Computational Semantics Meets Artificial Grammar Learning: Spatial Language Acquisition by Intelligent Agents

Gosse Minnema | s1529528
Supervisor: Dr. Crit Cremers

BA Thesis | Linguistics | Language & Cognition
Leiden University | June 2017

Quick links

Run the model online

https://gossminn.github.io/AlienLearningAgents/AlienLearningAgentsBuilds2/

View the source code

https://github.com/gossminn/AlienLearningAgents

Abstract

Artificial Grammar Learning (AGL) is a powerful experimental paradigm for testing specific hypotheses about language acquisition but is limited because of its reliance on meaningless grammatical structures. Meanwhile, formal and computational semantics provide rigorous ways to define and calculate meanings for formal languages, but are typically only used to describe or simulate the linguistic competence of adult speakers. This thesis attempts to connect these two fields by proposing a new type of AGL experiment that uses a language with both a context-free syntax and a formally defined semantics which can be used to express spatial relationships between objects. Moreover, using a computer simulation in which an Intelligent Agent (IA) acquires such a language, it shows how this new paradigm can be used to test psycholinguistic hypotheses about the acquisition of both syntax and semantics.

Contents

1 Introduction

1.1 Theoretical background

1.1.1 Artificial Grammar Learning

Artificial Grammar Learning (AGL) is an experimental paradigm within the field of psycholinguistics that is used for testing hypotheses about language acquisition. In an AGL study, researchers first create a formally defined artificial language, and then have participants try to learn this language in a lab setting. This has at least three major advantages: first, the languages can be formally defined, meaning that, in contrast to 'real', natural languages, they have well-understood properties, so that researchers have a much better idea of the final outcome of the learning process. Second, the artificial languages are often extremely limited (with a syntax consisting of a small set of simple rules and semantics and pragmatics being absent altogether), making it easier to test very specific hypotheses about the acquisition of abstract patterns that would otherwise be very hard to investigate. Finally, because the languages are completely artificial and not tied in any way to human speech, they are suitable for use in experiments with non-human participants.

A wide range of phenomena have been studied using AGL-like experiments. For example, Saffran et al. (1996) showed that infants could learn to segment speech sounds into words using only statistical cues, Marcus et al. (1999) found that infants can learn simple 'algebraic' rules (i.e., distinguish between sequences of the form 'AAB' versus of the form 'ABB'), and Spierings and ten Cate (2014) found evidence that zebra finches are sensitive to human prosody. However, the 'core business' of AGL research seems to be trying to address questions relating to the nature of syntax acquisition: for example, can other animals learn the type of hierarchical structures found in human language? And how do humans do it in the first place?

In the AGL literature, these questions are often approached from the perspective of Formal Language Theory (FLT). Much of the debate revolves around the distinction between regular languages, which can be generated by a Finite State Grammar (FSG) and supraregular languages, which need a more powerful mechanism such as a Context-Free Grammar (CFG). This distinction is considered to be psycholinguistically important because there is a consensus that human languages are supraregular, while even the most language-like communication systems found in other animals do not seem to exceed the regular level (Fitch and Friederici 2012; Beckers et al. 2012). It is quite uncontroversial that humans and at least some other animals can acquire at least some FSGs from meaningless input strings in an experimental setting; for example, Fitch and Hauser (2004) showed that both humans and tamarin monkeys can learn to distinguish between grammatical and ungrammatical sentences in a so-called '$(AB)^n$'-language, implying that they have acquired the FSG underlying this language.[1]

[1] In these languages, all sentences consist of any number of sequences of one 'A'-element followed by one 'B'-

The picture is more complicated for CFGs: despite their success with FSGs, the tamarins in Fitch and Hauser's experiment failed to learn an artificial language of the type 'A^nB^n' (in which valid sentences are comprised of a certain number of A-elements followed by the same number of B-elements), which was argued to require at least a CFG to generate it. By contrast, the human participants in this experiment were able to learn both types of languages. This was long taken as evidence that any grammatical capacities of non-human animals were limited to FSGs. Later, this view was challenged by Gentner et al. (2006), who found that European starlings (a songbird species) were in fact able to learn A^nB^n languages. Based on these findings, it was claimed that not just supraregularity, but also recursion was not unique to humans.[2]

However, these claims are controversial: while Fitch and Friederici (2012) agree that the results of the AGL studies with A^nB^n languages mentioned above show evidence of supraregular abilities in at least humans and starlings (although they strongly reject Gentner et al's recursion claim), some other authors have contested this, even when it comes to humans. For example, in a follow-up experiment with only human participants, Hochmann et al. (2008) suggested that the human participants in Fitch and Hauser (2004)'s experiment might not actually have learned any grammatical rules but could merely have relied on the acoustic properties of the stimuli. Moreover, Zimmerer et al. (2011) partially replicated Fitch & Hauser's study (again, with human participants only) and concluded that while group level results seemed to indicate that learning was successful, only a few individual participants actually acquired the underlying grammar rather than superficial approximations of it.

In response to the difficulties with A^nB^n grammars, some AGL researchers turned to different types of artificial grammars for investigating the acquisition of supraregular grammars. Shirley (2014) was the first psycholinguistic and neurolinguistic experiment that made use of Lindenmayer Grammars (or L-Systems), a type of formal language used by theoretical biologists for modeling processes like plant and algae growth (see Prusinkiewicz and Lindenmayer 2004 for an overview) but that was viewed as similar enough to human language structure for use in AGL studies. The results of Shirley's experiment looked promising, with both neurolinguistic and behavioral data appearing to indicate that her participants (human adults) indeed had acquired an L-system grammar. However, a later AGL experiment that also used L-systems but presented the stimuli in a different form (using drum sounds instead of human speech) and used another type of control stimuli failed to replicate these findings (Geambaşu et al. 2016).

Thus, in spite of a growing body of AGL experiments investigating the acquisition of supraregular grammars by various species, no one has yet been able to present a watertight argument that even humans can acquire such a grammar in an experimental setting. Although it is quite plausible that new experimental research will deliver the kind of evidence needed for such an argument, the mere fact that this has not happened so far is very surprising given the broad consensus that in real life, humans do in fact acquire CFGs (and even more powerful mechanisms). An interesting question is whether these difficulties are related to the fact that the

element (where A and B are classes of meaningless 'words'). For example, if *no*, *ba*, *la*, and *wu* are A-elements and *li*, *pa*, and *lo* are B-elements, two examples of possible sentences would be *no li ba pa* and *la pa wu mo no li* (Fitch and Hauser 2004).

[2] The recursion claim was based on the idea that the most likely grammar for generating A^nB^n languages would involve the recursive rewrite rule $S \rightarrow A\ S\ B$, which would produce a syntax tree where the outermost A would be connected to the outermost B, the second-outermost A to the second-outermost B, etc.

artificial languages used in AGL experiments, in contrast to natural languages, are purely syntactic and have no semantics at all. In their overview of the AGL literature, Fitch and Friederici (2012) mention only one learning experiment involving an artificial language with meaning, which dates back more than 35 years (Morgan and Newport 1981). Interestingly, that study's results are quite relevant to the current discussion since they showed that participants, when presented with both semantic information and information about the constituent structure of the artificial language they were learning, were able to infer the CFG underlying this grammar.

1.1.2 Computational semantics

Whether or not inferring more supraregular grammars from meaningless input will eventually turn out to be possible, the fact remains that in natural language, hierarchical syntax does not exist in a vacuum but is closely linked to compositional semantics. Although semantics is often perceived as being more abstract or elusive (and therefore more difficult to study), language researchers have been developing formal and rigorous ways to account for meaning since the late 1960s, when the logician Richard Montague started working on a model-theoretic semantics for natural languages. Initially, linguists were unaware of this work, but only a few years later, the first efforts were made to integrate Montague's approach to semantics with generative linguistics (Partee 1973), and formal semantics is now part of mainstream linguistics.

Because of its logical and mathematical origins, formal semantic theories lend themselves very well for implementation in a computer model. For example, DELILAH is a system that can parse and generate a sizeable fragment of the syntax and semantics of the Dutch language (see, for example, Cremers and Reckman 2008). Computational semantics also has other applications: for example, it can be used to facilitate automatic information extraction from texts or in dialogue systems (Blackburn and Bos 2003).

However, with a few exceptions, computational semantics seems to have a very synchronic perspective on language: it is concerned with representations of semantic knowledge that exist at a given point in time, and not with how these representations come into being in the first place. One of the exceptions is the MODOMA (Mother-Daughter Machine) project that is currently being worked on by David Shakouri at Leiden University and that aims to add a learning component to DELILAH. However, no results of this project have been published yet. Another example of a diachronic perspective on computational semantics is Luc Steels' work on language experiments with robots; for example, Spranger and Steels (2012) describe an experiment in which robots learn to use a 'pidgin German' to describe spatial configurations and acquire the mental representations necessary for doing so. However, this experiment was as much about language evolution as it was about language acquisition, making it somewhat less relevant for the present discussion.

In any case, there does not seem to be any connection so far between the AGL literature on the one hand and the formal and computational semantics literature on the other. In some sense, this is strange, since AGL and formal semantics seem to be a very natural fit: while AGL experiments use well-defined formal languages to test very specific hypotheses about language acquisition, formal semantics uses formal techniques from logic and mathematics to test specific hypotheses about natural language meaning. It should therefore be possible to combine the two, and create an artificial language with a formal semantics for use in an AGL experiment.

This lack of interaction might be unfortunate, not just for AGL researchers, but also for formal and computational semanticists: as noted in Chierchia and McConnell-Ginet (2000), a standard textbook for formal semantics: "as linguists, our focus is on modelling the cognitive systems whose operation, in some sense, 'explains' linguistic phenomena. Linguistics is an empirical science, and in that respect it is like physics and unlike (pure) mathematics" (p. 2). Arguably, any formal or computational model of a cognitive system is not complete without a theory of how the representations it is trying to model come into existence.

1.2 Aim and method

1.2.1 Research objectives

The purpose of this thesis is twofold: on the one hand, it aims to contribute to the field of AGL by proposing a new type of AGL experiment that uses artificial languages with a formally defined semantics and by showing how hypotheses about the acquisition about such languages can be developed and tested using a computer simulation. On the other hand, it wants to contribute to computational semantics by showing how meaning representations can be acquired by an artificial intelligence instead of being hard-coded into a system.

It is important to note that the main contribution that this thesis tries to make is methodological rather than empirical in nature: the computational model that is developed is primarily intended as a demonstration that it is possible for an IA to acquire a specific type of artificial language, and not as a psycholinguistic hypothesis about how and whether humans might acquire such a language (or, indeed, a real, natural language). In this sense, this research is not a true AGL study but belongs primarily to the computational semantics tradition.[3] In the artificial intelligence literature, a distinction is sometimes made between 'systems that think like humans' and 'systems that act like humans'; whereas the first approach (also called the *cognitive modeling approach*) tries to replicate human intelligence in a psychologically realistic way, the second approach (the so-called *Turing Test approach*) is concerned with emulating human intelligence on a behavioral, but not on a psychological level (Russell and Norvig 2003). If this distinction were to be applied to computational linguistics, the model developed in this thesis would be closer to the second approach than to the first approach.

