# The Unified Plausibility Parser: how story-understanding systems can read stories written for humans

by

Emanuele Ceccarelli

B.S., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Department of Electrical Engineering and Computer Science
July 28, 2017

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Patrick H. Winston
Ford Professor of Artificial Intelligence and Computer Science
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Christopher J. Terman
Chairman, Masters of Engineering Thesis Committee

# The Unified Plausibility Parser: how story-understanding systems can read stories written for humans

by

Emanuele Ceccarelli

Submitted to the Department of Electrical Engineering and Computer Science
on July 28, 2017, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

In this thesis, I describe how the difficult task of understanding ungrammatical and complex sentences may be tackled by leveraging the power of expectation. Genesis, a state of the art story understanding system, currently struggles to read any sentences that have not been written specifically with it in mind, even when these sentences are understandable by humans with little to no ambiguity. If we want to develop a computational account of human intelligence via Genesis, then it is of fundamental importance that Genesis can interact directly with text meant to be read by humans. For this purpose, I designed and implemented in Java the Unified Plausibility Parser (UPP), a tool whose goal is to interpret complex and ungrammatical sentences and translate them to a language that Genesis can understand. UPP has access to Genesis's knowledge base, and it uses it to develop expectations about what sentence structures should be. Then, UPP can tackle ungrammatical and complex sentences by generating plausible interpretations to them and selecting the one that best fits the expectations previously developed. UPP develops and uses expectation with the method of lattice learning. In this this thesis, I will also describe a new lattice learning framework, which can represent the plausibility of complex sentences using only positive examples. Using a knowledge base of just a handful of examples, UPP can successfully parse and disambiguate the meaning of sentences that were previously impossible to understand by Genesis or any of its subsystems.

Thesis Supervisor: Patrick H. Winston
Title: Ford Professor of Artificial Intelligence and Computer Science

# Acknowledgments

I am truly grateful to Patrick Winston for his guidance on this thesis project and much beyond that. He let me work with freedom and allowed me to follow what I envisioned, but at the same time he provided me with valuable advice. Besides helping me think, he taught me how to speak, how to read and how to write, and he made me discover a new passion for expressing my own ideas.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

After reading this introduction, you will understand why it is important for Genesis to understand sentences written for humans, and what are some of the barriers that stop it from achieving this goal. If we want to develop a computational account of human intelligence via Genesis, then the issue of understanding these sentences is crucial: without this ability, Genesis cannot interface itself with real human writing.

As you read, you will become familiar with my argument: that is, that expectation is the fundamental ingredient that allows humans to interpret the sentences that Genesis struggles with. This is the main principle that guided me in the implementation of the Unified Plausibility Parser (UPP), a system whose purpose is to parse and disambiguate such sentences.

In this thesis, I walk you through the salient details of my implementation of UPP, showing how these complex sentences can be understood. In this introduction I limit myself to painting in wide brushstrokes a picture of UPP. After reading it, you will have gained gain a broad understanding of how the system works and why it has been designed this way.

## 1.1 The problem: understanding human writing is important but difficult

Written or oral, language is universally recognized for its importance in allowing people to interact, share knowledge and stories. However, humans tend to sometimes overlook the processes via which they understand the words that they read or they listen to. As we interact with the world, we may take these processes for granted; however, as we design artificial systems with the same goal, the importance and difficulty of these processes becomes apparent.

Genesis interacts with the world by the means of story understanding: this interaction can only be severely limited if the stories it can read are not something that humans write for humans. Currently, Genesis can only read sentences (or stories) that obey a strict structure. Allowing Genesis to read stories meant for humans would be a leap comparable to the one between a robot interfaced with a simulation and one interfaced with reality itself.

Genesis reads sentences in English using Boris Katz's START[4] program. Given a sentence in English, START attempts to translate it to innerese, Genesis's inner language. START has a number of valuable properties: it is somewhat flexible, it requires no training, and when successful it is generally very fast. However, START can only go that far in understanding human sentences, and the reason for this is intrinsic in the program: START bases its interpretations purely on a strict grammatical and syntactic structure to which sentences are expected to adhere. In absence of this structure, START stumbles in the ambiguities and the variations of human language, which are too many to be directly encoded by any programmer.

Where START stumbles, in my opinions, lies the key to the vault of human language interpretation. If we are to build a system that understands stories like a human, we must step beyond grammar and structure and examine the other aspects of sentences that humans use when they read.

## 1.2 Understanding ungrammatical sentences is a key human capability

In the previous section, I identified the necessity of a grammatical and syntactic sentence structure as the main weakness of START; in other words, as the one barrier that would prevent it (and by extension, Genesis) from leaping to the next level of understanding. There is no better way to break this barrier, in my opinion, than to exercise imagination and think about the opposite of cases: the one of understandable sentences whose structure is plainly incorrect. Consider, for example, the following sentence:

*The pigeon the wolf eats.*

To the human reader, it is obvious that the sentence is ungrammatical, but it is normally intuitive that the intention of the writer was to describe a wolf eating a pigeon. On the other hand, the START parser cannot say anything about the meaning of this sentence. This difference of capabilities illustrates very well what in my opinion is START's largest shortcoming. At a first glance, this pigeon-wolf example is not very realistic. While understandable, it is not what people would say if they were to describe a wolf eating a pigeon. That being said, I believe that being able to parse it would be an impressive and important achievement for Genesis. In the first place, this pigeon-wolf sentence is humanly understandable and increasing the size of the set of humanly readable things that Genesis can parse is valuable in and of itself. More importantly, however, understanding this pigeon-wolf sentence would mark the rupture of the structural barrier that currently prevents Genesis to scaling up towards real stories. One additional interesting remark is that sentences whose structure is scrambled are particularly common in poetry. Genesis reading poetry was never the goal of my research, but it is an interesting application that naturally arises from UPP and that could be further explored in the future.

Understanding ungrammatical sentences is the main focus that drove my implementation of UPP, but I must make one thing clear before moving forward: while structure and grammar are not everything, they are still a fundamental part of what makes sentences readable. Any system that forgets about grammar entirely would be far less biologically plausible than START, and would also be bound to heavily struggle with ambiguities. Grammar and meaning are both important and they factor in our reasoning in complementary ways: that is why I argue that any parsing program will need to leverage both.

## 1.3 An approach: sentence interpretation guided by expectation

The pigeon-wolf sentence of the previous section is, as I already argued, easy for humans to understand. Compared with the grammatically correct version of the same sentence, however, our pigeon-wolf takes a couple of moments to read through. My goal when implementing UPP was to imagine a biologically plausible reasoning that a human could perform in that couple of moments, and then reproducing the same reasoning in the form of a computer program.

### 1.3.1 How do we, as humans, attempt sentence interpretation?

When glancing over the pigeon-wolf sentence, a few features immediately stand out. Eating is a commonly used verb, and it normally expects a subject and an object. The pigeon and the wolf are possible nouns: either of them could be the subject (or object) of the action of eating. Who eats, however, and who is eaten? To a human, this is quite obvious: a wolf can easily be thought to eat a pigeon, while the opposite interpretation is very unrealistic. This step is completely absent in START: to that parser, as long as the sentence structure is correct, our pigeon could happily eat a train. I argue that this step is the key to understanding complex sentences, and that there are two important human abilities that go into it:

- Humans can form semantic expectations, and these expectations play an active role when humans are trying to evaluate whether a scenario is plausible. Anyone who saw a wolf eat a pigeon would build an expectation about that being a realistic scenario, and against the opposite happening.

- Humans have strong generalization abilities. Most of us haven't seen or heard of wolves eating pigeons, but we have seen or heard of other mammals attacking other birds, and that is sufficient for us to draw our conclusions.

### 1.3.2 Translating the human method into UPP

This concludes the main idea that I want to convey through this thesis: proper understanding of human language heavily uses expectation. This idea is the cornerstone of UPP, which works in two steps.

Firstly, UPP must be able to generate interpretations for a sentence. This generation must be guided by expectation: not all possible arrangement of words in a sentence are born equal in our minds. Secondly, we again use expectation to evaluate the plausibility of these interpretations.

Underlying these two steps there must be another, less apparent part: the one in which examples and rules are processed to build these expectations. As I explain in the later parts of the thesis, learning from examples is very similar to evaluating plausibility, so the programs that perform these two functions largely overlap.

These steps were the motivation in the implementation of UPP, which is divided in two modules that correspond to them. The capabilities of the two modules are described in the next two sections of the introduction, and the details of their implementation will come in the next two chapters of this thesis.

In order to implement UPP, I started by reviewing some of the existing methods[3][7] for sentence parsing and disambiguation that existed within and outside of Genesis. I was not the first one to tackle the issue of understanding complex and ungrammatical sentences, but I believe that the existing methods had fundamental shortcomings that prevented them from being the answer to the problem.

After familiarizing myself with these methods, I designed my own: the decision to call it Unified Plausibility Parser comes from the fact that it always aimed to unite grammar and meaning into a single measure of plausibility. The natural step to follow was the implementation of the two modules of UPP. As I will explain in more depth in the rest of this thesis, the interpretation-generating module rests some of its capabilities on the plausibility-evaluating one, so I implemented this second module first.

