    (kiss
       (MEANING ((KISS K) (KISSER K M) (KISSEE K J)))
       (REFERENT K)
       (SEM-CAT ((AGENT K M) (PATIENT K J))))
```

```
    (John
       (MEANING ((JOHN J)))
       (REFERENT J))
    (Mary
       (MEANING ((MARY M)))
       (REFERENT M)))
Syntactic structure:
   ((TOP
       (SYN-SUBUNITS (Mary John kiss)))
    (kiss
       (FORM ((STEM kiss) (STRING kiss kiss)))
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
         (NUMBER ?number) (PERSON ?person))))
    (John
       (FORM ((STEM John) (STRING John John)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD))))
    (Mary
       (FORM ((STEM Mary) (STRING Mary Mary)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD)))))

---------------- Con rules ---------------------
Rules applied:
   TRANSITIVE
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (S)))
    (S
       (HEAD kiss)
       (SEM-CAT ((AGENT K M) (PATIENT K J)))
       (SEM-SUBUNITS (Mary kiss John))))
Syntactic structure:
   ((TOP
       (SYN-SUBUNITS (S)))
    (S
       (FORM ((MEETS Mary kiss)
              (MEETS kiss John)))
```

```
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
         (NUMBER SING) (PERSON 3RD)))
       (SYN-SUBUNITS (Mary kiss John))))

Try more constructions? [y/n] y

Rules applied:
   ES
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (S)))
    (S
       (SEM-CAT ((AGENT K M) (PATIENT K J)))
       (SEM-SUBUNITS (es Mary John))))
Syntactic structure:
   ((TOP
       (SYN-SUBUNITS (S)))
    (S
       (FORM ((MEETS es John)
              (MEETS Mary es)))
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
         (NUMBER SING) (PERSON 3RD)))
       (SYN-SUBUNITS (es Mary John))))

Try more constructions? [y/n] y
No rules could be applied.

Final structures:
Semantic structure:
((TOP
    (SEM-SUBUNITS (S)))
 (kiss
    (MEANING ((KISS K) (KISSER K M) (KISSEE K J)))
    (REFERENT K)
    (SEM-CAT ((AGENT K M) (PATIENT K J))))
 (S
    (SEM-CAT ((AGENT K M) (PATIENT K J)))
    (SEM-SUBUNITS (es Mary John)))
 (John
```

```
    (MEANING ((JOHN J)))
    (REFERENT J))
 (Mary
    (MEANING ((MARY M)))
    (REFERENT M))
 (es
    (HEAD kiss)
    (SEM-CAT ((AGENT K M) (PATIENT K J)))
    (SEM-SUBUNITS (kiss))))
Syntactic structure:
((TOP
    (SYN-SUBUNITS (S)))
 (kiss
    (FORM ((STEM kiss) (STRING kiss kiss)))
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 3RD))))
 (S
    (FORM ((MEETS es John)
           (MEETS Mary es)))
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 3RD)))
    (SYN-SUBUNITS (es Mary John)))
 (John
    (FORM ((STEM John) (STRING John John)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 3RD))))
 (Mary
    (FORM ((STEM Mary) (STRING Mary Mary)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 3RD))))
 (es
    (FORM ((MEETS kiss es)
           (STRING es es)))
    (SYN-CAT ((CONSTITUENT V) (NUMBER SING) (PERSON 3RD)))
    (SYN-SUBUNITS (kiss))))

==> UTTERANCE: (Mary kiss es John)
```

B.6 Parsing of "I kiss Mary"

```
================== PARSING ======================
initial utterance: (I kiss Mary)
Syntactic structure:
((TOP
    (FORM
     ((STRING Mary Mary) (STRING kiss kiss)
      (STRING I I)
      (MEETS I kiss) (MEETS kiss Mary)))))

--------------- Morph rules ------------------
No rules could be applied.