1.2.2 Method and thesis structure

This thesis starts by introducing a type of AGL experiment which uses languages with not just a formal syntax but also a formal semantics for expressing spatial relationships (chapter 2). The general characteristics of this class of languages, as well as the specific language used in the computer simulation, are discussed. Additionally, the psycholinguistic relevance of this new type of experiment is explained. Next, chapters 3-5 describe the computer model, which consists of two Intelligent Agents (IA)[4]: one is the 'parent' and is pre-programmed with full

[3] Although computational models are not uncommon in AGL, they are typically used in conjunction with experimental evidence to make an empirical claim; see, for example, Alhama and Zuidema (2016).

[4] An IA is any computational entity that has some sort of 'embodiment' and can interact with its surroundings. This could be a robot with a physical presence in the real world; in this case, the agents 'live' in a virtual, spatial

knowledge of the target language, while the other is a 'child' and is not equipped with any information specific to this language (although, as will become clear, it does by no means start with a 'blank slate') but only with learning mechanisms for extracting information from the parent's utterances, for producing its own utterances and for evaluating the feedback it gets back from the parent. The learning algorithms are largely deterministic (learning takes place primarily through logical reasoning, although probabilities also play a limited role) and the knowledge that is acquired is symbolic (the child learns explicit representations of the target language). The main advantage of this approach is that it makes the learning process very transparent: it often easy to find out why exactly the child makes a particular decision, and it is also possible to compare the representations that the child has acquired to those of the father.

In chapter 3, a high-level overview of the model is given: what are the basic problems that it to solve? What general assumptions does it make about the learning process? Then, chapter 4 discusses how the model actually works: a detailed explanation is given of the logic of the learning algorithms it uses and of the representations of the language that are built up. Chapter 5 completes the description of the simulation by discussing the technical implementation of the model (the programming language and platform used, how the program is structured, and the user interface). Finally, in chapter 6, a critical assessment of the model and its possible relevance for future AGL experiments is given.

environment.

2 Artificial Grammar Learning with spatial semantics

2.1 Languages

2.1.1 General characteristics

For the purposes of the new type of AGL experiment that is introduced here, a new, very specific class of artificial languages (called 'C') was designed. The languages in C all have a very similar, and extremely limited, syntactic and semantic structure. However, the number of possible languages is infinitely large[1] and the languages may superficially look very different. An overview of the defining characteristics of this 'language space' is given in table 2.1.1. Although the 'child' in the learning simulation developed for the purposes of this thesis should in principle be able to learn any of the languages in this space, only one 'father', with knowledge of only one of these languages, is currently implemented. This specific language (abbreviated L_1) is used for illustrative purposes.[2]

2.1.2 Syntax

The syntax of the languages in C is very limited. Strictly speaking, since these languages have a finite set of possible sentences, these languages would not need a grammar at all: an exhaustive list of all sentences would be sufficient to define the language. However, even if we ignore this issue, another question immediately arises: what type of grammar is appropriate? In principle, a simple FSG would suffice: since word order is completely fixed within each language, the syntax can be defined in a completely linear way. For example, in L_1, every sentence is a string consisting of a noun, followed by a modifier, followed by a verb, and finally another noun and a modifier. In figure 2.1, this is visualized in the form of a state diagram.

However, although FSGs are certainly powerful enough from a syntactic point of view, it is unclear whether they are also semantically interpretable. As far as I am aware, in the linguistic literature, interpretation procedures are always discussed in relation to context-free or more powerful grammars. In principle, there is no reason to assume that it would be impossible to define an interpretation procedure for a finite-state language, but it is unclear what such a procedure should look like. An obvious problem is that finite-state languages do not have a hierarchical constituent structure, which seems likely to make compositionality more complicated. For this reason, a more natural way of defining the language is to use a CFG. In table 2.2,

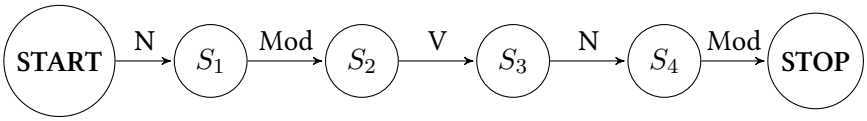
[1] This is trivially true since all it takes for two languages to be considered distinct is to have a different set of words. Since the phonological form of a word is an arbitrarily long string, of which infinitely many exist, there are also infinitely many distinct languages.

[2] This language predates my research for this thesis: I created it in collaboration with a fellow student, Marianne de Heer Kloots, for the purposes of another project (see `https://github.com/mdhk/TalkToAliens`). The syntax and lexicon of the language were adapted slightly for the purposes of the present project.

Table 2.1: Defining characteristics of the set of languages in C.

Domain	Feature	Characteristics
Syntax	Syntactic categories	Three distinct categories: 'noun' (N), 'verb' (V), 'modifier' (Mod)
Syntax	Phrase structure	Sentences consist of a subject-NP plus a predicate (verb and object-NP). NPs consist of a noun and a modifier. Finite number of possible sentences; no self-embedded structures.
Syntax	Word order	Fixed word order within each language, but no restrictions on specific word order. For sentences: SOV, SVO, VSO, OVS, VSO, VOS are all possible; for NPs: both Noun-Modifier and Modifier-Noun are possible.
Semantics	Utterance types	Only declarative sentences that describe the 'here-and-now'
Semantics	Domain	Only descriptions of spatial relationship between two animals and the spatial location and orientation of each animal.
Semantics	Lexicon	Limited number of concepts; each word denotes exactly one concept; no homonyms or synonyms

Figure 2.1: State diagram of a possible FSG for L_1



rewrite rules for a CFG generating L_1 are given, and a phrase structure diagram of the sentence *wu ba kepa zu po* is given in figure 2.2. Although this CFG is weakly equivalent to (generates the same set of strings as) the FSG in figure 2.1, the CFG captures the generalization that L_1 is a languages in which sentences have SVO order and NPs have N-Mod order while the FSG does not.

2.1.3 Semantics

Whereas the syntax of the languages in C quite dull, their semantics is what sets them apart from languages typically used in AGL experiments. In all languages in C, all sentences describe situations in a simple visual world. Each situation consists two objects, which have a particular spatial configuration relative to each other and relative to a river (which flows from east to west through the visual world). Objects can be either an animal (of one of three species: a blue duck, a red frog, or a yellow rabbit) or a chair. An example situation is depicted in figure 2.3.[3]

[3]Note that the only relevant elements in the spatial world are the two objects and the river. The rest of the scenery (the bridge, trees, mountains, etc.) is purely decorative.

Table 2.2: Rewrite rules of a possible CFG for L_1

Non-terminal rules	**Terminal rules**
S → NP VP	V → { *kaba, kapa, kabo, kapo, keba, kepa, kebo, kepo* }
VP → V NP	N → { *wu, zu, lu, du* }
NP → N Mod	Mod → { *ba, pa, bo, po* }

Figure 2.2: Phrase structure diagram for a sample sentence from L_1

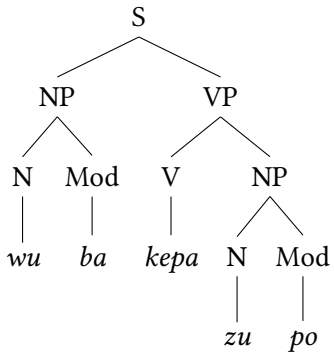


Figure 2.3: An example of a situation that can be described by the languages in C

Every sentence expresses three aspects of the situation: (i) the species of the objects in the situation, (ii) the spatial orientations of these objects relative to the river, and (iii) the spatial relationship between the two objects. Moreover, all languages in C share the same, fixed, set of concepts for describing situations. In contrast to languages like English, which express spatial relationships in 'egocentric' terms (for example, in a sentence like 'X is to the left of Y', the precise meaning of 'to the left' depends on the orientation of X), the languages in C use the river as a reference point for expressing spatial relationships (i.e., instead of 'X is to the left of Y', one would say 'X is located upstream of Y'). In this respect, these languages are similar to more 'exotic' languages like Ternate Malay or rGyalron (a language spoken in China), which also use geographical reference points for expressing spatial relationships (see Prins 2011; Litamahuputty 2012).

In order to be able to give a rigorous account of the meanings of each sentence, a logical model[4] is needed against which linguistic expressions can be evaluated. It is also necessary to introduce a form of intensionality[5] in order to account for the fact that reality is not static but can change (situations can differ with respect to the objects that are present, the spatial orientations of these objects and the spatial relationships between the objects) while still assigning a fixed meaning to every possible expression (for example, in L_1, *lu* 'duck' should always have the same meaning, regardless of the situation the word is uttered in). One way to do this is to define a separate model M_s for every possible situation s. Then, the fixed meaning of each expression in a language can be defined as an evaluation function (usually written as $[\![\cdot]\!]^M$) that takes in a logical model and outputs the semantic value (for example, a truth value in the case of a sentence) of the expression in that model. As an illustration, a model corresponding to the situation in figure 2.3 is given in table 2.3. Finally, in order to describe the relationship between syntax and semantics, a type system is needed. In the languages in C, there is a one-to-one correspondence between syntactic categories and semantic types. Table 2.4 lists these correspondences, along with examples from L_1.[6]

2.2 Psycholinguistic applications

There are many possible questions that could be asked about the acquisitions of languages such as those in C (or any other artificial languages with a formal semantics). However, from the perspective of the AGL literature, the most interesting questions have to do with the relationship between syntax and semantics: for example, does the presence of semantics make languages

[4] The model sketched here is loosely based on M_1 in Chierchia and McConnell-Ginet (2000, p. 124), but in a slightly simplified way: the variable assignment function g_1 is omitted (given the lack of real quantifiers in the languages of C, it is unnecessary to include it).

[5] In intensional semantics, linguistic expressions do not directly refer to the outside world but can be evaluated in different ways in different situations (or 'worlds'). To avoid the complexity of Possible Worlds Theory (PWT), which is the standard way of dealing with this, this thesis treats intensionality in a more simplistic way (cf. Chierchia and McConnell-Ginet 2000, p. 285) by simply assuming a separate model for each possible situation.

[6] The type system sketched here, which consists of the primitive types $< e >$ (entity/individual) and $< t >$ (truth value) and of functions combining other types, is based on the type system for a fragment of English described in Chierchia and McConnell-Ginet (2000, p. 89), but with adaptations to accommodate the specifics of the languages in C

Table 2.3: Logical model M_1 corresponding to the situation in figure 2.3

General
Let model M_1 be a pair $\langle U_1, V_1 \rangle$, where U_1 is a set of individual entities ('the universe'), and V_1 is a function that assigns to each predicate (representing basic meanings in the languages in C) an extension in U_1.