Eventually, I constructed some examples that I could use to demonstrate the capabilities of UPP. UPP was successful in understanding these examples, which lead to the final goal of incorporating UPP as a subsystem of Genesis.

## 1.4 UPP evaluates sentence plausibility

UPP is a two-sided system. On one side, UPP generates plausible sentence interpretations and on the other side it evaluates these interpretations. These two goals are closely linked: both of them rely on some measure of plausibility for sentence interpretations. There are two broad notions for this plausibility, one being related to the meaning being acceptable and one being related to the grammatical structure being correct.

The first of these notions is the one that we saw in play when we were trying to understand the meaning of our pigeon-wolf sentence. Grammatically "The pigeon eats the wolf" is just as correct as "The wolf eats the pigeon", and it is this notion of plausibility of meanings that allows us to resolve the ambiguity. As we look closer at this resolution, we realize that it is more complicated than it seems. Pigeons and wolves are both capable of eating and of being eaten; when we put them together, however, one of the sentence interpretations seems wildly implausible.

The second notion is the one that helped us place the correct parts of sentence in their place. It is the one that prompted us to look for a verb, so that we would have something that the sentence could revolve around. Again, this is tougher than it seems—to humans, the action of eating evokes the necessity of an object as well as a subject; other actions will evoke different expectations.

In chapter 2 of this thesis I will resolve the questions that I just opened by diving much more deeply in the method of lattice learning and in how I took this method to the next level. It will argue why this method fits the bill that UPP needs, and show the power of the measure of plausibility that emerges.

## 1.5 UPP generates sentence interpretations

Once UPP had a good system in place for the evaluation of plausibility, the next step was to leverage this information for the purpose of generating sentence interpretations. This step is perhaps the toughest to understand within the framework of human reasoning.

Going back once more to the pigeon-wolf sentence, we realize that we are making an explicit choice between the pigeon and the wolf as the subject. However, we are never even considering a sentence interpretation where eating and wolfing are both verbs (where wolfing is a rarely used verb used to describe particularly voracious eating). Clearly, any interpretation of that kind would not pass the test of plausibility, but it is even more interesting to me that this interpretation is apparently pruned before being brought to the test itself.

This example shows that expectation plays a large role in the process of generating interpretation. I would also argue that using expectation in this way is exactly what makes humans so fast at tasks like this one—much faster than START or UPP or any other artificial intelligence system I have ever encountered. The key challenge of interpretation generation, then, will be to properly incorporate expectations into the system. To illustrate this, in the later phases of development of UPP, I enhanced the candidate interpretation generator in a way that would use expectation more heavily. After this improvement, longer sentences were understood in a couple of minutes, while before they would often take ten times as long or be too computationally intensive for the testing computer.

In chapter 3 of this thesis I will further clarify this point, and describe my implementation of the interpretation generation system.

## 1.6 The context: UPP is part of Genesis, a story-understanding system

As I mentioned at the beginning of this introduction, Genesis interacts with the world by understanding stories and UPP aims to broaden the set of stories that Genesis can interact with. In this introduction, I explained my vision for UPP, but this vision is difficult to understand without some background about Genesis itself.

Genesis is a story-understanding system whose goal is to develop a comprehensive computational account of human intelligence[10]. It was and keeps being developed by Patrick Winston and many students who were inspired by thinking about how story understanding can be crucial in achieving this ambitious goal. Not only story understanding is something that differentiates humans from all other sentient animals, it is also something that often strikes as most intuitively and naturally human. This focus on story understanding is a trait that makes Genesis a truly unique artificial intelligence system.

In order to understand stories, Genesis must first develop a representation of the entities that take part of them, and the events that occur. In Genesis, it is important to ground these representation in symbolic thinking, another key human capability. Once a representation is established, which is something I discuss in more detail in Chapter 2, Genesis can use rules to tie these events and entities together meaningfully.

Genesis can read stories that range from a 100-sentence version of Shakespeare's Macbeth to a description of the Estonia-Russia cyber-war, and perform a number of human-like tasks on them. Genesis can detect important themes and concepts, such as revenge; it can answer questions and provide summaries; it can model character traits as well as the reader's own biases.

Genesis can accomplish all these tasks on stories that are written in sentences that are simple enough for START to parse, as I already explained earlier in this introduction. I believe that allowing Genesis to scale up towards more complex sentences would be a truly important achievement, because it would allow these tasks to be performed on a broader set of stories. In the rest of this thesis, I describe how expectation and plausibility provide the key to this important leap.

# Chapter 2

# UPP measures plausibility with lattices

In the introduction of this thesis, I described how UPP approaches the problem of interpreting ungrammatical and complex sentences by generating sentence interpretations and evaluating the plausibility of these interpretations. The main guiding principle of my thesis is that at the core of this idea of plausibility lies human expectation—in other words, that humans judge the plausibility of a scenario by comparing it with what they have already seen.

How to build and use expectation to evaluate sentence plausibility is the central topic of this chapter. I start by describing previously developed systems within Genesis whose goals and methods are relevant to UPP. Then, I describe the method of lattice learning, which I extensively used in UPP. More importantly, I describe how I analyzed lattice learning, identified its shortcomings and eventually took it to the next level by introducing the concept of Frame Nested Learning. I also comment on the complexity of the plausibility-evaluating side of UPP.

After reading this chapter, you will be significantly more familiar with the design on the plausibility-evaluating side of UPP. Any questions you might have from the introduction will probably be answered. You will be able to explain how this module of UPP works to any kind of audience, regardless of their technical expertise.

## 2.1 Plausibility evaluation in Genesis

The problem of evaluating sentence plausibility in Genesis is a broad one, and it touches on a variety of topics. One, which was a focus of my thesis, is concerned with the parsing of ungrammatical sentences. However, other students in the past have developed relevant systems within Genesis: Eli Stickgold's DISAMBIGUATOR[7] (developed in 2011) and Josh Haimson's PAP[3] (developed in 2016).

### 2.1.1 DISAMBIGUATOR finds the correct meaning for words

As the name suggests, DISAMBIGUATOR concerned itself with the disambiguation of terms in sentences. For example, given a sentence such as "The hawk flies", DISAMBIGUATOR would be able to tell that the hawk mentioned is a bird and not a politician with militarist tendencies. Word sense disambiguation is quite important to Genesis and also has relevance when it comes to the problem of parsing sentences whose ambiguity is enhanced by the presence of a nonstandard structure. However, DISAMBIGUATOR works only on sentences that Genesis has been able to parse already, so it was difficult for me to take advantage of DISAMBIGUATOR in any meaningful way. As I will explain in more detail later in this thesis, I believe that the development of UPP resulted in the creation of a disambiguating system as a byproduct. This is something that I did not have in mind originally but I believe that it can be more successful than DISAMBIGUATOR in word sense disambiguation.

### 2.1.2 PAP takes an important step towards ungrammatical sentence interpretation

PAP, which stands for Perceptual Alignment Parser, is a more relevant point of reference for my work. Aiming lattice learning directly at ungrammatical sentences, PAP is close to UPP both in purpose and in method. PAP, however, had two significant drawbacks that prevented it from being the system that would allow Genesis to scale up. First and foremost, PAP was designed to deal exclusively with subject-verb-object sentences, which greatly limited its range of application. Secondly, to PAP, all sentences were seen as bags of words—the original structure had no bearing on the sentence interpretation. Nevertheless, I found PAP to be immensely useful as a starting point for my thoughts on ungrammatical sentence interpretation.

The most important contribution that DISAMBIGUATOR and PAP had towards my work lies in the method of lattice learning. In the next two sections of this chapter, I carefully describe this method, and I explain how I took this method to the next level.

## 2.2 WordNet threads can represent unique meanings

In order to understand lattice learning, it is first important to understand threads. Threads in the context of word meaning were introduced by Vaina and Greenblatt in their Thread Memory paper[8], essentially as conceptual links between semantic nodes. In order to more clearly explains what this means, let us first consider an example of a thread:

thing living-thing animal bird hawk

Each of the words in the thread has some meaning. However, we can imagine some of these words having multiple meanings as well. Consider, for example, the word "hawk". As I had already pointed out in the previous section, a hawk can be both a bird and a militarist-leaning politician. The use of threads eliminates such ambiguity. Consider the words in this hawk thread: between each word and the following, we may imagine having an inverse "is-a" relationship. The thread, then, represent a hawk which is a bird which is an animal, and so on. It is always perfectly clear which of the meanings of the word hawk this thread refers to. This meaning uniqueness property is quite powerful, and it should come to no surprise that it was central to the DISAMBIGUATOR system.

### 2.2.1 The Thread System and Genesis entity classes

Beyond DISAMBIGUATOR, threads are used all throughout Genesis. As a story understanding system, Genesis needs instruments to represent the contents of its stories. It does so by using entities, and entities largely depend on Threads. There are four different classes of entities in Genesis:

- Entities themselves are defined by a primed thread and a thread bundle. Entities are normally things that appear in the story: they can refer to physical items like hawks and people as well as abstract concepts like happiness and greed. These entities have a unique meaning that is defined by their primed thread, as well as alternative meanings defined by the rest of the bundle: this is how threads have an important role in Genesis. For the sake of clarity, I will from now on refer to this kind of entity as Entity, while the general entity term will not be capitalized.