-------------- Lex-stem rules -----------------
Rules applied:
   KISS, I, MARY
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (Mary I kiss)))
    (I
       (MEANING ((SPEAKER ?speaker)))
       (REFERENT ?speaker))
    (Mary
       (MEANING ((MARY ?mary)))
       (REFERENT ?mary))
    (kiss
       (MEANING
        ((KISS ?kiss)
         (KISSER ?kiss ?kisser) (KISSEE ?kiss ?kissee)))
       (REFERENT ?kiss)
       (SEM-CAT ((AGENT ?kiss ?kisser)
                 (PATIENT ?kiss ?kissee)))))
Syntactic structure:
   ((TOP
       (FORM ((MEETS I kiss)
              (MEETS kiss Mary)))
       (SYN-SUBUNITS (Mary I kiss)))
    (I
       (FORM ((STEM I) (STRING I I)))
       (SYN-CAT
```

```
         ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
          (NUMBER SING) (PERSON 1ST))))
    (Mary
       (FORM ((STEM Mary) (STRING Mary Mary)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD))))
    (kiss
       (FORM ((STEM kiss) (STRING kiss kiss)))
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
         (NUMBER ?X-3323) (PERSON ?X-3324)))))

---------------- Con rules ---------------------
Rules applied:
   TRANSITIVE
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (S)))
    (S
       (HEAD kiss)
       (SEM-CAT ((AGENT ?kiss ?kisser)
                 (PATIENT ?kiss ?mary)))
       (SEM-SUBUNITS (I kiss Mary))))
Syntactic structure:
   ((TOP
       (SYN-SUBUNITS (S)))
    (S
       (FORM ((MEETS I kiss)
              (MEETS kiss Mary)))
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
         (NUMBER SING) (PERSON 1ST)))
       (SYN-SUBUNITS (I kiss Mary))))

Try more constructions? [y/n] y

No rules could be applied.

Final structures:
Semantic structure:
```

```
((S
    (HEAD kiss)
    (SEM-CAT ((AGENT ?kiss ?kisser)
              (PATIENT ?kiss ?mary)))
    (SEM-SUBUNITS (I kiss Mary)))
 (I
    (MEANING ((SPEAKER ?kisser)))
    (REFERENT ?kisser))
 (Mary
    (MEANING ((MARY ?mary)))
    (REFERENT ?mary))
 (TOP
    (SEM-SUBUNITS (S)))
 (kiss
    (MEANING ((KISS ?kiss)
              (KISSER ?kiss ?kisser) (KISSEE ?kiss ?mary)))
    (REFERENT ?kiss)
    (SEM-CAT ((AGENT ?kiss ?kisser)
              (PATIENT ?kiss ?mary)))))
Syntactic structure:
((S
    (FORM ((MEETS I kiss)
           (MEETS kiss Mary)))
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 1ST)))
    (SYN-SUBUNITS (I kiss Mary)))
 (I
    (FORM ((STEM I) (STRING I I)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 1ST))))
 (Mary
    (FORM ((STEM Mary) (STRING Mary Mary)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 3RD))))
 (TOP
    (SYN-SUBUNITS (S)))
 (kiss
    (FORM ((STEM kiss) (STRING kiss kiss)))
```

```
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 1ST)))))


Parsed meaning: ((KISS ?kiss)
                 (KISSER ?kiss ?kisser)
                 (KISSEE ?kiss ?mary)
                 (SPEAKER ?kisser) (MARY ?mary))
```

B.7 Generation of "I kiss Mary"

```
================== PRODUCING ====================
Semantic structure:
((TOP
    (MEANING ((KISS K) (KISSER K S) (KISSEE K M)
              (SPEAKER S) (MARY M)))))

-------------- Lex-stem rules -----------------
Rules applied:
   KISS, I, MARY
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (Mary I kiss)))
    (kiss
       (MEANING ((KISS K) (KISSER K S) (KISSEE K M)))
       (REFERENT K)
       (SEM-CAT ((AGENT K S) (PATIENT K M))))
    (I
       (MEANING ((SPEAKER S)))
       (REFERENT S))
    (Mary
       (MEANING ((MARY M)))
       (REFERENT M)))
Syntactic structure:
   ((TOP
       (SYN-SUBUNITS (Mary I kiss)))
    (kiss
       (FORM ((STEM kiss) (STRING kiss kiss)))
       (SYN-CAT
```