Universe
$U_1 = \{object\_1, object\_2\}$

Assignment function
$V_1(duck) = \{object\_1\}, V_1(chair) = \{object\_2\}, V_1(rabbit) = \emptyset, V_1(frog) = \emptyset$
$V_1(oriented\_upstream) = \{object\_1\}, V_1(oriented\_downstream) = \emptyset$ $V_1(oriented\_towards\_river) = \{object\_2\}, V_1(oriented\_away\_from\_river) = \emptyset$
$V_1(on\_same\_side\_of\_river) = \emptyset$ $V_1(further\_upstream\_than) = \{\langle object\_2, object\_1 \rangle\}$ $V_1(further\_downstream\_than) = \{\langle object\_1, object\_2 \rangle\}$ $V_1(further\_towards\_river\_than) = \{\langle object\_1, object\_2 \rangle\}$ $V_1(further\_from\_river\_than) = \{\langle object\_2, object\_1 \rangle\}$

Table 2.4: Syntactic categories and semantic types, with examples from L_1 (denotations are relative to M_1 (table 2.3))

Category	Type	Example	Intension	Extension
S	$< t >$	*lu po kaba wu ba*	$[\![\textit{kaba wu ba}]\!]^{M_1}([\![\textit{lu po}]\!]^{M_1})$	$\top$
NP	$< e >$	*lu po*	$[\![\textit{po}]\!]^{M_1}([\![\textit{lu}]\!]^{M_1})$	$object\_1$
VP	$< e, t >$	*kaba wu ba*	$[\![\textit{kaba}]\!]^{M_1}([\![\textit{wu ba}]\!]^{M_1})$	$\{object\_1\}$
N	$< e, t >$	*lu*	$\lambda x.duck(x)$	$\{object\_1\}$
Mod	$<< e, t >, e >$	*po*	$\lambda f.x \mid f(x) \wedge oriented\_upstream(x)$	–
V	$< e, < e, t >>$	*kaba*	$\lambda x.\lambda y.further\_towards\_river\_than(y, x)$	$\{\langle object\_1, object\_2 \rangle\}$

Note: The notation '$x \mid \ldots$' means 'the unique entity x such that ...'

the syntax easier to learn? And does it affect what kind of syntactic rules are acquired? As mentioned in the previous section, each of the languages in C can described by at least two types of grammars: FSGs and CFGs. Given that FSGs are computationally weaker (and were shown to be acquirable by both humans and some other animals), it could be hypothesized that learners, when confronted with one of these languages, would be inclined to learn the FSG rather than the computationally more complex CFG. On the other hand, the type of semantics sketched here seems to necessarily require some sort of constituent structure, which only the CFG provides. From a semantic perspective, another set of questions is also relevant: where do semantic structures come from? How does the type of formal semantics used in the previous section relate to the semantic knowledge that learners actually acquire?

This thesis does not try to answer any of these questions, but does try to set up (a first attempt at) an experimental framework for addressing these questions using a computer simulation (see chapter 3 onwards). This simulation starts with a number of hypotheses about the learning process and shows that under these hypotheses, it is possible for an IA to learn L_1 (and in principle any of the other languages in C) in a limited time frame and using relatively little input. While this, in itself, does not imply anything about the acquisition of such languages by humans, it could be a starting point for future, 'real' AGL experiments: for example, it would be interesting to see whether humans could learn these languages at all under the same circumstances, how much input and feedback they would need, and whether they would make the same kinds of mistakes.

3 The simulation I: a high-level overview

3.1 General characteristics

The previous chapter created the foundation for a new type of AGL experiment, introducing a class of artificial languages with a formally defined semantics, and a visual world that can be described by these languages. The present chapter builds on this foundation by showing how this new type of experiment can be simulated in a computer model involving two IAs: a 'child' trying to learn the language L_1, and its 'parent', who already possesses full knowledge of this language and who assists the child by providing example sentences and by offering feedback on the child's own utterances. A working version of this model can be found online at `https://gossminn.github.io/AlienLearningAgents/`; it can run in any modern web browser. See chapter 5 for information on how to use the user interface.

3.1.1 The agents

The two types of agents found in the simulation share a number of characteristics. First of all, they are 'grounded', which means that they do not exist in a vacuum but are situated in space and time and have access to information about their surroundings. However, this groundedness is quite limited, especially when compared to the agents used in other simulations involving spatial language. For example, the agents in Spranger and Steels (2012) are physical robots and make use of cameras for gathering information about their surroundings. The model developed here takes a radically different approach: it makes use of virtual, rather than physical, agents, and these agents have access to their surroundings in a much more abstract way which does not involve any type of visual perception. Instead, perception is entirely 'concept-driven': the agents have knowledge of a certain number of basic concepts, and can, at any given point in time, determine which of these concepts are applicable in the current situation. For example, the agents both have a concept of 'rabbit', and know whether or not the present situation involves at least one rabbit.

Other characteristics shared between both types of agents include that they both 'live in the moment': they perceive only one situation at a time, and cannot remember the past or predict the future.[1] Moreover, the agents have very limited communicative abilities: they can only produce true sentences describing the current situation. This type of communication is effectively useless for the purposes of information transmission (the parent cannot tell the child anything it did not already know, or vice versa), but is completely geared towards language acquisition: the parent speaks to provide the child with feedback, and the child speaks so that the parent can evaluate its progress.

[1] It is important to note here that although neither of the two agents has direct access to past or future situations, for the child agent, past experience does, of course, play an important role (without some sort of memory, learning would be impossible).

The obvious difference between the two types of agents is that while the parent already possesses complete representations of the syntax and semantics of the target language from the start of the simulation, the child does not initially have access to any information specific to the target language, but is equipped with a learning information for acquiring these representations. However, the mechanisms used for storing and using linguistic representations are identical for both the parent and the child agents (see chapter 5).

3.1.2 Parent-child interaction

The simulation consists of a series of situations (such as the one depicted in figure 2.3). During each situation, the two agents will communicate with each other about that situation. This takes place in three turns. In the first turn, the *parent turn*, the parent utters a sentence which accurately describes the current situation (for each situation, there are between two and eight sentences that are true in that situation; the parent randomly selects one of these). Then, in the *child turn*, the child uses the sentence provided by the parent to update its internal representations of the vocabulary, syntax, and semantics of the language. If the child feels 'confident' enough about its knowledge of the language, it then produces an utterance of its own; otherwise, it remains silent. The third turn is the *evaluation turn*. This only takes place when the child has said something; the parent evaluates the child's sentence and determines whether it is an accurate description of the situation. If this is the case, the parent is 'happy' (positive feedback), and if not, it is 'angry' (negative feedback). Finally, the simulation moves on to the next situation.

3.2 The learning process

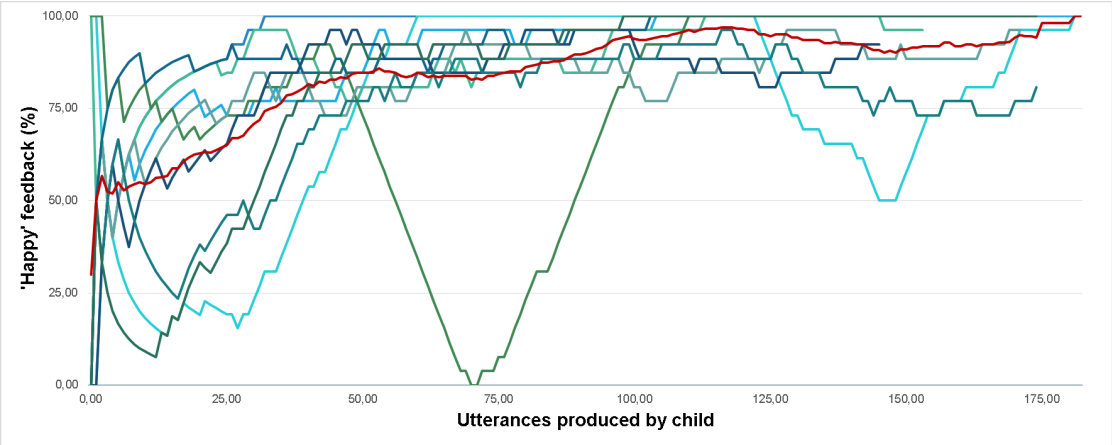
3.2.1 Overview

The child's goal is to acquire its parent's language. If learning is successful, then that means that at some point, the child's linguistic competence should be equivalent to that of the parent, and it should stop making mistakes. Figure 3.1 shows that the number of mistakes does indeed decrease over time, and that in most cases the positive feedback rate has converged to 100% within 200 sentences uttered by the child (and in some cases even within 35 sentences).[2] However, note that the learning process is not linear: the child sometimes has to revise or even completely reset its knowledge base, which leads to setbacks in its feedback scores.

In order to acquire the language, the child has to complete three tasks: it must categorize the words that it encounters into syntactic categories, infer phrase structure, and assign meaning representations to both individual words and to higher-level constituents. The approach that the child takes to address these problems mixes parallel ('learn everything at the same time') and serial ('complete learning in one area, then move on to the next') strategies. The parallel aspect of this approach comes from the fact that every time that the child gets an input sentence, it uses the new information to update its knowledge in each of the three areas (categories, phrase structure, and semantics). However, since these areas are often dependent on

[2] Figure 3.1 shows that this is not the case in three out of ten test cases. However, it is very likely that if the simulation would have been allowed to run longer, these cases would have reached 100% as well.

Figure 3.1: Percentage of the last 25 utterances by the child that were correct (prompting 'happy' feedback from parent). Different shades of blue represent different test runs of the simulation; the red line represents average development.



each other, some degree of serialism is also required. For example, when the child decides to remove or add a syntactic category, this causes its knowledge about correspondences between syntactic categories and semantic types to become invalid. The child solves this by 'resetting' its semantic knowledge. This means that semantic learning is always completed later than category learning.

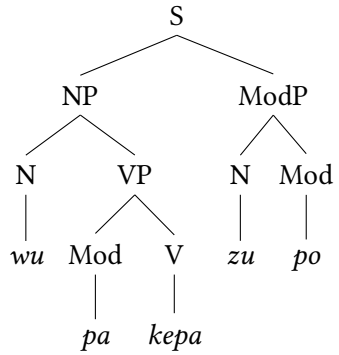
3.2.2 Syntactic categories

Syntactic categories are postulated purely on the basis of distributional evidence: for example, the child can infer that the verbs *kaba* and *kapo* belong to the same category because they always appear to the right of a modifier (such as *po*) and to the left of a noun (such as *wu*). In doing this, the child assumes that if two words have the same form, they are identical (and therefore belong to the same category).[3]

Initially, the child does not have any knowledge about which categories exist in the target language or even about how many categories there are. Moreover, the categories that are eventually acquired are merely labels for identifying which words 'belong together', and have no direct link to semantics (although they do play a role in the acquisition of semantics, see section 3.2.4). Thus, the learning model developed here contrasts sharply with the so-called Semantic Bootstrapping Hypothesis (see Pinker 1996, p. 40, for arguments in favor of this hypothesis), which holds that language learners have an innate set of semantically defined categories and only have to link words that they acquire to these preexisting categories.

[3] An earlier version of L_1 turned out to be unlearnable under this assumption given that in that version, all locatives could also be used as verbs. Since modeling such complex word class phenomena was never the main goal of this thesis, it was decided to adapt the language slightly.

Figure 3.2: Alternative phrase structure diagram for a sample sentence from L_1



3.2.3 Phrase structure

The target language that the child agent learns can in principle be described by a FSG, but for semantic reasons it makes more sense to represent its grammar using a CFG (see section 2.1.2). The child does not have any initial knowledge about the phrase structure of the target language it learns, but is equipped with 'templates' for postulating phrase structure rules (i.e., it knows *that* there are phrase structure rules, but not *which* ones). Additionally, it makes a few basic assumptions about the properties of phrase structure rules. For example, it assumes that all rules are binary (this restriction is implicit in the way that the templates are structured) and that they always continuous (the child will only try to group together constituents that are adjacent to each other). Finally, there is only one level of syntactic representation: the child does not distinguish between levels such as 'deep structure' and 'surface structure' (and does not need to do so given the rigidity and simplicity of the language that it learns).