- Functions are like Entities but with a slot for a subject, which is an entity itself. Functions tend to be intransitive verbs, such as "to appear": the sentence "a bird appears" will be a Function whose primed thread corresponds to the action of appearing and where the bird is the subject.

- Relations are similar to Functions, but have a slot for an object as well as one for a subject. Our interpretation of the pigeon-wolf sentence of the first chapter is a good example of a relation; in that case, the primed thread of the Relation corresponds to the action of eating.

- Sequences are entities with an arbitrary number of slots for elements, which can be filled with other entities. An example of a Sequence might be a Roles Sequence, where each of the elements has a role in the sentence. In the sentence "The wolf eats the pigeon with his teeth in the forest," the object of the eating Relation is one such Roles Sequence that contains elements that tell us about the object eaten, the location where eating occurs, and the tool used for eating.

Pretty much any concept that is commonly seen in English can be expressed in terms of these entity classes. In Genesis, this is how everything is represented: it is the inner language that the system uses to speak to itself.

In Genesis, threads come from WordNet[6], which provides a large tree of hierarchical meanings which can be used to describe entities. The hierarchical nature of these meanings will be fundamental to the next section: but for now, two things are important. In the first place, we must understand how the structure of threads allows us to associate every entity with a unique meaning. Secondly, we must see how different classes of entities allow us to create sentences with clear structures. Putting together threads and entity classes is what allows Genesis to represent concepts in a way that is unambiguous and meaningful.

## 2.3 Basic lattices can represent plausibility

Given a collection of threads, such as the one described in the previous section, we can imagine how they would fit in a forest of trees, which we will call a lattice. The words in the threads would be the nodes of the forest, and the edges between them would be the "is a" relations. This lattice will have one or a few roots, corresponding to the most general categories, like "thing" and "action". For each node in the lattice, walking from the root to the node will produce a thread corresponding to the node. Each entity, then, can be linked to a node: the node whose thread is its primed thread. The branches at the top of this tree draw high level separations, such as between artifacts and living things; at the lower level of the trees the branches separate closely related items, such as different species of birds. Figure 2-1 shows a lattice.

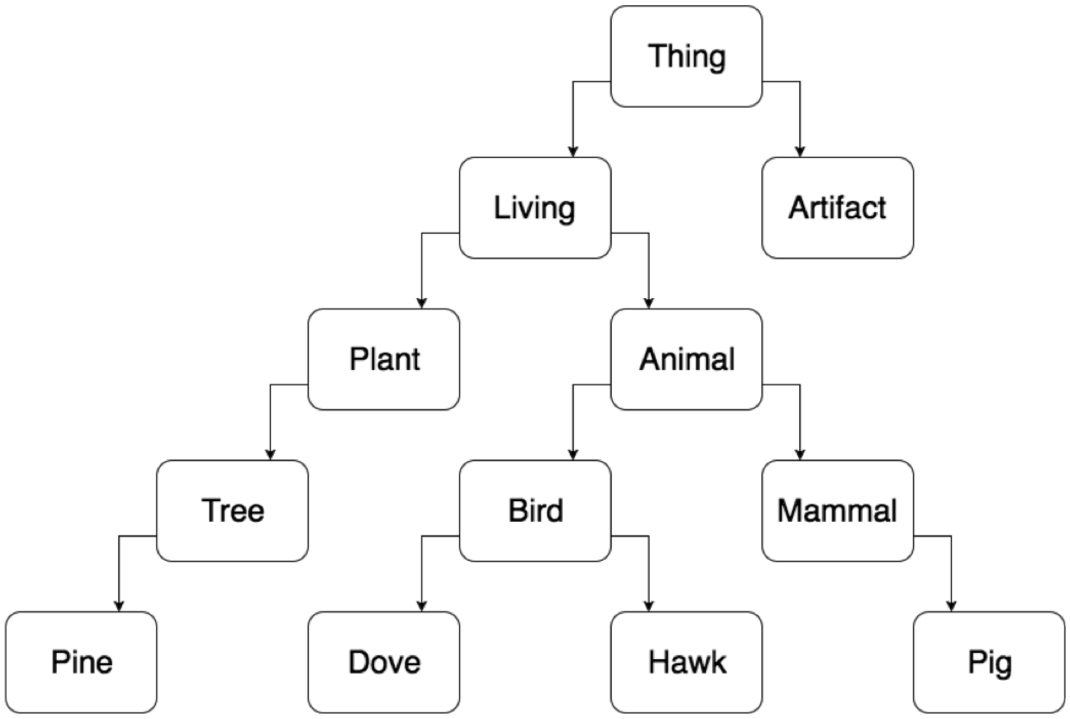

Figure 2-1: A basic lattice.

### 2.3.1 Lattices can be used to represent the ability to do something

Suppose now that the goal is to determine the set of entities that are capable of performing a specific action; for example, all entities that can fly. Remember that all possible meanings (or threads) that an entity could have exist as nodes in the lattice. Then, the ability to fly can be represented by a lattice: because we can mark differently nodes denoting something that can fly and nodes denoting something that cannot fly. This defines a bijection between sets of threads that can fly and marked lattices. This step is a very crucial one and Figure 2-2 should make it very clear.

Lattice learning, as the name suggests, refers to a technique to learn the marked lattice; that is, to mark all nodes according to whether or not the node's thread denotes something that can fly. Basic lattice learning works by reading examples, and categorizing them as positive or negative. A sentence like "The dove flies" is a positive example, and "Pigs cannot fly" is a negative example. Whenever an example is read, it is associated with the lattice node that it belongs to; for example, reading

"Pigs cannot fly" will place a negative example in the "pig" tree node in the "subjects that can fly" lattice.

Then, the marked lattice is processed as follows: **any node is in the "able set" (in the example of doves and pigs, the set of entities that can fly) if and only if it belongs to a subtree with positive examples but no negative examples**.

Besides this definition, I also find it useful to think about lattices as a different way of framing ability questions. If we ask, "Can a hawk fly?", we can rephrase the question as "Is there anything in virtue of which it should be possible that a hawk flies?" This slightly more convoluted question gives us a possibility to answer systematically—in this case, we can say that the hawk is a bird, that all observed birds fly, so the hawk can fly in virtue of the fact that it is a bird. This question-based definition will be very useful in the later sections of this chapter: as we will see, this systematic answer can be further generalized.

Before moving forward, I want to put some emphasis on the concept of negative examples. In this section, I mentioned two examples: "The dove flies" and "Pigs cannot fly." In other discussions of traditional lattice learning, we would often see, instead, "The dove flies" and "The pig does not fly." This is, for example, what we see in PAP, one of the systems I mentioned in the first section of the chapter. I believe, however, that this type of example is not really what we should be looking for. If a writer ever needs to say that some animal doesn't fly, it typically is because that animal could actually fly, it just chooses not to for some reason. For this reason, upon reading that a pig does not fly, I would not be able to infer anything about pigs' inability to fly. In order to infer such inability, I need a stronger statement, such as "Pigs cannot fly." Note that this statement is both an example and a rule. To the lattice, pigs are an example of something that cannot fly. At the same time, however, "Pigs cannot fly" is a rule: it's a universal statement about pigs which can be taught and that can inform all future thought. For this reason, in the rest of this thesis I will call *negative rules* the statements that are traditionally called *negative examples*: I believe that the name is more fitting for what we are really talking about.

Also, we need some care when using the term *rule*, because that term is used with another meaning in Genesis's story understanding system. An example of a Genesis rule could be *If A kills B, then B becomes dead.* These rules are not to be confused with my negative lattice rules. Luckily, there is no room for confusion in this thesis as I never talk about Genesis rules.

Lattice learning is a powerful method: it is sufficient to read that "The dove flies" and that "Pigs cannot fly" to infer that the only flying entities are the ones in the "bird" subtree. Figure 2-2 shows the lattice learning's result. This easy generalizability has a double value: not only it means that training lattices does not require large datasets, it also means that learning lattices has a similarity with human thought, which has strong generalization ability.