```
         ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
          (NUMBER ?number) (PERSON ?person))))
    (I
       (FORM ((STEM I) (STRING I I)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 1ST))))
    (Mary
       (FORM ((STEM Mary) (STRING Mary Mary)))
       (SYN-CAT
        ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
         (NUMBER SING) (PERSON 3RD)))))

---------------- Con rules ---------------------
Rules applied:
   TRANSITIVE
Semantic structure:
   ((TOP
       (SEM-SUBUNITS (S)))
    (S
       (HEAD kiss)
       (SEM-CAT ((AGENT K S) (PATIENT K M)))
       (SEM-SUBUNITS (I kiss Mary))))
Syntactic structure:
   ((TOP
       (SYN-SUBUNITS (S)))
    (S
       (FORM ((MEETS I kiss)
              (MEETS kiss Mary)))
       (SYN-CAT
        ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
         (NUMBER SING) (PERSON 1ST)))
       (SYN-SUBUNITS (I kiss Mary))))

Try more constructions? [y/n] y

No rules could be applied.

Final structures:
Semantic structure:
((TOP
```

```
    (SEM-SUBUNITS (S)))
 (S
    (HEAD kiss)
    (SEM-CAT ((AGENT K S) (PATIENT K M)))
    (SEM-SUBUNITS (I kiss Mary)))
 (kiss
    (MEANING ((KISS K) (KISSER K S) (KISSEE K M)))
    (REFERENT K)
    (SEM-CAT ((AGENT K S) (PATIENT K M))))
 (I
    (MEANING ((SPEAKER S)))
    (REFERENT S))
 (Mary
    (MEANING ((MARY M)))
    (REFERENT M)))
Syntactic structure:
((TOP
    (SYN-SUBUNITS (S)))
 (S
    (FORM ((MEETS I kiss)
           (MEETS kiss Mary)))
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 1ST)))
    (SYN-SUBUNITS (I kiss Mary)))
 (kiss
    (FORM ((STEM kiss) (STRING kiss kiss)))
    (SYN-CAT
     ((CONSTITUENT V) (LEX-CAT VERB REGULAR)
      (NUMBER SING) (PERSON 1ST))))
 (I
    (FORM ((STEM I) (STRING I I)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 1ST))))
 (Mary
    (FORM ((STEM Mary) (STRING Mary Mary)))
    (SYN-CAT
     ((CONSTITUENT NP) (LEX-CAT PROPER-NOUN)
      (NUMBER SING) (PERSON 3RD)))))
```