In contrast to how it acquires syntactic categories, when acquiring phrase structure rules the child uses semantic rather than distributional evidence when postulating constituents. The reason for this is that, because of the extreme rigidity of the syntax of the target language (all sentences have exactly the same syntactical structure), almost no distributional cues exist that could be used for choosing between different possible syntactic structures. To a human observer (especially one with linguistic training!), the phrase structure diagram in figure 2.2 might appear to be the (only) obvious way to represent the syntax of L_1, but in fact, from a purely syntactic point of view, something like figure 3.2 would be equally valid. However, defining a compositional semantics based on such a structure would likely be extremely complicated.

To prevent these kinds of problems, the child uses semantic evidence to postulate phrase structure rules: it assumes that two constituents can for a higher-level constituent if and only if their semantic types are compatible. For example, once the child has determined that *wu* has type $< e, t >$ and *pa* has type $<< e, t >, e >$, it can infer that, since the meaning *pa* is a function that takes the meaning of *wu* as an argument, the two can form a constituent of type $< e >$. By contrast, *pa* and *kepa* have incompatible types ($<< e, t >, e >$ and $< e, < e, t >>$,

respectively) and cannot form a constituent.[4] This process works in a 'bottom-up' way: the child starts with individual words (e.g., *wu* and *pa*) and tries to group them into higher-level constituents (e.g., NP), and then moves on to the next level and tries to combine these constituents into even larger constituents (e.g. S); this process is repeated until the entire sentence is dominated by a single node.

3.2.4 Semantics

While the child IA has little to no 'innate' knowledge about the specific phrase structure rules and syntactic categories of the target language, this is not entirely true for semantic learning. Although the child does not initially possess any representation of the meanings of words or sentences, it does possess a (partially hard-coded) set of concepts, each of which corresponds to the meaning of exactly one word in the target language and has a semantic type. For example, at the start of the learning process, the child already knows that there exists a concept of 'rabbit', and that this concept has type $< e, t >$; all it has to do is find out which word in the target language denotes this concept. Although this assumption about the learning process might seem very unrealistic, there are several reasons to believe that it might be quite reasonable.[5]

First of all, there is reason to believe that concepts, even if they are not innate, are acquired separately from language. Hurford (2014) argues that while it has often been claimed that concepts are necessarily "bound up with language", "[this] view is now substantially eroded" (p. 61). According to Hurford, it is likely that non-human animals (although lacking language) possess concepts of some sort, and that humans had concepts (or "proto-concepts") before they evolved language. Part of the evidence that he uses to back up this claim comes from language acquisition experiments with chimpanzees and other non-human primates that were conducted from the 1960s onwards. While these experiments failed to show that apes could learn syntax, they did show quite convincingly that they can learn individual word meanings, and the concepts attached to them (for a recent study, see Beram and Heimbauer 2015). Thus, it seems plausible to assume that concepts are at least partially independent of language and could be present at the start of the language acquisition process.

Second, the assumption that concepts have semantic types seems to be very theoretically loaded and appears to contradict the idea that concepts are independent of language. However, this is not the case: arguably, the types used in formal semantics in fact have a high degree of theory-neutrality and are independent of language. The type system used here (which is based on that used in Chierchia and McConnell-Ginet 2000) is based on the basic types $< e >$ (entity) and $< t >$ (truth value) and functions combining these types. These two notions are

[4] In principle, this would be possible if the child would be allowed to posit new semantic types (which is not possible in the model developed here). For example, if the child already knew that the combination of *pa* and *kepa* should have type $< t >$, it could change the type of *pa* to $<< e, < e, t >>, t >$. However, because the child uses a bottom-up approach for figuring out phrase structure, it does not know the resulting type in advance and thus has no basis for positing new types. Additionally, preventing the creation of new types keeps the learning process as possible and makes sure that the type system does not become unnecessarily complex.

[5] As mentioned in section 1.2, this thesis does not make any empirical claims about language acquisition. However, in order for the semantic approach to AGL (and the computational model attached to it) developed here to be credible, it is necessary that the assumptions about the learning process that are made are at least somewhat plausible.

extremely uncontroversial and well-grounded outside of linguistic theory: they merely imply that there are 'things' that exist and that there exists a distinction between truth and falsity.[6] The same goes for functions: these are simply ways to link the basic notions of entities and truth values to each other in various ways. If someone is said to possess knowledge of a concept of type $< e, t >$, this simply means that they have the ability to produce a value of type $< t >$ (i.e., say 'yes' or 'no') when given something of type $< e >$ (i.e., some kind of object in the outside world). Thus, the fact that the child agent in the model developed here is 'born' with the concept of 'rabbit' only implies that the child can distinguish between rabbits and non-rabbits, which does not seem to be an outrageous hypothesis.

[6] Additionally, there is reason to believe that the distinction between truth values and entities is quite fundamental and plays an important role in the learning process: once a child realizes that the meanings of a name ('Daddy') and a sentence ('Daddy drinks coffee') have a very different nature, it can start figuring out the relationships between these meanings (e.g. if 'Daddy' has type $< e >$ and the entire sentence has type $< t >$, then perhaps 'drinks coffee' has type $< e, t >$) (Crit Cremers, personal communication, June 2017).

4 The simulation II: representations and algorithms

4.1 Overview

The learning model developed here is symbolic in nature: knowledge is represented in an explicit way, and the representations used are based on those developed in formal linguistics. Thus, the way in which the syntax and semantics of L_1 are stored in the parent's (and eventually also the child's) 'mind' is very similar to how how they were described in chapter 2 of this thesis. By contrast, in a subsymbolic model (e.g., an artificial neural network), knowledge would be stored in a very different, 'distributed' way (e.g., in the form of activation patterns of artificial neurons) that would not necessarily bear any resemblance to how theoretical linguists believe linguistic knowledge is structured. Although subsymbolic models are becoming increasingly popular and powerful, when it comes to simulating an AGL experiment, a symbolic approach has at least one major advantage: it makes it possible to explain what exactly the child has learned, and to compare this to the father's knowledge.

Both the child and the parent IA store their linguistic representations in a so-called *knowledge set*. A knowledge set consists of four parts:

1. *Category set:* stores information about syntactic categories;
2. *Rule set:* stores phrase structure rules (including information about meaning composition);
3. *Vocabulary set:* stores the agent's lexicon;
4. *Meaning hypothesis set:* stores hypotheses about word meanings (only relevant for the child agent).

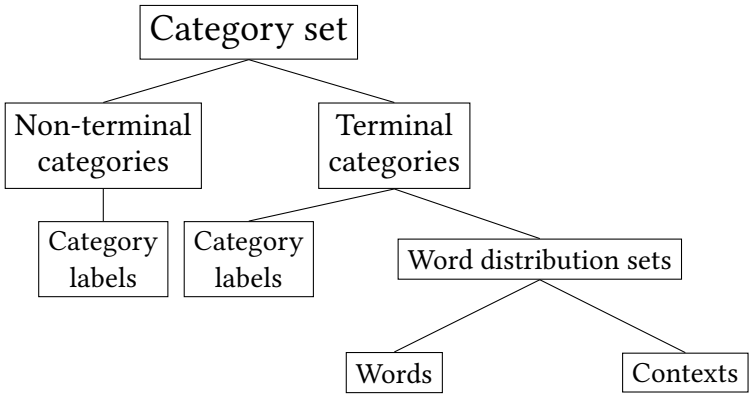
The remainder of this chapter explores these four domains in more detail. Section 4.2 explains the structure of the different types of representations and the relationships between them, and section 4.3 describes the algorithms that the child uses to acquire them.

4.2 Knowledge representation

4.2.1 Category sets

Category sets (schematized in figure 4.1) are used to represent the syntactic categories of the target language. A distinction is made between terminal categories (e.g., Noun) and non-terminal categories (e.g., Noun Phrase). The representation of a non-terminal category is simply a label (with a numerical identifier) that can be used in rewrite rules or in phrase structure trees. For the parent agent, the representation of a terminal category is also only a label, but

Figure 4.1: Structure of category sets



for the child agent, it also includes information about the word distributions that each label is based on. This information is stored in a *word distribution set* which consists of a collection of words, along with the collections of words that can occur to the left and to the right, respectively, of these words. Another difference between the parent's and the child's category sets is that the child has two separate sets of terminal categories at any given time, a more conservative (or 'raw') and a more progressive (or 'generalized') one, while the parent has only one, definitive, set. The reason that this is the case has to do with how the learning process works; see section 4.3.1.

4.2.2 Rule sets and phrase structure trees

Knowledge about how syntactic and semantic composition work are stored in rule sets (see figure 4.2 for a schematic overview). Rule sets consist of three parts: a collection of terminal rules, a collection of non-terminal rules, and a reference that marks which of the non-terminal rules is the 'root rule' (i.e., the rule corresponding to the highest node in a phrase structure tree, e.g. S $\rightarrow$ NP VP). A terminal rule (e.g. N $\rightarrow$ { *wu, zu, lu, du* }) is simply a mapping between a terminal category label and a collection of vocabulary items. Non-terminal rules are somewhat different: they are mappings between a non-terminal category label on the one hand and two other category labels (all rules are binary) on the other hand, along with a label indicating the direction of semantic composition. For example, the agents' representation of the rule NP $\rightarrow$ N Mod also includes the information that the semantic value of Mod is a function that takes in the semantic value of N as its argument.

The rules in rule sets are used for parsing and generating phrase structure trees. Like rules, trees are always binary, and consist of two types of nodes: terminal nodes and non-terminal nodes. Terminal nodes are simply vocabulary items, while non-terminal nodes consist of a pair of branches (each of which can be either terminal nodes or non-terminal nodes), along with a semantic representation and a category label.

Figure 4.2: Structure of rule sets

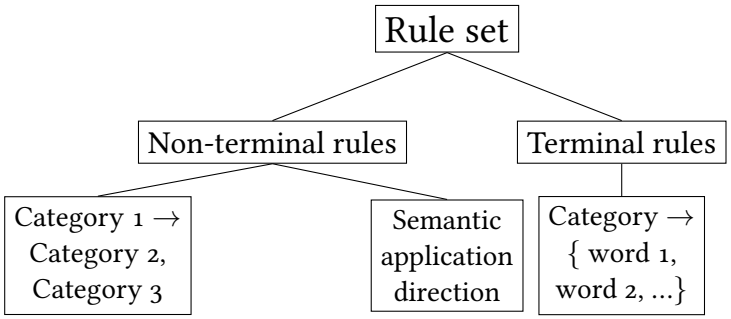
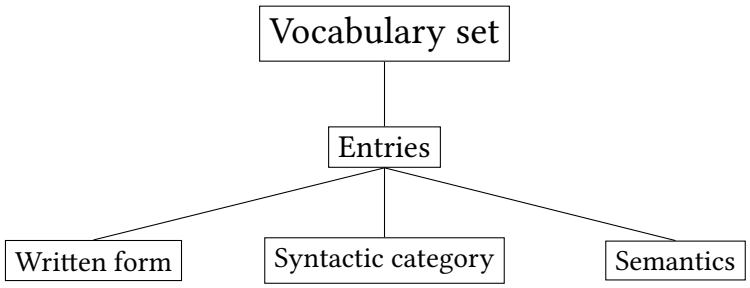


Figure 4.3: Structure of vocabulary sets



4.2.3 Vocabulary sets

Vocabulary sets (see figure 4.3) are used to store the agents' lexicons. They are simple collections of vocabulary items, each of which represents a single word and stores information about the word's written form, its syntactic category and a representation of its semantics. The parent's and the child's vocabulary sets have exactly the same structure.