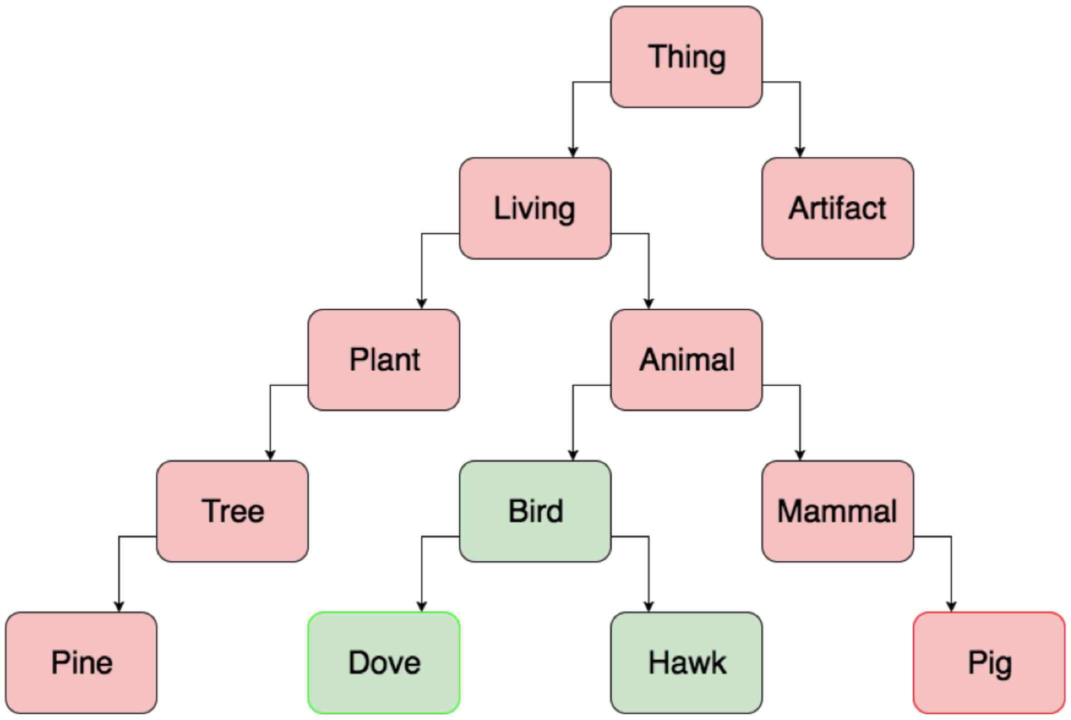

Figure 2-2: The "can fly" lattice after seeing the aforementioned examples. Entities corresponding to green nodes can fly while those corresponding to red nodes cannot. Note that two examples are sufficient to infer marks on other entities.

From this notion of ability it is is easy to extract a notion of plausibility. Suppose that we were trying to evaluate the plausibility of a sentence that says "The hawk flies." The sentence could be deemed plausible if the events it describes are all plausible. In this case, there is only one event, a flying hawk, and by looking at our lattice for the action of flying in Figure 2-2 we can determine that that event is plausible, so the sentence itself is plausible.

### 2.3.2 Lattice learning falls short in three ways

The concepts of lattice learning and thread hierarchies are crucial to the entirety of this thesis: lattices like the ones described in the previous parts of this section are one of the most important structures and they will be mentioned multiple times throughout the chapter. Before moving forward, however, it is important that we familiarize ourselves with the three largest shortcomings of basic lattice learning. To illustrate these shortcomings, let us go back once again to the pigeon-wolf sentence that I presented in the introduction. That sentence might appear trivial to the reader, but it can fluster our lattice learner in a number of ways.

In the first place, basic lattice learning requires negative rules. Looking at the pigeon-wolf scenario makes the process of identifying appropriate negative rules puzzling: pigeons and wolves can eat or be eaten, depending on the context. We could imagine setting negative rules against animals eating animals larger than them, but they would not really fit in the lattice framework; in any case, devising a large number of negative rules to fit all scenarios could easily get out of hand. On a second thought, even the simpler scenario of the not-flying-pig poses some problems: we would need a few negative rules per action in order to develop a complete account of plausibility and these rules would need to be given by the programmer. The reason for this mirrors what happens in real life: rules rarely exist so explicitly, which means that they either need to be taught or can be inferred from positive examples. Teaching all these rules would make scaling difficult; on the other hand, if they could be inferred from positive examples only, no problem would arise. Positive examples are easy to find: any readable sentence that talks about an event is a positive example. If we want lattice learning to be used on a large scale, its reliance on negative rules needs to be eliminated or at least reduced.

The second point is closely tied to the first: in real life, plausibility does not only exist in an all-or-nothing form: more often, it exists on a scale. A pigeon could feed on the remains of a dead wolf—it is just harder to imagine than a wolf eating a pigeon. If we want lattice learning to reflect the subtleties of reality, it must measure plausibility on a scale rather than as an all-or-nothing term.

The last point is perhaps the most obvious when one looks at the pigeon-wolf example: most of the time, it is the context that decides the plausibility of any given action. As I said multiple times, a pigeon could be imagined eating—it is the context (the fact that the object of the eating is a wolf) that makes the plausibility of the interpretation plummet. If we want lattice learning to produce meaningful results on complex sentences, then it must be enhanced to incorporate contexts.

With these three points in mind, we can move on to the next two sections: Section 2.4 will discuss a solution to the first and the second problem, while Section 2.5 will discuss a solution to the third issue I mentioned.

## 2.4 Beyond lattice learning: Positive Example Learning and plausibility on scale

As I argued in the last section, finding positive examples is easier than finding negative rules. That being said, the idea of lattices is not to be discarded just because it relies on these universal negative statements.

### 2.4.1 How can humans make decisions when they have no negative rules?

In order to take lattice learning to a place where negative rules are no longer necessary, I think it is useful to perform this thought experiment. Let us imagine what happens in the mind of a child that sees wild animals for the first time. With no one around him, the child has no access to rules. As he wanders through nature, the child sees a number of birds flying from branch to branch or across the sky. Occasionally, he sees a few bugs hovering in the air. However, he doesn't see any large mammals fly, or really any other animal that isn't a bird or an insect.

Let us now imagine that the child comes across a dog for the first time. He has no positive evidence about dogs' ability to fly, but he also has no rules according to which dogs should not be able to fly. Nevertheless, we can be quite sure that the child will be surprised if he sees the dog suddenly flying—because he never saw anything similar happen.

### 2.4.2 The concentration of positive examples tells us something about plausibility

This example should illustrate the difference between basic lattice learning and my idea of lattice learning. If the child's reasoning followed basic lattice learning, then he would expect the dog to be able to fly—after all, dogs belong to the subtree of animals and the subtree of animals contains positive examples and no negative rules. I argue that, instead, the absence of positive examples in regions of the lattice functions like a negative rule. Let us consider a slightly different idea of lattice learning. Whenever a positive example is seen, that counts as evidence towards the plausibility of something similar happening. In the case of flying, as we draw positive examples out of reality, we will notice over time that all those examples lie in the subtrees rooted at "bird", "insect", "plane", and so on. When the evidence is so concentrated in a set of regions of the lattice, then we gain an expectation about that part of the lattice being the correct one, and all other regions of the lattice being less plausible.

To our child, after seeing a single example of a flying dove, the lattice will look something like what is shown in Figure 2-3. As the picture clearly shows, plausibility here exists on a scale.

I called this type of lattice learning Positive Example Learning. Positive Example Learning solves two of the issues with basic lattice learning: it reduces reliance on negative rules and it represents plausibility on a scale.

The next section of this chapter will discuss the third of the problems with basic lattice learning, that is the interplay of contexts.

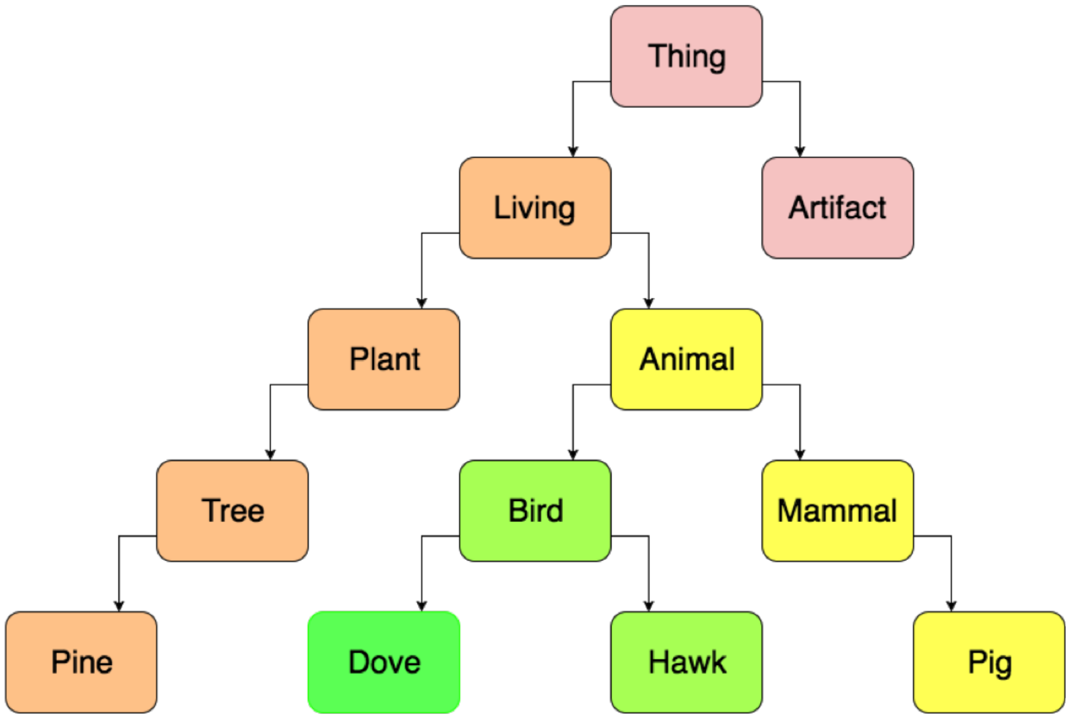

Figure 2-3: The "can fly" positive lattice after seeing the example of a dove. Colors tending to green show higher plausibility. Note that a single example allows for some weak inference already.