```
==> UTTERANCE: (I kiss Mary)
```

Bibliography

Dovier Agostino, Pontelli Enrico, and Rossi Gianfranco. Set unification. *arXiv:cs.LO/0110023v1*, October 2001.

A. Baronchelli, M. Felici, E. Caglioti, V. Loreto, and L. Steels. Sharp transition towards shared vocabularies in multi-agent systems. *J. Stat. Mech.*, P06014, 2006a. doi: 10.1088/1742-5468/2006/06/P06014.

Andrea Baronchelli, Luca Dall'Asta, Alain Barrat, and Vittorio Loreto. Strategies for fast convergence in semiotic dynamics. In Luis M. Rocha et al., editor, *Artificial Life X*, pages 480–485. MIT Press, 2006b.

John Batali. Computational simulations of the emergence of grammar. In J. Hurford and M. Studdert-Kennedy, editors, *Approaches to the evolution of language: social and cognitive bases*. Cambridge University Press, Cambridge, 1999.

John Batali. The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In Ted Briscoe, editor, *Linguistic evolution through language acquisition: formal and computational models*. Cambridge University Press, Cambridge, UK, 2002.

Tony Belpaeme. Simulating the formation of color categories. In Bernhard Nebel, editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 393–398, Seattle, WA, 2001. Morgan Kaufmann, San Francisco, CA.

B.K. Bergen and N.C. Chang. Embodied construction grammar in simulation-based language understanding. In J.O. Ostman and M. Fried, editors, *Construction Grammar(s): Cognitive and Cross-Language Dimensions*. John Benjamin Publ Cy., Amsterdam, 2003.

Derek Bickerton. Catastrophic evolution: the case for a single step from protolanguage to full hman language. In J.R. Hurford, Studdert-Kennedy M.,

and Knight C., editors, *Approaches to the evolution of language*, pages 341–358. Cambridge University Press, New York, 1998.

Derek Bickerton. Language origins and evolutionary plausibility. *Language and Communication*, 11(1-2):37–39, 1991. doi: 10.1016/0271-5309(91)90014-M.

Joris Bleys. On the origins of recursive rules: A usage based account. In *Accepted for the 7th evolution of language conference (Evolang 7)*, 2008.

H. Brighton and S. Kirby. The survival of the smallest: Stability conditions for the cultural evolution of compositional language. In J. Kelemen and P. Sosk, editors, *ECAL01*, pages 592–601. Springer-Verlag, 2001.

Henry Brighton. Compositional syntax from cultural transmission. *Artificial Life*, 8(1), 2002.

Angelo Cangelosi and Domenico Parisi, editors. *Simulating the Evolution of Language*. Springer Verlag, London, 2001.

Noam Chomsky. *The Minimalist Program*. MIT press, Cambridge Ma., 1995.

Noam Chomsky. *Rules and representations*. Blackwell, Oxford, UK, 1980.

Noam Chomsky. Discussion of putnam's comment's. In Piattelli-Palmarini, editor, *Language and Learning: the debate between Jean Piaget and Noam Chomsky*. Harvard University Press, 1982.

Noam Chomsky. Language and problems of knowledge: The managua lectures. MIT Press, 1988.

M.H. Christiansen and S. Kirby, editors. *Language Evolution*. Oxford University Press, Oxford, 2003.

C.M. Conway and M.H. Christiansen. Sequential learning in non-human primates. *Trends in Cognitive sciences*, 5:539–546, 2001.

W. Croft. *Syntactic categories and grammatical relations: The cognitive origin of information*. University of Chicago Press, 1991.

Joachim De Beule. Simulating the syntax and semantics of linguistic constructions about time. In Nathalie Gontier, Jean Paul van Bendegem, and Diederik Aerts, editors, *Evolutionary Epistemology, Language and Culture - A non-adaptationist, systems theoretical approach*, Theory and Decision Library - Series A: Philosophy and Methodology of the Social Sciences. Springer, Dordrecht, 2006.

Joachim De Beule. The emergence of compositionality, hierarchy and recursion in peer-to- peer interactions. In *Accepted for the 7th evolution of language conference (Evolang 7)*, 2008.