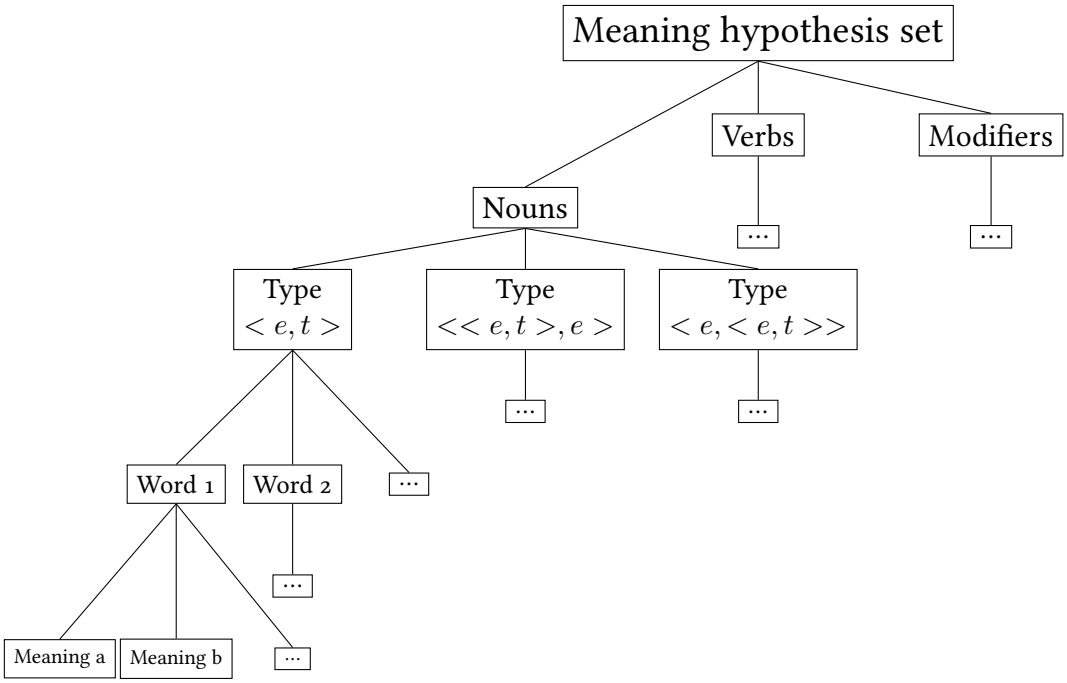
4.2.4 Meaning hypothesis sets

Both individual vocabulary items and nodes in phrase structure trees have a semantic representation. As explained in section 2.1.3, in order to be able to assign a fixed meaning that is valid across different situations to an expression, intensional representations should be used. For example, the extension of a verb is a set of pairs of entities with a particular relationship, but in different circumstances, different relationships exist between different entities. Thus, the fixed meaning of a verb should not be a single set of pairs, but a function that takes in a logical model and returns the set of pairs that the verb denotes under that logical model.

The learning model described here also adopts this approach, but only for vocabulary items: the same vocabulary items are used in different situations, and should therefore have a fixed meaning, but larger units (such as noun phrases or sentences) are always constructed in the context of a particular situation. Thus, it is possible to represent their meaning using 'simple' semantic values rather than using interpretation functions.

Another difference between the semantics of vocabulary items and of larger constituents

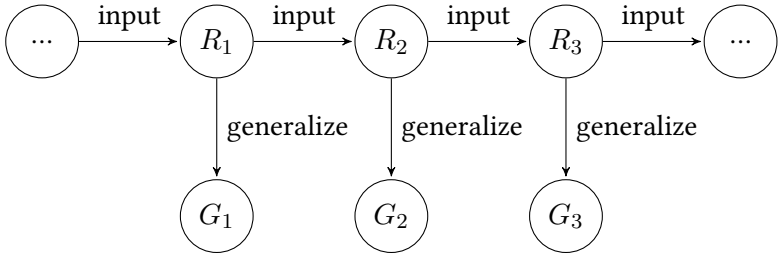
Figure 4.4: Structure of meaning hypothesis sets



lies in how they come in to existence: while the latter are dynamically constructed from their component parts, the former have to be acquired and stored in some way. For the parent agent, lexical meanings are simply pre-programmed, but the child agent has to form its own hypotheses about what each word means.

The learning algorithm it uses to do this (see section 4.3.3) requires a structured representation of the hypothesis space. This is provided in the form of meaning hypothesis sets (see figure 4.4). A meaning hypothesis set is organized on several levels. First, it is organized by syntactic category: hypotheses about the meanings of nouns are stored separately from those about the meanings of verbs and modifiers. Second, within each category, hypotheses are organized by semantic type: the child does not know in advance which syntactic category corresponds to which semantic type, but does assume that such correspondences exist. Therefore, it is sensible to separate hypotheses according to which (for example) noun meanings have type $< e, t >$ from those according to which they have type $< e, < e, t >>$. Finally, meaning hypothesis sets are organized per word: for each category and for each type, it can store a set of possible meanings of that type for each word in that category.

Figure 4.5: Update cycle of raw (R) and generalized (G) category sets



4.3 Learning algorithm

4.3.1 Category learning

Every time that it receives a new input sentence, the child updates its knowledge about terminal categories.[1] This process consists of two steps: first, the child uses the new input to update its 'raw' category set, which contains groups of words of which the child is absolutely sure that they belong to the same category. Next, it tries to generalize over this raw category set by comparing every category in the raw category set to every other category set and merging similar categories. The result of this stored in a second, 'generalized' category set.

Thus, at any point in time, the child agent has two category sets in memory. New information is always added to the existing raw category set and then the new version of the raw category set is used to produce a new generalized category set. This means that only the raw category set is accumulative in the sense that every new version is based on the previous version, while each new version of the generalized category set is based on the latest raw category set rather than on the previous version of the generalized category set. The main advantage of this is that only knowledge that is guaranteed to be correct is accumulated over time, which prevents overgeneralization. This dynamic is schematized in figure 4.5.

Raw category sets are updated as follows: first, the input sentence is split into a set of trigrams (each consisting of one of the words in the sentence and the words to its immediate left and right). For example, the sentence *wu ba kaba zu pa* becomes $\{\langle$ *#, wu, ba* $\rangle, \langle$ *wu, ba, kaba* $\rangle, \langle$ *ba, kaba, zu* $\rangle, \langle$ *kaba, zu, pa* $\rangle, \langle$ *zu, pa, #* $\rangle\}$ (where # indicates a word boundary). Then, these trigrams are compared to the previous raw category set. If a particular word is already known, the context of that word is added to the word distribution set of the category containing the word. For example, suppose that the raw category set R_1 contains a terminal category with the words *ba* and *po* and the context $\{$*zu, du* $\}$ _ $\{$*kepa, kabo*$\}$. Then, after processing the trigrams listed above, the context of *ba* in the input sentence will be added to this distribution set, so that the next version of the category set (R_2) will contain the updated context $\{$*zu, du,* ***wu*** $\}$ _

[1] Due to the simplistic nature of the target language, the child can make a very rigorous distinction between terminal and non-terminal categories. This might not be possible with more complex target languages: for example, in a natural language like English, proper names (a terminal category) have the same distribution as NPs/DPs. In order to be able to recognize this, it is probably necessary to allow for some degree of overlap between terminals and non-terminals.

{*kepa, kabo,* ***kaba***} for this category.[2] However, if a word in the input sentence is not yet part of a category but has a known context, the word will be added to appropriate category in the category set. For example, if there is a category with context {*zu, du* } _ {*kepa, kabo*}, and the unknown word has context *zu* _ *kabo*, then the word will be added to that category. Finally, if a word in the input sentence is unknown, and if its context does not overlap with the context of any existing category, then a new category is created with a distribution set containing only that word and its context.

In this way, the child will only group words together that really belong together, but it will miss generalizations and posit too many distinct categories. This problem is solved in the generalization step: all categories in the raw category set are compared pairwise, and if two categories are found to be similar enough they are merged into one, new category. Then, the categories that result from this are again compared to each other and merged if appropriate; this process is repeated until no further merging is possible. Two categories are considered 'similar enough' if they have any overlap in either their left-hand or their right-hand context. For example, if one category contains the words *pa* and *bo* with the context {*zu, du* } _ {*kepa, kabo*}, and if another category contains the words *po* and *ba* with the context {*wu*} _ {*kaba*}, then these two categories will be merged, resulting in a new category containing the words *pa, bo, po,* and *ba* with context {*zu, du, wu* } _ {*kepa, kabo, kaba*}. This enables the child to make the correct generalizations and to (eventually) reduce its number of categories to three.

4.3.2 Phrase structure learning

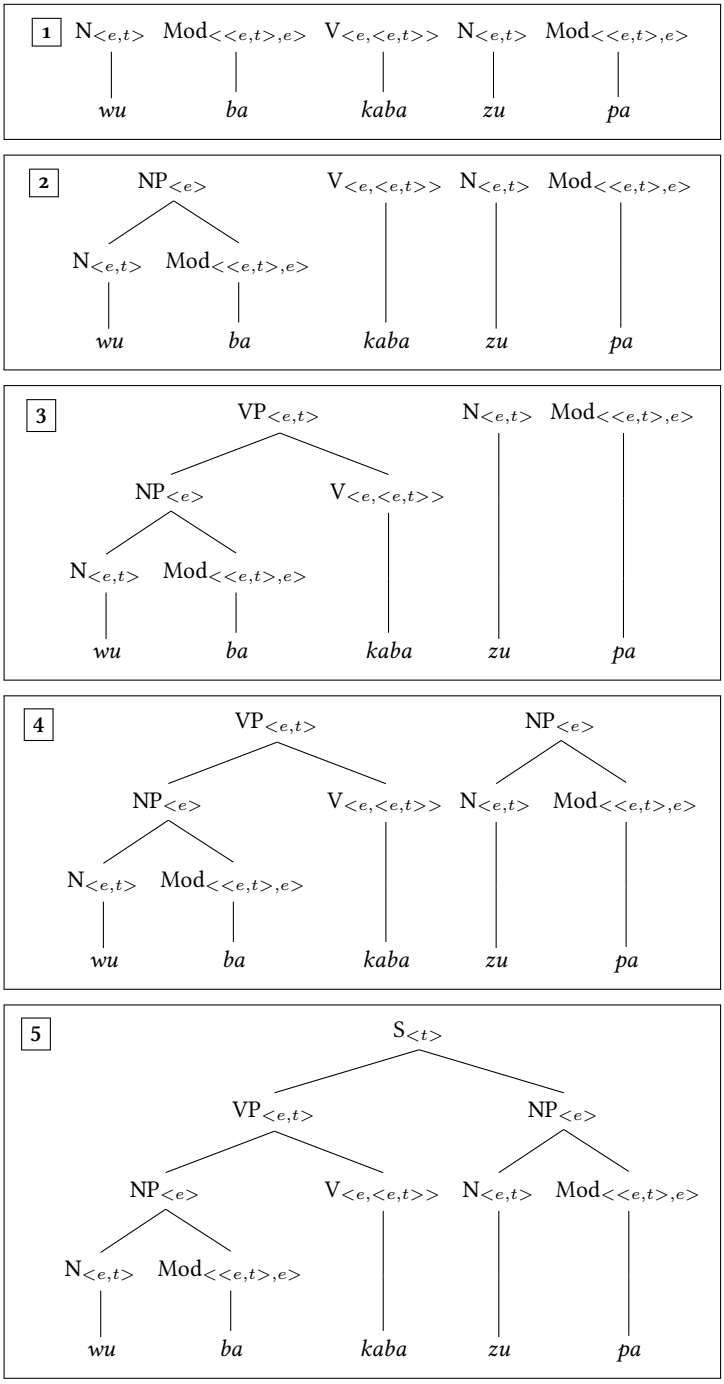
As explained in section 3.2.3, phrase structure learning is guided by semantics. Hence, this process cannot start until the child is reasonably 'confident' about its hypotheses about word meanings and is ready start producing utterances of its own (see section 4.3.3). However, once that is the case, the phrase structure learning process itself is relatively straightforward: since the syntax of the target language is completely rigid, so all that the child has to do is find phrase structure rules for a single input sentence from its parent, which can then be used to generate and parse all other possible sentences in the target language.