## 2.5 Nested lattices can represent the interplay of contexts

As I argued already, contexts have significant importance in establishing a sentence's plausibility. In my description, a context is a very broad term that refers to all the things mentioned about an action that are accessory to the action itself. The ensemble of location, manner, tools used and so on makes up the context in which any action occurs, and each of the elements of a context may be explicit or implicit. Let us consider a few sentences where contexts affect plausibility. First let us consider a basic case:

*The hawk flies.*

This is a simple statement with an action and a subject. The sentence is plausible, as we could certainly see if we run this sentence by a lattice test like the one we saw in Section 2-3 and Section 2-4. There is no explicit context around the action: let's see what happens when we add something.

*The hawk flies in the sky.*

Adding context this way doesn't change the plausibility of the sentence at all. In fact, the change does nothing but make explicit a context that the reader probably already had in mind. Let us consider, on the other hand, this alternative:

*The hawk flies in the sea.*

Now the sentence strikes us as odd. In this case, adding context to the sentence changed its plausibility.

### 2.5.1 Thinking of lattices as tools to reframe questions will help understand the interplay of contexts

As these examples illustrate, the interplay of contexts is crucial when determining the plausibility of a sentence. In order to solve this issue, I introduced the concept of lattice nesting. To explain the reasoning behind lattice nesting, I go back to the question-based definition of lattice plausibility, the one that I had given in Section 2.3.

From that definition, we can evaluate a sentence as plausible if there is anything in virtue of which that sentence should be plausible. Is there anything in virtue of which we should believe that a hawk can fly in the sky? Before answering that question, let us suppose that our evidence consists of seeing a lot of doves and other non-hawk birds flying in the sky, enough to think that birds fly in the sky, but not in any other specific locations and that no other animals can fly. Then, the hawk can fly in the sky by virtue of being a bird, because all birds can fly in the sky.

Let us try to break this argument down further. When we ask the question, "Can the hawk fly in the sky?", we are asking whether there is any plausible statement of the form "X can do Y in context Z" where a hawk is a X, flying is a Y and being in the sky is a Z. In this case, such a statement exists, and it is “A bird can fly in the sky.” Because of that, we can find that our sentence is plausible.

Now, is there anything in virtue of which we should believe that a hawk can fly in the sea? Again, we want to find a statement of the form “X can do Y in context Z” but now being in the sea must be a Z. Because there is no such plausible statement, the sentence is considered implausible.

### 2.5.2 Nesting lattices can represent subsets

It turns out that this type of reasoning can be very well described by nesting lattices. Given a lattice (for example, the one that describes the plausibility of the action of flying), we can add a lattice underneath each of its nodes. Let us look at an example in Figure 2-4.

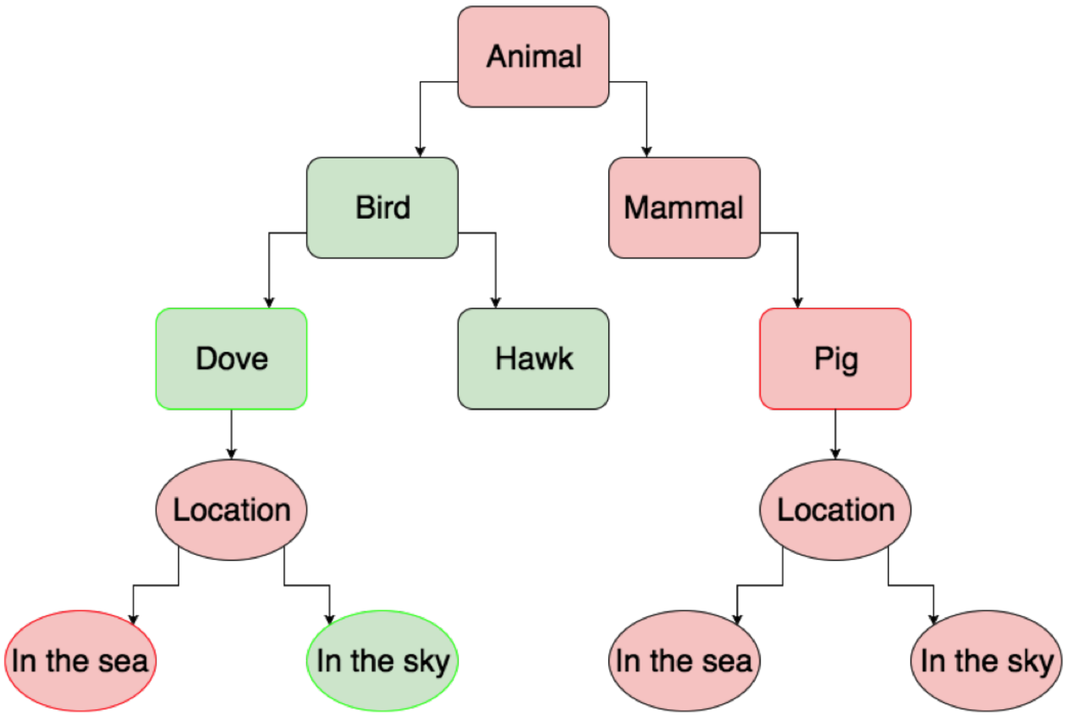

Figure 2-4: A nested lattice. Imagine as if that nested lattice was underneath every node of the top lattice.

In order for an action to be plausible, it must be plausible at all levels. If we want to inquire whether a hawk can fly in the sky, we start by looking at the hawk node.

We have no examples there, so we walk up a node in the top-level lattice. We reach the bird node, and we know that there are examples of birds being able to fly. So, we look into the lattice nested at "bird", and we look for the "in the sky" node. We are convinced that it is plausible for a bird to fly in the sky, so we can state that our sentence is plausible.

Note that the figures shown have the original red-green basic lattice coloring. For explanatory reasons, I find it easier to discuss nestings under all-or-nothing plausibility, but the same results can be obtained as we substitute "plausible" with "plausible enough" and use plausibility on a scale as from Section 2.4.

At the appearance, this type of method can be extended to arbitrarily complex contexts, as long as we can add sufficient nestings. For example, if we want to add context about what our birds are carrying, the lattice in Figure 2-5 shows an example.

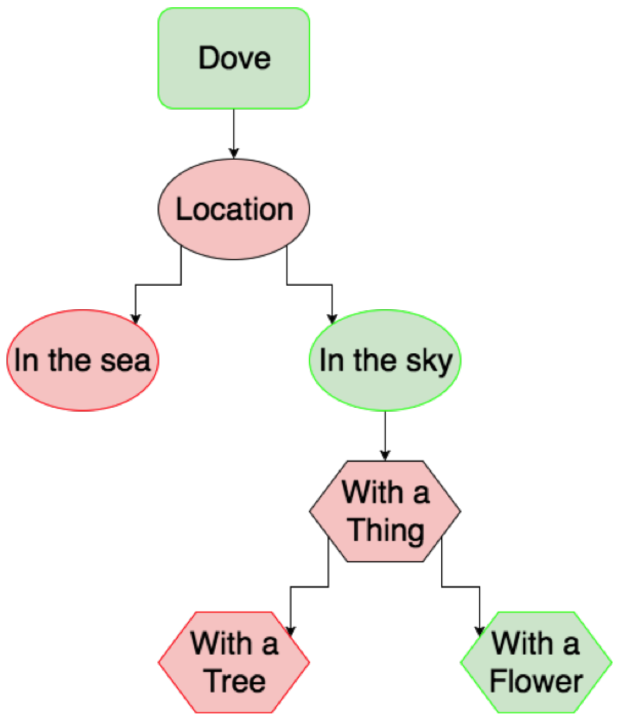

Figure 2-5: A double nesting. The method works as before.

What Figure 2-5 depicts, however, is just a concept: before we dive in its implementation we must recognize a few issues with it. A natural question that arises when looking at Figure 2-5 is what decides the ordering of lattices. Should "The dove flies with a flower in the sky" be represented in the same manner as "The dove flies in the sky with a flower"? While the idea of nesting lattices is appealing, it needs some structure in order to be truly applicable. I will explain how I decided this structure in the next section of this chapter.

### 2.5.3 What happens when the context changes with the story?

Before moving on to the remaining parts of this chapter, I want to examine one last case about context plausibility. As I said, contexts can be both implicit and explicit. Explicit contexts can be dealt with using the method of nesting. Implicit contexts can essentially be forgotten: if there exists any context under which an action is plausible, then we may just assume that that is what the author intended (which among other things could open the path for lattice learning as a method to infer implicit contexts, which would be an interesting problem in its own regard). What if, however, a context is explicit but does not appear in the sentence? To exemplify this, let's look at a new sentence.

*The man inside the painting was moving.*

This sentence seems implausible—people in paintings don't move. This could have been just another example of a time where adding context reduces plausibility. Let us now imagine, however, that this sentence was found in a book from the Harry Potter world. There are multiple mentions of paintings being animated in the Harry Potter world, so our sentence would be totally plausible there. Even if context does not always appear in sentences, it doesn't mean that we can forget about it entirely. Beyond this, throughout stories contexts can evolve. If we are reading *Macbeth*, the plausibility of Duncan talking changes dramatically as we move from the first act (where Duncan is alive) to the last three (where he is dead). UPP does not actively deal with dynamic context changes, but as will be clearer in the next section I designed UPP so that this issue could be dealt with smoothly.