Joachim De Beule and Benjamin K. Bergen. On the emergence of compositionality. In *Proceedings of the 6th International Conference on the Evolution of Language*, pages 35–42, 2006.

Joachim De Beule and Bart De Vylder. Does language shape the way we conceptualize the world? In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, 2005.

Joachim De Beule and Luc Steels. Hierarchy in fluid construction grammar. In U. Furbach, editor, *Proceedings of KI-2005*, volume 3698 of *Lecture Notes in AI*, pages 1–15, Berlin, 2005. Springer-Verlag.

Joachim De Beule, Bart De Vylder, and Tony Belpaeme. A cross-situational learning algorithm for damping homonymy in the guessing game. In Luis M. Rocha et al., editor, *Artificial Life X*, pages 466–472. MIT Press, 2006.

Bart de Boer. Emergence of sound systems through self-organisation. In Michael Studdert-Kennedy, James R. Hurford, and Chris Knight, editors, *The Evolutionary Emergence of Language : Social Function and the Origins of Linguistic Form*. Cambridge University Press, Cambridge, 2000a.

Bart de Boer. Evolution and self-organisation in vowel systems,. *Evolution of communication*, 3(1):79–102, 1999.

Bart de Boer. Self organization in vowel systems. *Journal of Phonetics*, 28: 441–465, 2000b.

Bart de Boer and Patricia K. Kuhl. Infant-directed vowels are easier to learn for a computer model. *Journal of the Acoustical Society of America*, 110(5 pt 2):2703, 2001.

Bart de Boer and Patricia K. Kuhl. Investigating the role of infant-directed speech with a computer model. *Acoustic Review Letters On-line*, 2002. (in preparation).

Bart De Vylder and Karl Tuyls. How to reach linguistic consensus: A proof of convergence for the naming game. *Journal of Theoretical Biology*, 242(4): 818–831, October 2006. doi: 10.1016/j.jtbi.2006.05.024.

Anatoli Degyarev and Andrei Voronkov. Equality elimination for semantic tableaux. Technical Report 90, Computer science department, Uppsala University, Upsalla, Sweden, 1994.

W. Tecumseh Fitch. Kin selection and "mother tongues": A neglected component in language evolution. In Kimbrough Oller and Ulrike Griebel, editors, *Evolution of Communication Systems: A Comparative Approach*, pages 275–296. MIT Press, Cambridge, MA, 2004.

Timothy Q. Gentner, Kimberly M. Fenn, Daniel Margoliash, and Howard C. Nusbaum. Recursive syntactic pattern learning by songbirds. *Nature*, 0, 2006.

A. Goldberg. *Constructions: A Construction grammar approach to argument structure*. University of Chicago Press, Chicago, 1995.

Stephen J. Gould. The limits of adaptation: is languauge a spandrel of the human brain?, October 1987. paper presented to the Cognitive Science Seminar, MIT,.

Takashi Hashimoto and Takashi Ikegami. The emergence of a net-grammar in communicating agents. *BioSystems*, 38:1–14, 1996.

Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. The faculty of language: What is it, who has it, and how did it evolve? *Science*, 298: 1569–1579, 11 2002. doi: 10.1126/science.298.5598.1569.

N. Hyams. *Language Acquisition and the Theory of Parameters*. Reidel, Dordrecht, Netherlands, 1986.

Ray Jackendoff. *Foundations of Language: Brain, Meaning, Grammar, Evolution*. Oxford University Press, 2002.

Ray Jackendoff. Possible stages in the evolution of the language capacity. *Trends in Cognitive Sciences*, 3(7):272–279, 1999. doi: 10.1016/S1364-6613(99)01333-9.

Ray Jackendoff and Steven Pinker. The nature of the language faculty and its implications for evolution of language (reply to fitch, hauser, and chomsky). *Cognition*, September 2005.

D. Kapur and P. Narendran. Np-completeness of the set-unification and matching problems. In *Proceedings of the Eighth International Conference on Automated Deduction, Lecture Notes in Computer Science*, volume 230, pages 289–495. Springer-Verlag, 1986.

M. Kay. Functional unification grammar: A formalism for machine translation. In *Proceedings of the International Conference of Computational Linguistics*, 1984.