In order to do this, the child first retrieves category labels for each of the words in the input sentence from its category set. Next, it constructs phrase structure rules from the bottom up: it starts with the individual words and works its way up until the entire sentence is subsumed under a single node. An example of this process (with the input sentence *wu ba kaba zu pa*) is shown in figure 4.6.[3] As can be seen in the figure, at every step at the process the child tries to compose the two leftmost constituents. For example, at step 1, the two leftmost constituents are *wu* and *pa*. Since *wu* has type $< e, t >$ and *ba* has type $<< e, t >, e >$, these two can be successfully composed (with *ba* as a function and *wu* as its argument), prompting the child to postulate a new non-terminal category and a phrase structure rule (with a right-to-left semantic application direction) linking this category to the labels of *wu* and *ba*.

[2] The notation $\{w, x\}$ _ $\{y, z\}$ means that every word in a particular category can occur to the right of w and x and to the left of y and z.

[3] Note that the category labels (N, VP, S, ...) used are for illustrative purposes only: for the child, category labels are nothing more than meaningless numerical identifiers.

Figure 4.6: Phrase structure learning algorithm



In some cases, composition is unsuccessful. For example, at step 3, the child will first try to compose *[wu ba kaba]* with *zu*, which does not work since both constituents are of type $< e, t >$. If this happens, the child will move to the next pair of constituents (*zu* and *pa*, in this case) and try to compose these instead.

At step 5, the child has constructed phrase structure rules that subsume the entire sentence, and so phrase structure learning is complete. The child's syntactic (and to some extent, semantic) competence is now equivalent to that of the parent. However, equivalence does not entail equality: for example, compare step 5 of figure 4.6 to figure 2.2, which shows the parent's representation of phrase structure. While the child has acquired the rule S → VP NP, the 'original' rule used by the father is S → NP VP. In other words, the child has acquired an OVS language while its parent speaks an SVO language. This happens because the child always works from left to right and will form a VP from the first pair of an NP and a V that it encounters. However, this discrepancy is largely inconsequential: the child is still able to produce exactly the same sentences, with exactly the same truth conditions. However, the different order of the NP and the VP does have an interesting consequence for lexical meanings: while the parent's meaning representation for *kaba* is $\lambda x \lambda y.further\_towards\_river\_than(y, x)$, the child must acquire the exact opposite (or the 'passive' version) of this: $\lambda x \lambda y.further\_from\_river\_than(y, x)$[4] in order to arrive at the correct truth conditions. The reason that this always works is that for every verb in L_1, there is another verb with the opposite meaning. An interesting question would be whether this is also the case in the real world; for example, if humans have a concept 'X sees Y', do they automatically also have 'Y sees X' (or 'X is seen by Y')?

Another difference between the parent's and the child's phrase structure rules is that the child sometimes posits 'duplicate' rules. For example, each sentence contains two NPs. While the parent has a single, pre-programmed category NP and a single rule NP → N Mod, the child has two of each: at step 2, it posits them for the first time, and when it gets to step 4, it does not realize that a rule rewriting to the sequence 'N Mod' already exists and simply posits a new rule and a new non-terminal category. It is possible to modify the learning algorithm to avoid this, but doing so would make it more complex (and less efficient) and is unnecessary because having duplicate categories never causes problems.

4.3.3 Semantic learning

Arguably, lexical semantics acquisition is the most important part of the learning process described here. Since the child is already equipped with a set of basic concepts, its main task is to find the correct concept for each word in the target language. The child uses two sources of information for doing this: it can use input sentences it gets from the parent and compare these to its own conception of reality, and it can produce sentences of its own and ask its parent for feedback.

In both cases, what the child does is formulate hypotheses about what each word in each lexicon might mean and then test these hypotheses against the evidence that is available. When interpreting input sentences from the parent agent, it does this by looking at the logical model of the current situation and determining which concepts are 'active', i.e., denote a non-empty

[4] A different but equivalent way of writing this would be $\lambda x \lambda y.further\_from\_river\_than(x, y)$.

set. For example, in a situation with a rabbit and a chair, the concepts $\lambda x.rabbit(x)$ and $\lambda x.chair(x)$ denote the sets $\{object\_1\}$ and $\{object\_2\}$, respectively, while the other species-denoting concepts (e.g. $\lambda x.duck(x)$) denote $\emptyset$. The child can use this information to falsify incorrect hypotheses (but not to confirm correct hypotheses). For example, if the concept $\lambda x.duck(x)$ is inactive in the current situation, and if the input sentence contains the word *zu*, then the hypothesis that *zu* means 'duck' must be false: if there are no ducks in the situation, then none of the words used for describing that situation can mean 'duck'. However, the (correct) hypothesis that *zu* means 'rabbit' cannot be confirmed in this way: since $\lambda x.rabbit(x)$ is only one of the concepts that are active, the only conclusion that can be drawn is that it might be correct, not whether this actually is the case.

An important question is how the child should formulate its hypotheses in the first place. One possibility would be to randomly choose associate each word with a meaning, and when one of this hypotheses is falsified, randomly generate a new one. However, there are two main problems with this approach. The first one is that the child would have no way of knowing when it would have found the correct meaning for a given word: if a particular hypothesis is consistent with the logical model, that means that that hypothesis might be correct, but if only one hypothesis is considered at a time, it is impossible to know whether there are any other hypotheses that could also be true. The second problem is that considering only one hypothesis at a time is not very efficient. These problems can be solved by taking a different approach, in which all possible hypotheses are considered simultaneously. Under this approach, a hypothesis is generated for each possible word-meaning pair, and these hypotheses are stored in a meaning hypothesis set. This happens in a structured way: first, all possible hypotheses about correspondences between semantic types and syntactic categories are generated (examples of such hypotheses are 'nouns denote species' [correct] or 'modifiers denote spatial relationships' [incorrect]). For each of these hypotheses, subhypotheses are generated for all combinations of words and meanings with the appropriate categories and types.

Then, every time that the child receives a new input sentence, it tests all of the hypotheses about the words in the input against the logical model and removes hypotheses that are falsified from the meaning hypothesis set. In this way, it is possible to eliminate large numbers of incorrect hypotheses at once, which speeds up the learning process. The child can now also determine whether or not is has already found the correct hypothesis for a given word: if there is only one possible hypothesis left for that word, then that hypothesis must be the correct one.

Using this elimination process, the correct meanings for nouns and modifiers can usually be found after only a small number of input sentences. However, verbs are more difficult: since for every verb, there is another verb with the opposite meaning, in any given situation there will be at least two verbal concepts that are active. For example, if it is the case that $[\![\lambda x \lambda y.further\_from\_river\_than(y,x)]\!]^M = \{\langle entity\_1, entity\_2\rangle\}$, then it must also be true that $[\![\lambda x \lambda y.closer\_to\_river\_than(y,x)]\!]^M = \{\langle entity\_2, entity\_1\rangle\}$. Thus, when only considering words in isolation, without regard for syntactic context, it is never possible to eliminate all of the incorrect hypotheses for verbs.

The child can solve this problem by trying to formulate sentences of its own. In order for this to be successful, it is necessary that the child already knows which syntactic category corresponds to which semantic type: if this would not be the case, the phrase structure learning

algorithm (see section 4.3.2 above) would not work. Once this condition is met, the child has to decide which word meanings it wants to use to generate an utterance. This is done by producing a 'guess' based on the current state of the meaning hypothesis set: each of the hypotheses that has not yet been eliminated is assigned a score (initially, this will always be 0), and for each word, the hypothesis with the highest score (or, if multiple hypotheses have a shared first place, a random one from these hypotheses) is chosen. Next, these meanings are used to construct phrase structure rules, which the child can then use to define the set of possible sentences in the target language. Then, the child calculates truth values for each possible sentence using the guessed meanings, and the child randomly selects one of the sentences it thinks is true and utters it.[5] After this, the parent evaluates the sentence its child has uttered and announces its opinion about it. If the parent agent agrees that the sentence was true, it indicates that it is 'happy'; otherwise it is 'angry'.

The next step is that the child uses this feedback to evaluate its hypotheses. The main problem lies in determining, if the feedback was negative, which part (or parts) of the guess that was produced caused the resulting sentence to be false. For example, suppose that the child's utterance included the words *zu* ('rabbit') and *kaba* ('be closer to the river than') and that for both of these words there are several hypotheses that have not yet been eliminated. If the utterance prompted negative feedback, this does not necessarily mean that both of the guesses about the meanings of these words were wrong, but only that one of them is (assuming that the child guessed the correct meanings for the other words in the sentence). Thus, the child cannot use the negative feedback to eliminate any of the hypotheses that it guessed were correct. Likewise, if it received positive feedback, this does not entail that the guessed meanings were all correct: it is possible to produce a true sentence by accident.

One possible way around this problem is to use scores: if a certain utterance prompted negative feedback, then the scores of all of the guesses that were used to produce that hypothesis are decreased (possibly becoming negative), and if the feedback was positive, the scores are increased.[6] This can sometimes result in hypotheses that were correct having their scores decreased and in incorrect hypotheses having their scores increased. However, over time correct hypotheses should have higher scores than incorrect ones: if a certain hypothesis consistently (i.e. after repeatedly being used to produce utterances) leads to negative feedback, then the likelihood that it is actually correct will become increasingly small. Likewise, if hypotheses consistently lead to positive feedback, then the probability that it actually was correct increases. A downside of this approach is that it can never definitively determine whether or not a certain hypothesis is correct or not. However, the score distributions will stabilize over time: once the correct set of hypotheses is found, feedback will always be positive and the scores of these hypotheses will always increase, which in turn means that they will always be selected in subsequent guesses. Thus, while the child will never 'officially' decide which hypotheses were correct, from a certain point it will never change its guess again, which effectively means that the learning process is complete.