Nested lattices allow us to deal with the interplay of contexts. However, nested lattices can get messy, and with no good rules behind them they will inevitably become too unwieldy to be useful. The next section explains how I tied everything up in a single big idea, Frame Nested Learning.

## 2.6 Recapitulating: Frame Nested Learning and the Context Map

Section 2.3 introduced the idea of lattice learning to evaluate sentence plausibility. Then, Section 2.4 described how this idea can be generalized with Positive Example Learning, achieving plausibility on a scale and reducing reliance on negative rules. Section 2.5 explains how lattice nesting can be used to address the problem of the interplay of contexts. This section will explain how I tied these ideas together into Frame Nested Learning and a Context Map.

### 2.6.1 Genesis role frames allow us to elegantly nest lattices

The main open question from the previous section concerns the structure of nested lattices, which is messy and unwieldy. To deal with that, it is first of all useful to recall the four classes of entities used in Genesis to break sentences into understandable pieces. Those classes are described in more detail in Section 2.2. Let us consider them now from the perspective of nesting.

- Entities can be seen as nodes in any lattice.
- Functions can also be seen as nodes in any lattice, but also allow for a nested lattice, corresponding to the function's subject.
- Relations can be seen as nodes in any lattice, but expect a doubly nested lattice underneath them—the subject nested with the object.
- Sequences can be seen as nodes in any lattice, and can have an arbitrary number of elements appearing underneath them.

The heavy reliance on Genesis role frames is what inspired me to call this system Frame Nested Learning.

Frame Nested Learning resolves the structural problems that we were noting when considering standard nested lattices. Let's think, for example, about the two sentences "The dove flies in the sky with a flower" and "The dove flies with a flower in the sky." With Frame Nested Learning, the two sentences are represented in the exact same way: both "in the sky" and "with a flower" are elements of the roles sequence that is the object of the action of flying. Figure 2-6 shows how this would look in Frame Nested Learning. Thanks to Frame Nested Learning, the structure of sentences in Genesis is perfectly mirrored by the structure of the lattices: everything that can be said in Genesis can also exist, uniquely, in some lattice.

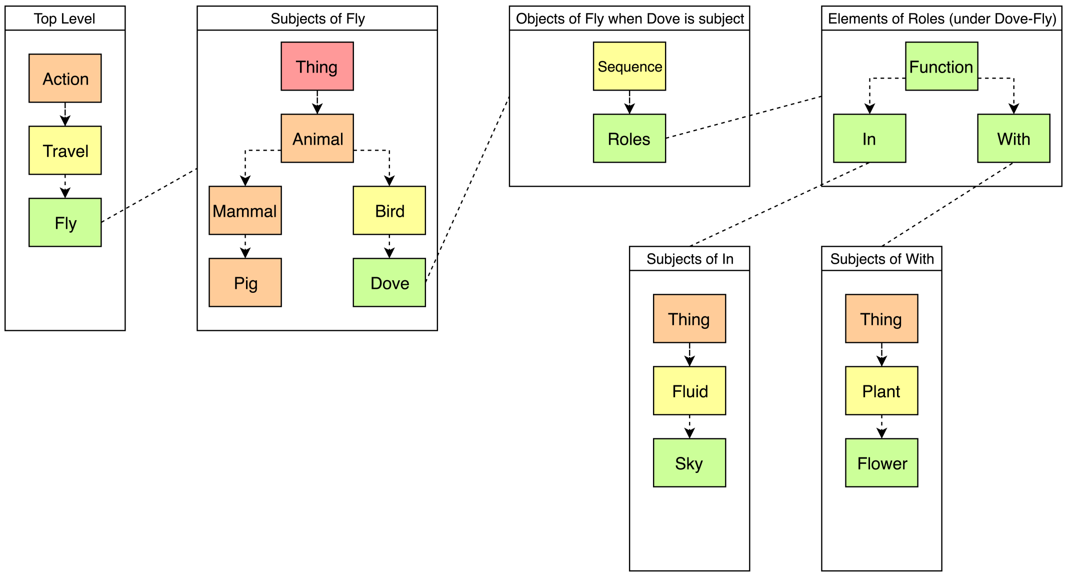

Figure 2-6: The lattice structure shown in more detail, with multiple elements defining a context.

### Overview of implementation

While implementing this system, I strived to keep the structure as uniform and simple as possible, both for my benefit as a programmer and with the goal of making a system which is easy to explain. The following paragraph describes this implementation with more details, which might be interesting to any readers who want to use Frame Nested

Learning for their own purposes.

A Frame Nested Learner (from now on described as FNL) holds a number of Frame Nested Nodes (FNN). These are linked as in a normal lattice learner; however, each FNN also has an underlying FNL. Sentence interpretations in Genesis may be described as entities: whenever an entity is learned by a FNL, it updates all the nodes in the lattice in the path from a lattice root to the FNN whose thread corresponds to the entity's primed thread. By updating a node, I mean two things: in the first place, adding weight to the node in a manner that represents the presence of a positive example, and in the second place, updating the FNL nested at the node by making it learn the entities that are contained in the entity that we were adding. The added weight is going to be at the full value at the last node in the thread, and will decay as we move far from it. Adding an example about a flying dove gives full weight to the "dove" node and to the lattices nested underneath it, but it gives less weight to the "bird" node and even less to the "animal" node. The decay is governed by a parameter given by the user, and further modified so that the cumulative value of the path from the root to the final node is constant.

This method of attributing weights to nodes is somewhat arbitrary and parameter-dependent, but I think it exhibits all the properties of hierarchy and generalization that I have been discussing in this chapter. At the end of Chapter 4 I discuss further how UPP makes decisions that are based on parameters, and sometimes the parameters can get arbitrary and difficult to set. In the majority of cases, UPP performs fairly well regardless of parameter choice, which makes me more confident that this method captures some important properties of meaning.

Genesis represents sentence interpretations in a wonderfully recursive way, which makes this sort of nesting quite natural. It is clear how Entities (which have no slots) and Functions (which have a single slot) fit this description. Relations, which have two slots, add them one nested in the other: the subject is directly nested and the object is nested when the subject's nestings are completed. For sequences, the lattice nested at the FNN actually holds all the elements. Note that this is different from what I represented in Figure 2-5 in the previous section but consistent with Figure

2-6. This choice wasn't dictated by conceptual reasons, but I believed that it would greatly reduce the complexity of the system, both in computational terms and in explanatory terms.

To obtain the plausibility of a sentence from a lattice, the algorithm is simple: it sums together the weights of the nodes that correspond to all sentences that contribute to the sentence's plausibility. For example, the plausibility score of a sentence that describes a hawk flying in the sky is the sum of the weights of the nodes that correspond to the sentence where a hawk travels in the sky, the one where an animal flies in some gas, and so on.

In order to clarify this system further, consider the following example, related to Figure 2-6. Learning that a dove can fly in the sky will put a value of 1 at the "sky" node which is nested under the "in" node, which is under "roles", "dove" and "fly." If we assume the standard decay to be 0.9, it will put a value equal to 0.9 at the "fluid" node nested under "in", "roles", "dove" and "fly." Similarly, "thing" will see an even lower score. The same applies at the higher levels: everything nested under "bird" will have the same pattern as everything nested under "dove", but multiplied by the decay factor. If, again, we assume that all decays are 0.9, the node "fluid" under "in" under "roles" under "bird" under "travel" will have a score of 0.729. Suppose that now we want to inquire about the plausibility score of a pig flying in the sea. This will be equal to the sum of the score at "fluid" under "in" under "roles" under "animal" under "travel" summed with every other node reachable by walking up from that node. It is clear how that will score lower than a similar example where the flying animal is a hawk: the hawk also receive a large score from the fact that it gets all the "bird" points. In this example, I assumed that the decays were constant. As a matter of fact, that is not how UPP actually calculates the decay - the decay rate is the same, but the values are chosen so that walking a long path gives the same total value as walking a short path. This change was necessary because in WordNet it is common to find threads of significantly different lengths, and I did not want that to impact plausibility scores.

The fourth chapter of this thesis will show a few additional real-life examples, which will further clarify these claims about implementation.

### 2.6.2 The Context Map allows us to change contexts within the story

Section 2.5.3 pointed out an interesting special case: sometimes, plausibility is greatly influenced by the general context of the story. Furthermore, this context cannot be kept static, because as a story evolves some things may change. Context evolution may not be linear, and as contexts change often the bulk of the plausibility rules stay the same. This, perhaps, is the most problematic characteristic of context switches: as we change contexts, we should not get rid of all information that we have previously accumulated.

Story-contexts are also hierarchical, and they can be represented by threads. For example, the second act of *Macbeth* could correspond to this thread:

all-worlds fiction-worlds macbeth-worlds second-act

Then, we can have a lattice on top of all lattices where the nodes correspond to these story-contexts. This idea has two benefits. Firstly, it is very similar in implementation to the rest of the architecture. Secondly, it is very flexible in allowing users of UPP to make the context evolve with the story. With simple rules such as "If a major character dies, create a new child context and state that in that context that character cannot talk", our plausibility measures can dynamically evolve. At a first glance, encoding so many of these rules looks like a daunting task. However, a recent development within Genesis that allows the usage of common sense rules from ConceptNet[5] will make this process considerably easier and more generalizable. This contribution was part of Bryan Williams's Master's Thesis[9], and I will discuss later in further detail it can be used to further enhance UPP.