Simon Kirby. Syntax without natural selection: How compositionality emerges from vocabulary in a population of learners. In C. Knight, J. Hurford, and M. Studdert-Kennedy, editors, *The Evolutionary Emergence of Language: Social function and the origins of linguistic form*. Cambridge University Press, 2000.

Simon Kirby. Spontaneous evolution of linguistic structure: an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110, 2001.

R.W. Langacker. *Grammar and Conceptualization*. Mouton de Gruyter, Den Haag, 2000.

Tom Lenaerts, Bart Jansen, Karl Tuyls, and Bart De Vylder. The evolutionary language game: An orthogonal approach. *Journal of Theoretical Biology*, 235 (4):566–582, August 2005. doi: 10.1016/j.jtbi.2005.02.009.

Philip Lieberman. *The Biology and Evolution of Language*. Harvard University Press, Cambridge, MA, 1984.

D. Lightfoot. Subjacency and sex. *Language and Comunication*, 11:67–69, 1991.

J. Mehler. Review of lieberman's "the biology and evolution of language". *Journal of the Acoustic Society of America*, 80:1558–1560, 1985.

G.F. Miller. *The Mating Mind: How Sexual Choice Shaped the Evolution of human Nature*. Doubleday, New York, 2001.

W. S-Y. Minett, J. W. andWang, editor. *Language Acquisition, Change and Emergence: Essays in Evolutionary Linguistics*. City University of Hong Kong Press, Hong Kong, 2005.

Josefina Sierra-Santibà nez. Prolog implementation of fluid construction grammar. Presented at the first FCG workshop, Paris, 2004.

Joakim Nivre. Dependency grammar and dependency parsing. Technical Report MSI report 05133, School of Mathematics and Systems Engineering, Växjö University, 2005.

Martin A. Nowak, Natalia L. Komarova, and Partha Niyogi. Evolution of universal grammar. *Science*, 291:114–118, 2001.

Masimo Piattelli-Palmari. Evolution, selection and cognition: From "learning" to parameter setting in biology and the study of language. *Cognition*, 31: 1–44, 1989.

Steven Pinker. *The language instinct: How the mind creates language*. W. Morrow, New York, 1994.

Steven Pinker and Paul Bloom. Natural languages and natural selection. *Behavioral and Brain Sciences*, 13:707–784, 1990.

C. Pollard and I. Sag. *Head-driven phrase structure grammar*. University of Chicago Press, Center for the Study of Language and Information of Stanford University, Chicago, 1994.

Carl Pollard. Lectures on the foundations of hpsg. Ohio State University, 1997.

D. Premack. 'gavagai!' or the future history of the animal language controversy. *Cognition*, 19:207–296, 1985.

W.V.O. Quine. *Word and Object*. The MIT Press, Cambridge, MA, 1960.

A. Radford. *Syntactic Theory and the Acquisition of English English Syntax*. Blackwell, Oxford, 1990.

Ivan A Sag, Thomas Wasow, and Emily M Bender. *Syntactic Theory, A Formal Introduction*. CSLI Publications, Leland Stanford Junior University, United States, 2nd edition, 2003.

A. Senghas and M. Coppola. Children creating language: Jow nicaraguan sign language acquired a spatial grammar. *Psychological Science*, 12(4):537–558, 2001.

Andrew D.M. Smith. Intelligent meaning creation in a clumpy world helps communication. *Artificial Life*, 9(2):559–574, 2003a.

Andrew D.M. Smith. *Evolving Communication through the Inference of Meaning*. PhD thesis, Theoretical and Applied Linguistics, School of Philosophy, Psychology and Language Sciences, University of Edinburgh, 2003b.

Kenny Smith, Henry Brighton, and Simon Kirby. Complex systems in language evolution: the cultural emergence of compositional structure. *Artificial Life*, 9(4):371–386, 2003a.

Kenny Smith, Simon Kirby, and Henry Brighton. Iterated learning: a framework for the emergence of language. *Artificial Life*, 9(4):3710386, 2003b.

Guy Steele. *Common LISP: the Language*. Digital Press, second edition, 1990a.

Guy Steele. *Common LISP: the Language*. Digital Press, second edition, 1990b.

L. Steels. A self-organizing spatial vocabulary. *Artificial Life Journal*, 2(3), 1995.