[5] If there are no true sentences, a new guess is produced and the process is repeated.

[6] In the current version of the learning model, scores are always increased or decreased by one point. An interesting follow-up experiment would be to test what happens when this is differentiated, for example when the 'bonus' for positive feedback is larger than the 'penalty' for negative feedback.

In principle, the way in which the semantic learning process as described in this section works should always lead to performance improving, rather than getting worse. The test runs discussed in the previous chapter (see figure 3.1) largely confirm this, but there are a few notable exceptions: the graph shows that in the early stages of the learning process, there is a lot of fluctuation in performance, and also that major setbacks sometimes occur, even in relatively late stages of the process. Small fluctuations are often caused by the fact that there is an element of randomness in the guessing process, which can lead to the child agent accidentally selecting incorrect hypotheses. However, larger decreases in performance have a different cause: every time that the child's category sets change, the semantic learning process has to be reset. The reason for this is that meaning hypothesis sets are structured by syntactic category, and when the syntactic categories change (i.e. when two categories are merged, or when a previously unknown word is added), all hypotheses about correspondences between syntactic categories and semantic types become uncertain. In principle, it should be possible to find a way around this (if a single word is added to a category, this should not lead to all prior hypotheses about words in that category being discarded), however, because of the way that the category learning algorithm works, this is by no means trivial.[7]

[7] The main difficulty is that, as shown in figure 4.5, each time that new information is added, a new generalized category set is generated 'from scratch'. This makes it hard to identify categories in different versions of the generalized category set with each other.

5 The simulation III: implementation

5.1 Visualization: Unity3D

5.1.1 Overview

Because the simulation developed here is all about spatial semantics, it makes sense to make use of visualizations in order to be able to get a sense of what is happening inside the model. The software that generates these visualizations partially predates this thesis: for an earlier project,[1] a fellow student (Marianne de Heer Kloots) and I designed an artificial language (of which L_1 is a slightly modified version; the original was simply called 'the alien language') as well as a virtual world for visualizing the meanings of the sentences in this language. This program was intended as a 'serious game' which could be used as part of an introductory linguistic class and could help students with understanding spatial semantics in non-western languages and with analyzing foreign languages in general.[2]

For this thesis, I extended this game with a learning model that implements the algorithms described in chapter 4. Most of the source code had to be rewritten, although the graphical design (which was largely Marianne's work) remains almost entirely unchanged. The platform used for creating the software is Unity3D (see `www.unity3d.com`), which is a professional-grade toolkit for developing games and other visual applications but is free for noncommercial use. Unity3D makes it relatively easy to create virtual 3D worlds using a graphical interface and to then manipulate these using scripts. An additional advantage is that software created using Unity3D can be run on a variety of platforms (e.g. on desktop PCs or on mobile devices such as phones or tablets) and can even run inside a web browser (a working version of the software of this thesis is available at `https://gossminn.github.io/AlienLearningAgents/AlienLearningAgentsBuilds2/`).

5.1.2 User interface

Figure 5.1 shows the user interface of the simulation. The interface has two important components: first, most of the screen is taken up by a spatial world with a river (B) and two entities (C and D); second, at the top of the screen there is a control panel (A). The control panel is shown in more detail in figure 5.2 and has two main functions. First, using two buttons and input fields, the user can pause and resume the simulation and specify for how long it should run and how fast it should run. Second, the control panel shows the current state of the model: there are two text boxes displaying the last sentence uttered by the parent and by the child,

[1] This was for the Honours Class 'Learning Through Virtual Reality', taught by Robin de Lange in the fall of 2016 at Leiden University. The source code of this project can be found at `https://github.com/mdhk/TalkToAliens`.

[2] A short essay explaining this project in more detail can be found at `https://goo.gl/UrGPbg`.

Figure 5.1: The interface of the model. *A: Control panel; B: River; C: Entity 1 (chair); D: Entity 2 (rabbit).*

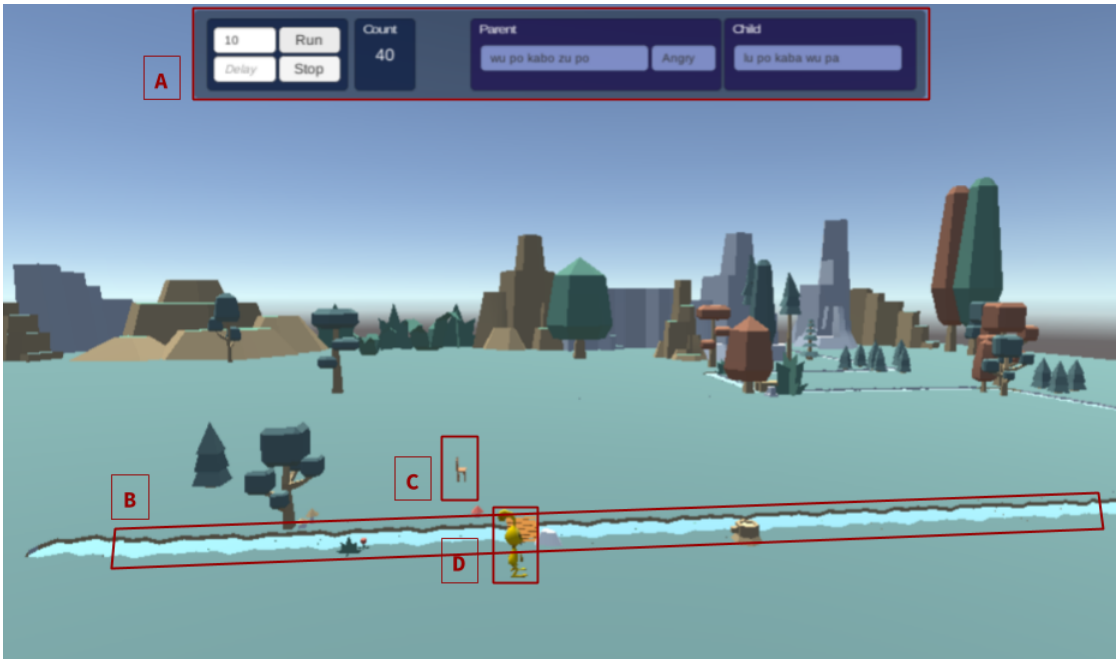
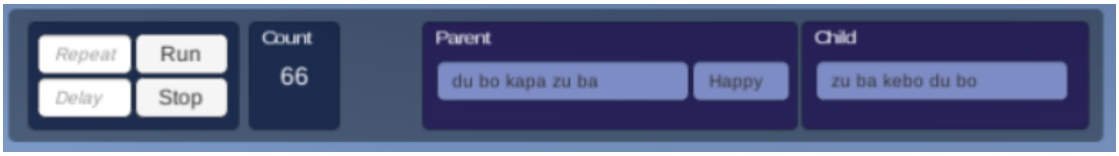


Figure 5.2: Control panel. *From left to right: 'repeat' and 'delay' input fields; 'run' and 'stop' buttons; counter; parent's sentence and feedback; child's sentence.*



respectively, while a third text box shows the parent's feedback when the child has 'said' something. The current 'time' of the simulation (there is a 'tick' every time that a new situation is generated) is also shown.

5.2 The program

5.2.1 Programming language and style

The programming language used for implementing the learning model and for scripting the user interface is C#. C# is a high-level, object-oriented language; it is very similar to Java (and to a lesser extent to C++ and Python) but is somewhat more flexible and has better support for functional programming (see below). It is not very commonly used for scientific computing but is widespread in enterprise development. The main reasons for using it for the purposes of this

thesis are that it is the language that is most widely used for developing Unity3D applications and that it is a modern, general-purpose language that can be used for almost anything and supports a wide range of programming styles.

A large part[3] of the program makes use of a programming style called *functional programming*. Functional programming is a paradigm that defines computations in terms of mathematical functions that describe relationships between pieces of data in the program. For example, one of the most important functions in the learning model is `Learn()`. This function will, given a description of the current state of the child agent, an input sentence from the parent and a logical model corresponding to the current situation, produce a description of the next state of the child agent. By contrast, using a more traditional, imperative style, the same would be achieved by specifying a sequence of instructions for modifying the child agent. Functional programming has several advantages. First, it enables the program to be written in a more declarative way: the focus is more on describing *what* the program should do, rather than *how* it should do it. Furthermore, it reduces the risk of errors: in functional programs, because data is 'immutable' (cannot be modified), computations never change existing values but always produce new ones, which forces the programmer to be more explicit about what the effects of these computations are. Unfortunately, an exhaustive description of how functional programming works and how it differs from other programming styles is beyond the scope of this thesis; see van Eyck and Unger (2010) for more details about functional programming and how it can be applied to formal linguistics.

5.2.2 Structure of the program

C# programs are organized using *namespaces* and *classes*. Classes are blueprints for data structures and their associated behavior. For example, the simulation contains a class called `ChildAgent` for representing the state of the child agent. This class specifies that each instance of this class (i.e., each version of the child agent) contains information about the knowledge set, the most recently uttered sentence, etc., and can be transformed into a new version of the child agent by the `Learn()` function. There are also so-called *static classes*, which only contain behaviour. For example, the algorithms for generalizing syntactic categories are stored in the static class `CategoryGeneralizing`, which does not define any data structures but only contains functions that operate on category sets (which are themselves defined in another class). Meanwhile, namespaces are simply collections of related classes, or of other namespaces[4].

The simulation has two main namespaces. The most important one of these is `LearningEngine`, which contains the 'business logic' of the model (see 5.3 for a schematic overview).

[3] Although functional programming works very well for defining the learning algorithms, it works less well (at least when using the Unity3D platform) for scripting visualizations and for processing user input. For this reason, the simulation makes use of the 'functional core, imperative shell' principle, which holds that business logic (defined in a functional style) should be separated as much as possible from interaction with the outside world (defined in an imperative style); see Bernhardt (2012a) and Bernhardt (2012b). Also note that even in the business logic part of the simulation, some functions are 'impure' because they make use of random numbers or write to log files. This would require more advanced constructs such as monads in purely functional languages such as Haskell, but is unproblematic in mixed functional/imperative languages such as C#.

[4] Namespaces can also contain other type definitions, such as enums or interfaces; although these are used a few types in the program, they will not be discussed here.

Figure 5.3: Organization of the `LearningEngine` namespace

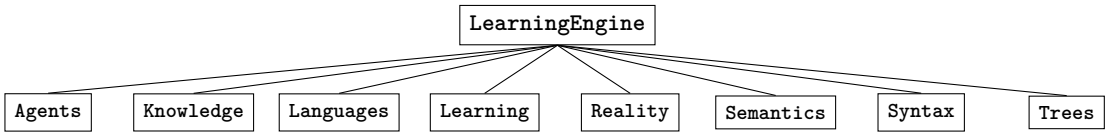
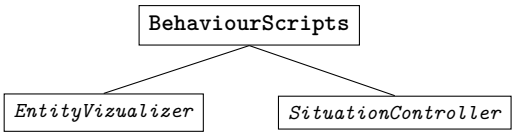


Figure 5.4: Organization of the `BehaviourScripts` namespace (namespace names are printed in **boldface**, class names are printed in *italics*)



`LearningEngine` has eight sub-namespaces: first of all, `Agents` contains classes for representing child agents and parent agents. `Knowledge` contains the blueprints for knowledge representations (category sets, vocabulary sets, etc.; see section 4.2). The `Languages` namespace defines hard-coded instances of these blueprints, which are used for constructing the parent agent, while `Learning` provides functions that the child agent can use for constructing and updating its own versions of these. The other four namespaces are not necessarily related to the learning algorithm itself but take care of the preconditions which allow the learning process to take place: `Reality` contains classes for representing situations and the entities in these situations; `Semantics` is used for representing logical models, semantic types, and contains pre-made basic concepts. Finally, `Syntax` contains blueprints for syntactic categories and rewrite rules, and `Trees` has classes for modeling phrase structure diagrams as well as a simple parser.