### 2.6.3 The plausibility-evaluating side of UPP satisfies three important principles

This section described how the plausibility-evaluating side of UPP can be tied together using role frames and how this all fits with a Context Map. We have now developed a system that can be used to resolve questions of plausibility, maintaining three important principles:

- This system learns from few examples and easily generalizes. A single sentence is sufficient to build realistic expectations.
- This system can produce meaningful results by purely observing examples, with negative rules being enriching but not required.
- This system robustly takes into account the different contexts that actions are part of, both explicitly and implicitly.

## 2.7 Laziness can reduce the complexity of a large lattice system

The architecture that I described in Section 2.6 is relatively simple to describe, but potentially difficult to hold inside a standard computer's memory. To see this, consider a sentence such as the simple "John eats chicken because John likes chicken". In its inner language, Genesis describes this sentence with the following, much more complicated, ensemble of entities:

(rel cause (seq conjuction (rel like (ent John) (seq roles (fun object (ent chicken)))))
(rel eat (ent John) (seq roles (fun object (ent chicken)))))

### 2.7.1 Lattice nesting is cursed by exponential size increases

With more than ten entities appearing, in approximate terms, a lattice will need more than ten levels of depth in order to represent the nestings properly. Each lattice needs to have a number of nodes of the order of the number of elements in each thread, which is normally between a couple and a dozen.

The real issue, however, is that as we go one level deeper, each node gives birth to a full lattice: that is, the number of nodes at the lowest level will be exponential in the number of levels. Clearly, even the simple sentence given as example in this section would cause our system to grow to the point of having ten figures worth of lattices. In order to combat this issue, I have employed two measures: procrastinating the expansion of nodes until it is necessary, and cutting the amount of double-nestings due to relations. In both cases, I would say that the key to reduced complexity is laziness: the principle of not doing anything unless it is necessary.

### 2.7.2 Not expanding nodes heavily reduces the number of lattices created

Any example of a flying dove is also an example of a flying columbiform bird, which is why as we learn something about flying doves we update the lattice under columbiform birds as well. However, until we see sentences talking about columbiform birds that are not doves, the information we have about columbiform birds is unnecessary. In general, while lattices are theoretically very large structures, these structures present significant uniformities until there are sentences that cause these uniformities to break. What this means in practical terms is that if we carefully track how the examples seen affect different parts of the lattice, then more often than not we don't need to access the lattice nested at any node because we can infer its structure from other lattices that we accessed. To bring this back to our example, the lattice nested around columbiform birds will look just like the one nested around doves, which means that there is no need to ever access it. As long as we don't need to access a node, we don't need to expand it: what I mean with that is that we don't need to initialize the lattice nested underneath it. If we ever do need to access that lattice, we can expand it then: each node will contain a history of the operations that were procrastinated until expansion.

Realistically, if nodes are expanded only as needed, the size of UPP will only grow as much as the examples actually draw meaningful regions in the lattices. So, our system can grow large only when it is actually necessary for it to grow large, meaning that most of the time its complexity can be contained.

### 2.7.3 Cutting double-nestings due to Relations can reduce the complexity of the most dangerous examples

Another remark about the complexity of lattice systems is that, out of all types of entity within Genesis, Relations are the ones that cause the largest computational costs. When a Relation is observed, the lattices store a nested lattice with the subject and to all the nodes of that subject lattice another lattice corresponding to the object is nested. Relations are often self-sufficient sentences, and multiple Relations in the same sentence are not necessarily related. Because of this, I decided to simplify the structure of UPP by allowing at most one Relation double-nesting per sentence: all deeper relations are nested with their subject only and learned as self-sufficient sentences as well. Unlike the late-expansion method, this practice is somewhat arbitrary and constitutes a real tradeoff. To understand this, let's go back to our sentence of John eating the chicken. Because of this simplification, reading "John eats chicken because John likes chicken" is the same as reading "John eats something in some condition because John likes something in some condition", as well as "John eats chicken" and "John likes chicken". While some information is lost, I still believed that this was a reasonable tradeoff to make, since the lattice downsizing is quite major. Figure 2-7 shows the structure of the nesting of lattices before this tradeoff and Figure 2-8 shows it afterwards.

### 2.7.4 UPP's computations could be performed in parallel

A recurring theme of this thesis, so far, has been that the structure of sentences in Genesis is elegantly hierarchical. Hierarchies are powerful because of their generalization properties: as we saw a few times, the same principles can be applied at many

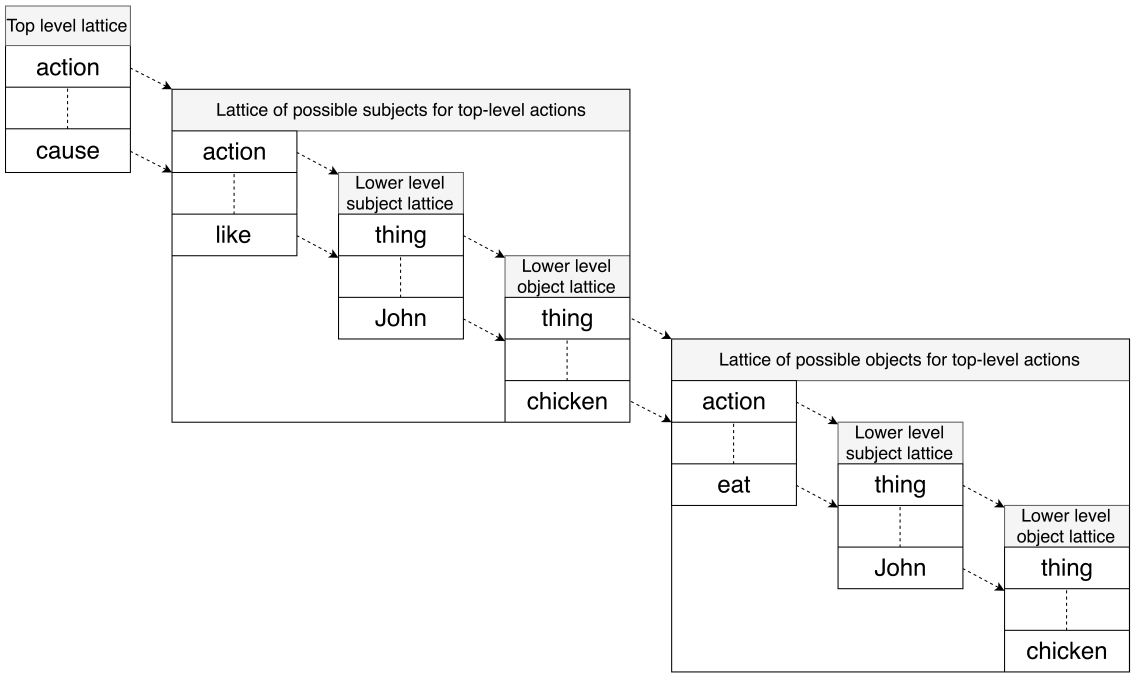
Top level lattice
action
cause
Lattice of possible subjects for top-level actions
action
like
Lower level subject lattice
thing
John
Lower level object lattice
thing
chicken
Lattice of possible objects for top-level actions
action
eat
Lower level subject lattice
thing
John
Lower level object lattice
thing
chicken

Figure 2-7: The lattice nesting represented after reading 'John eats chicken because John likes chicken", before cutting double relation nestings. Note that in this picture I skipped the nestings due to sequences with a single element (the nesting would be a little more convoluted because of "seq conjuction", "fun object" and so on).

levels inside UPP. However, hierarchies that exhibit recursive properties have an additional boon: as we proceed down the hierarchy, computations can be split across different processors. In the case of UPP, where space complexity is possibly more of a concern than time complexity, this means that different lattices can be stored in different memory units. I did not attempt to parallelize UPP, but I believe that if Frame Nested Learning evolves enough to become a primary component of Genesis, then this represents one of the main avenues towards a more efficient system.

## 2.8 Limitations

In Sections 2.6 and 2.7, I described how I gave the plausibility-evaluating side of UPP some structure and how I made it less susceptible to complexity issues. The choices I described in these sections, however, also came with a few drawbacks, which limit the extent to which UPP can draw information from some sentences.