Luc Steels. The synthetic modeling of language origins. *Evolution of Communication*, 1(1):1–34, 1997.

Luc Steels. The origins of syntax in visually grounded robotic agents. *Artificial Intelligence*, 103(1-2):133–156, 1998.

Luc Steels. The talking heads experiment. Available from the VUB Artificial Intelligence Laboratory, Brussels, Belgium, 1999.

Luc Steels. Constructivist development of grounded construction grammars. In *Proceedings of the 42nd Assoc. for Comp. Linguistics Conference, Barcelona*, 2004.

Luc Steels. Emergent adaptive lexicons. In P. Maes, editor, *Proceedings of the Simulation of Adaptive Behaviour Conference*. The MIT Press, Cambridge, Ma., 1996.

Luc Steels. The emergence and evolution of linguistic structure: from lexical to rammatical communication systems. *Connection Science*, 17(3-4):213–230, 2005.

Luc Steels. The recruitment theory of language origins. In L. Nehaniv, C. Lyon, and A. Cangelosi, editors, *Emergence and Evolution of Linguistic Communication*, pages 129–150. Springer, Berlin, 2006.

Luc Steels and Tony Belpaeme. Coordinating perceptually grounded categories through language. A case study for colour. *Behavioral and Brain Sciences*, 2005. Accepted as target article.

Luc Steels and Martin Loetzsch. Perspective alignment in spatial language. In Kenny R. Coventry, Thora Tenbrink, and John. A Bateman, editors, *Spatial Language and Dialogue*. Oxford University Press, 2007. to appear.

Luc Steels and Paul Vogt. Grounding adaptive language games in robotic agents. In Phil Husbands and Inman Harvey, editors, *Proceedings of the Fourth European Conference on Artificial Life (ECAL'97), Complex Adaptive Systems*, Cambridge, MA, 1997. The MIT Press.

Luc Steels and Pieter Wellens. How grammar emerges to dampen combinatorial search in parsing. In Paul Vogt et al., editor, *Symbol Grounding and Beyond: Proceedings of the Third International Workshop on the Emergence and Evolution of Linguistic Communication*, pages 76–88. Springer, 2006.

Luc Steels, Frederique Kaplan, Angus McIntyre, and Joris Van Looveren. Crucial factors in the origins of word-meaning. In A. Wray, editor, *The Transition to Language*. Oxford University Press, Oxford, UK, 2002.

Luc Steels, Joachim De Beule, and Nicolas Neubauer. Linking in fluid construction grammar. In Royal Flemish Academy for Science and Art, editors, *Proceedings of BNAIC-05*, 2005.

L. Sterling and E. Shapiro, editors. *The art of PROLOG*. MIT Press, Cambridge, Massachusetts, 1986.

Joris Van Looveren. Artificial agents and natural determiners. In Wolfgang Banzhaf, Thomas Christaller, Peter Dittrich, Jan T. Kim, and Jens Ziegler, editors, *Advances in Artificial Life (Proc. of ECAL 2003)*, pages 472–481. Springer-Verlag, Germany, 2003.

Joris Van Looveren. An analysis of multiple-word naming games. In A. Van den Bosch and H. Wiegand, editors, *Proceedings of the 12th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC00), Kaatsheuvel, the Netherlands*, 2000.

Joris Van Looveren. Robotic experiments on the emergence of a lexicon. In Ben et al Kröse, editor, *Proceedings of the 13th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC01), Amsterdam, the Netherlands*, 2001.

Remi Van Trijp. The emergence of semantic roles in fluid construction grammar. In *Accepted for the 7th evolution of language conference (Evolang 7)*, 2008.

Paul Vogt. Grounding language about actions: Mobile robots playing follow me games. In Meyer, Bertholz, Floreano, Roitblat, and Wilson, editors, *SAB2000 Proceedings Supplement Book*. International Society for Adaptive Behavior, 2000.

Paul Vogt and Hans Coumans. Investigating social interaction strategies for bootstrapping lexicon development. *Journal of Artificial Societies and Social Simulation*, 6(1), Januari 2003.

W. von Humboldt. *Ueber die Verschiedenheit des Mensch- lichen Sprachbaus*. Dummlers, Bonn, 1836.

Ludwig Wittgenstein. *Philosophical Investigations.* Macmillan, New York, 1953.