The other top-level namespace is `BehaviourScripts`, which handles visualization and the user interface. It consists of two classes, `EntityVisualizer` and `SimulationController`. The task of `EntityVisualizer`: it receives an abstract representation of a situation and then uses this to place the appropriate entities at the appropriate position in the virtual world. Meanwhile, `SimulationController` represents the global state of the simulation and responds to user input.[5] It stores the most recent version of the child agent, the parent agent, and the current situation. When the 'run' button is clicked, the following steps are executed (and repeated for a specified number of times): first, a new situation is generated at random, and this visualization is passed to the `EntityVisualizer`. Then, the parent agent produces a new utterance based on the situation; this utterance is then shown in the control panel and used to update the child agent. Finally, if the child has also produced an utterance, this sentence will also be shown in the control panel and then passed back to the parent agent. Finally, the parent produces feedback which is then used to update the child agent again.

[5] `SimulationController` is the 'imperative shell' in the 'functional core, imperative shell' model (Bernhardt 2012a; Bernhardt 2012b): it is itself mutable and interacts with the graphical user interface, but the representations of the child agent, the parent agent, and the situation (which together form the 'functional core') that it stores are immutable and are updated using pure (or almost pure) mathematical functions.

6 Conclusion

6.1 Review of the model: problems and successes

The learning simulation developed in this thesis tries to bring together computational semantics and Artificial Grammar Learning (AGL) in two ways: first, it aims to bring formal and computational semantics to AGL by introducing a new type of experiment that makes use of languages with a formally defined semantics, and by showing how an Intelligent Agent (IA) could learn it. Second, it wants to bring AGL to computational semantics by showing how meaning representations can be automatically inferred from input rather than being manually defined by a linguist. At first glance, it seems to be quite successful in both of these areas: the child agent in the simulation is able to approximate its parent's linguistic competence in a relatively short amount of time (satisfying the first aim) and makes use of formal representations that have not been hard-coded but are created dynamically (satisfying the second aim).

However, there are at least two major potential problems. The most obvious one is that the class of artificial languages used is extremely simplistic: these languages have a grammar that not only lacks many of the features (such as recursion) that AGL researchers are typically most interested in, but is also extremely rigid and completely devoid of syntactic or semantic variation. In part, these kinds of limitations are an unavoidable and perhaps even desirable feature of any AGL study: by avoiding the complexities of natural languages, researchers are able to develop and test very specific hypotheses about language acquisition. However, they also allow the learning model developed here to get away with some very unrealistic hypotheses: for example, without the assumption that syntax is completely rigid, the phrase structure acquisition algorithm it uses would not work at all.

The second important problem is that the model makes a number of quite heavy assumptions about the initial state of the learning process: it is not only assumed that the child already possesses a number of basic concepts that it can use for representing word meanings and that these concepts are linked to a particular type system (see section 3.2.4 for arguments in favor of these assumptions), but also that it knows how to structure the knowledge about the target language that it is going to acquire (for example, that it should construct phrase structure rules). The main problem with these types of assumptions is that it is often very hard to test them empirically: for example, we simply do not know how humans' internal representations of language are structured, so it is impossible to determine how realistic the assumptions made by the model are.

Although these are serious issues (and there are undoubtedly many more), they are primarily of an empirical, rather than a methodological nature. As stated in section 1.2, it was never the intention of this thesis to make any empirical claims but rather to produce a 'proof of concept' of how AGL experiments using languages with a formally defined semantics could work and how IAs can acquire such languages. Thus, even if some of the specific hypotheses that were

made about the learning process turn out to be untenable, the overall approach can still be successful.

6.2 Suggestions for future research

Arguably the most interesting follow-up study would be to conduct an actual AGL experiment that tests whether or not humans can learn the type of language that is used in the simulation developed here. Various experimental setups are possible, but one possibility would be to emulate the circumstances of the learning process in the simulation as much as possible: the participant would have the role of the child agent, while a computer (or possibly a second participant) would serve as the parent agent. The participant could then be asked to listen to (or read) the parent's utterances and to produce an utterance of their own once they are ready, and their performance could be compared to that of the child agent in the simulation. There are several points of comparison that could be especially revealing: for example, what kind of mistakes would participants make? How soon do they start producing their own sentences? And, perhaps most interestingly, what would their learning curve look like? As was shown in figure 3.1, the child agent's performance initially fluctuates wildly and then tends to converge towards 100% (while experiencing many minor and sometimes also major 'hiccups' along the way). Whether or not this applies to humans as well might say something about whether or not the knowledge they acquire is definitive or whether they sometimes need to revise the hypotheses they have made.

More traditional AGL experiments with this type of language are also possible: for example, participants could first be exposed to a subset of the sentences of the language (possibly along with pictures showing the situations that these sentences describe), and then be presented with another set of sentences and asked to judge whether these are grammatical and/or semantically accurate. In this way, it would be harder to compare the participants' performance to that of the child agent in the simulation. However, there are other interesting possibilities: for example, it would be interesting to test whether or not giving participants any semantic information (in the form of pictures or otherwise) affects the learning process.

It would also be interesting to extend the type of artificial languages developed in this thesis to include more complex semantic phenomena. A very simple way to do this would be to introduce negation, for example by including a morpheme in every sentence that indicates whether or not the rest of the sentence is true. It would be interesting to see how this type of modification to the language would affect the learning curve (and whether or not the child agent could learn it at all).

Bibliography

Alhama, R.G. and W. Zuidema (2016). "Generalization in Artificial Grammar Learning: Modelling the Prospensity to Generalize". In: *Proceedings of the 7th Workshop on Cognitive Aspects of Computational Language Learning*. Stroudsburg: Association for Computational Linguistics.

Beckers, G.J.L., J.J. Bolhuis, K. Okanoya, and R.C. Berwick (2012). "Birdsong Neurolinguistics: Songbird Context-Free Grammar Claim is Premature". In: *NeuroReport* 23 (3), pp. 139–145.

Beram, M.J. and L.A. Heimbauer (2015). "A Longitudinal Assessment of Vocabulary Retention in Symbol-Competent Chimpanzees (Pan troglodytes)". In: *PLOS One* 10 (2).

Bernhardt, G. (2012a). *Boundaries*. Presentation at RubyConf 2012 (Denver, Colorado), available at https://www.youtube.com/watch?v=yTkzNHF6rMs.

Bernhardt, G. (2012b). *Functional Core, Imperative Shell*. Screencast, available at https://www.destroyallsoftware.com/screencasts/catalog/functional-core-imperative-shell.

Blackburn, P. and J. Bos (2003). "Computational Semantics". In: *Theoria* 18 (1), pp. 27–45.

Chierchia, G. and S. McConnell-Ginet (2000). *Meaning and grammar: An introduction to semantics*. Cambridge, Massachusetts: MIT Press.

Cremers, C. and H. Reckman (2008). "Exploiting Logical Forms". In: *Proceedings of the 18th Meeting of Computational Linguistics in the Netherlands*. Ed. by S. Verberne, H. van Halteren, and P. Coppen. Utrecht: Landelijke Onderzoeksschool Taalwetenschap, pp. 5–20.

Fitch, W.T. and A. Friederici (2012). "Artificial Grammar Learning Meets Formal Language Theory: An Overview". In: *Philosophical Transactions of the Royal Society B* 362, pp. 1933–1955.

Fitch, W.T. and M.D. Hauser (2004). "Computational Constraints on Syntactic Processing in a Nonhuman Primate". In: *Science* 303 (5656), pp. 377–380.

Geambaşu, A., A. Ravignani, and C.C. Levelt (2016). "Preliminary Experiments on Human Sensitivity to Rhythmic Structure in a Grammar With Recursive Self-Similarity". In: *Frontiers in Neuroscience* 10 (281).

Gentner, T.Q., K.M. Fenn, D Margoliash, and H.C. Nusbaum (2006). "Recursive Syntactic Pattern Learning by Songbirds". In: *Nature* 440 (7088), pp. 1204–1207.

Hochmann, J., M. Azadpour, and J. Mehler (2008). "Do Humans Really Learn A^nB^n Artificial Grammars From Exemplars?" In: *Cognitive Science* 32 (6), pp. 1021–1036.

Hurford, J. (2014). *The Origins of Language: A Slim Guide*. Oxford: Oxford University Press.

Litamahuputty, B. (2012). "Ternate Malay: Grammar and Texts". PhD thesis. Leiden University.

Marcus, G.F., S. Vijayan, S. Bandi Rao, and P.M. Vishton (1999). "Rule Learning by Seven-Month-Old Infants". In: *Science* 283 (5398), pp. 77–80.

Morgan, J.L and E.L. Newport (1981). "The Role of Constituent Structure in the Induction of an Artificial Language". In: *Journal of Verbal Learning and Verbal Behavior* 20, pp. 67–85.

Partee, B. (1973). "Some Transformational Extensions of Montague Grammar". In: *Journal of Philosophical Logic* 2 (4), pp. 509–534.

Pinker, S. (1996). *Language Learnability and Language Development*. Cambridge, Massachusetts: Harvard University Press.

Prins, M. (2011). "A Web of Relations: A Grammar of rGyalrong Jiǎmùzú (Kyom-kyo) Dialects". PhD thesis. Leiden University.

Prusinkiewicz, Przemyslaw and Aristid Lindenmayer (2004). *The Algorithmic Beauty of Plants*. New York: Springer-Verlag.

Russell, S. and P. Norvig (2003). *Artificial Intelligence: A Modern Approach*. Second Edition. Upper Saddle River: Prentice Hall.

Saffran, J.R., R.N. Aslin, and E.L. Newport (1996). "Statistical Learning by 8-Month-Old Infants". In: *Science* 274 (1926), pp. 1926–1928.

Shirley, E.J. (2014). "Representing and Remembering Lindenmayer-Grammars". PhD thesis. University of Reading.

Spierings, M.J. and C. ten Cate (2014). "Zebra Finches are Sensitive to Prosodic Features of Human Speech". In: *Proceedings of the Royal Society B* 281.

Spranger, M. and L. Steels (2012). "Emergent Functional Grammar for Space". In: *Experiments in Cultural Language Evolution*. Ed. by L. Steels. Amsterdam/Philadelphia: John Benjamins Publishing Company, pp. 207–232.

Van Eyck, J. and C. Unger (2010). *Computational Semantics with Functional Programming*. Cambridge: Cambridge University Press.

Zimmerer, V.C., P.E. Cowell, and R.A. Varley (2011). "Individual Behavior in Learning of an Artificial Grammar". In: *Memory & Cognition* 39 (3), pp. 491–501.