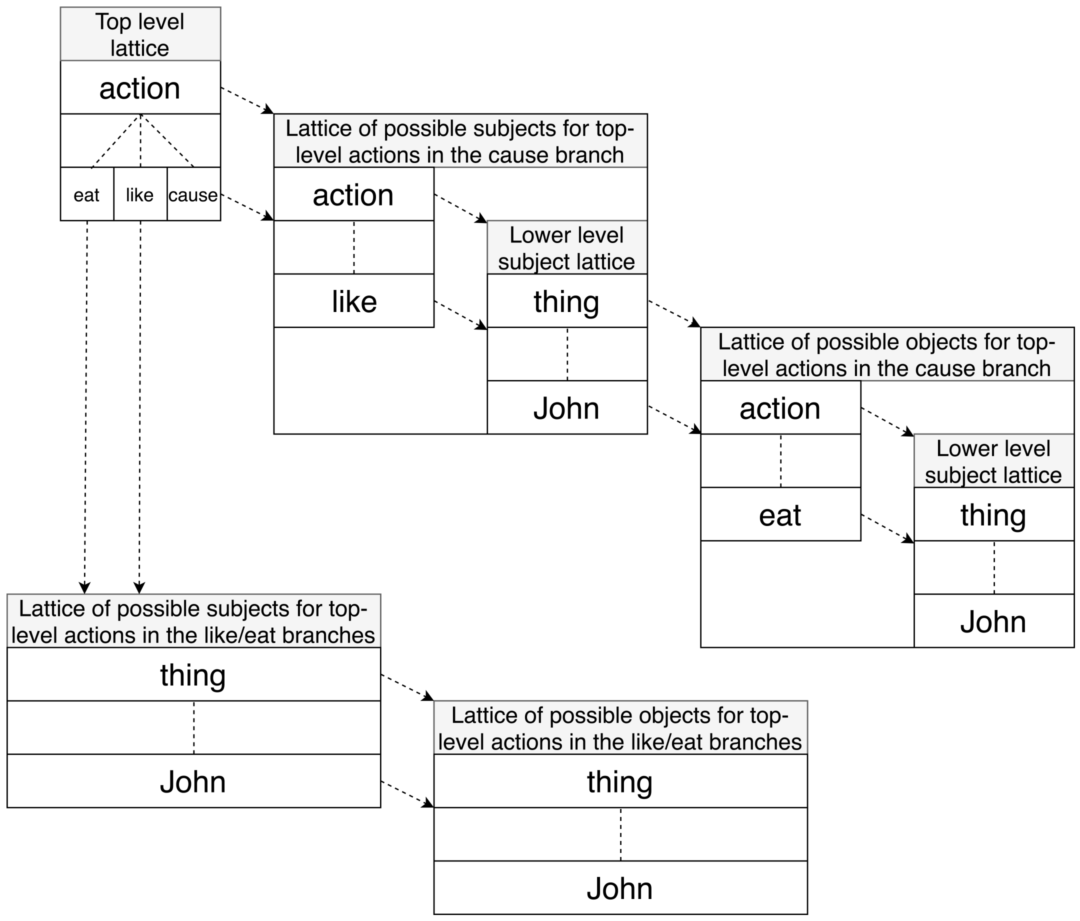
Top level lattice
action
eat
like
cause
Lattice of possible subjects for top-level actions in the cause branch
action
like
Lower level subject lattice
thing
John
Lattice of possible objects for top-level actions in the cause branch
action
eat
Lower level subject lattice
thing
John
Lattice of possible subjects for top-level actions in the like/eat branches
thing
John
Lattice of possible objects for top-level actions in the like/eat branches
thing
John

Figure 2-8: The lattice nesting represented after reading 'John eats chicken because John likes chicken", after cutting double relation nestings. Note that in this picture I again skipped the nestings due to sequences with a single element. Each lattice that we see in this picture is nested in multiple copies underneath each node of its parent lattice, so cutting lattice depth makes the number of total lattices exponentially better.

In Section 2.6, I mentioned the example of the sentence "The dove flies in the sky with a flower." A similarly structured sentence could be "The man kills the mosquito with the gun." While Frame Nested Learning gives structure to the lattices used to learn about this sentence, it also prevents some subtleties of the sentence from leaving the right mark in the lattice. A man can kill a mosquito, and can kill with a gun, but the two things just do not go together in the same sentence. This is something that UPP could understand, if it could nest different Sequence elements one under the other instead of at the same level—given the current structure, it cannot. On the other hand, nesting Sequence elements under each other would greatly increase the complexity of the lattices, both in terms of understandability and in terms of efficiency.

The fact that relations are not doubly nested also causes occasional issues: as far as UPP is concerned, saying that John eats chicken because he likes chicken is almost identical evidence as saying that John eats chicken because he likes pork. Again, this is something that UPP could understand more deeply if it had been constructed differently—and also something that could be changed painlessly. I believe, for example, that if UPP was parallelized, then it would become efficient enough to allow for some of the limitations I described in this section to be addressed.

This chapter described UPP's plausibility-evaluating side, but did not emphasize how this functionality is useful for the purpose of sentence parsing. In Chapter 3, I complete this picture of UPP, making it clearer how plausibility fits in the system. There are, however, many other ideas that can spark out of this plausibility framework: in Chapter 5 I discuss them in more detail for the interested readers.

# Chapter 3

# UPP generates plausible sentence interpretations

At this point, you know that UPP generates interpretations for sentences and then evaluates their plausibility. You should also have a clear idea of how UPP represents and evaluates plausibility using Frame Nested Learning and the Context Map. The main principle to keep in mind going forward is that UPP leverages the power of expectation in both of these tasks.

In this chapter I discuss in more detail the interpretation-generating side of UPP. I start by arguing for why the other side of the system is not sufficient for the purpose of making a human-like efficient parser. Then, I discuss how UPP combines a greedy approach with high-level optimization to walk an efficient path to generating acceptable sentence interpretations. I discuss how it is crucial for this side of UPP to interact with the lattices that were the main topic of Chapter 2. Finally, I explain how UPP, everything considered, builds a unified measure of plausibility, which allows it to pick an interpretation. I exemplify how this is done in the following chapter, Chapter 4.

After reading this chapter, you will be significantly more familiar with the design on the interpretation-generating side of UPP. Any questions you might have from the introduction will probably be answered. You will be able to explain how this module of UPP works to any kind of audience, regardless of their technical expertise.

## 3.1 When measuring plausibility is not enough

When I first approached the design of UPP, I was convinced that a robust lattice learning system like the one I described in Chapter 2 would be sufficient for the purpose of parsing ungrammatical sentences. Generally, a sentence is grammatically incorrect in one of only a few ways: either verbs and nouns are mismatched, or words are ordered in a faulty way, or there is something missing. All of these issues can be solved by reshuffling, adding token words, and making the sentences simpler and simpler until the START parser can read them. If this were satisfactory, then there would be no need for expectation in interpretation generation: all interpretations would be the same. This was my original approach, until I realized that there was something inhuman about it. In order to illustrate my point, I give the following example:

*The lion attacks the tiger because the tiger the lion scares.*

This sentence, while ungrammatical, is pretty simple to understand for humans. We could shuffle "The lion" with "scares" and we would get something perfectly grammatical. Consider now the following sentence:

*The lion attacks the tiger the tiger because scares the lion.*

Now, the sentence is very awkward to read, and seems very severely ungrammatical. It's quite surprising to notice that if we shuffle "because" with "the tiger", the sentence becomes perfectly grammatical, just like in the other sentence that we considered.

### 3.1.1 The hierarchical structure of sentences explains why not all shufflings are the same

The comparison between these two sentences should make something clear: two inversions in two different spots change our behavior considerably. In order to understand why this is the case, I find it useful to refer to Chomsky and Berwick's book, *Why*

*Only Us*[1]. Chomsky and Berwick propose a view of language that is rooted at the so-called Merge operation. Humans can combine basic sentences into more complicated sentences, possibly with a connector, and this is what allows the existence of complex structures. Going back to our lion-tiger example, the original sentence could have been originated by merging, with a causal connection, a sentence where a lion attacks a tiger and one where a tiger scares a lion.

I find Chomsky and Berwick's proposal to be most compelling, and I think it explains very well why switching "because" with "the tiger" is a much more destructive change to the sentence structure. The causation, here, is the connector between two subsentences: any ungrammaticality that is confined within one of the subsentences can be explained, but something that disrupts the hierarchy of sentences is on a different order of magnitude.

### 3.1.2 UPP must be designed as a hierarchical parser

I would argue, in fact, that UPP should be designed in a way so that it can only interpret sentences where the ungrammaticality doesn't confuse different subsentences together. It is useful for Genesis to be able to interpret a sentence only to the extent that the sentence is humanly understandable: otherwise, there is not even a benchmark to assess whether the interpretation is right or not. With this in mind, I decided that UPP would necessarily have to approach the interpretation of sentences in a hierarchical way: at each level, we need to identify the connector and recurse on the two sides.

This chapter will describe how such an approach can be designed, and how human expectation has a role in the process of identifying the connectors.

## 3.2 Dividing and conquering the hierarchy of sentences

In order to approach the problem of generating sentence interpretations, I start by considering the human approach towards the lion-tiger example of the previous section. As a reminder, the sentence considered was this:

*The lion attacks the tiger because the tiger the lion scares.*

At a first glance, we note that the sentence is expressing a causation. This brings us to dividing our problem into two smaller subproblems. On a side of the “because”, there is a grammatically correct sentence, “The lion attacks the tiger”. On the other side, we try parsing “The lion the tiger scares”. Then, we can produce two valid interpretations for this sentence: in one, the lion scares the tiger and in the other it is the tiger scaring the lion.

### 3.2.1 The hierarchy of sentences mirrors the nesting of lattices

As we split a sentence into subproblems, its structure begins to look more and more like Figure 3-1.

The most salient aspect of this picture is, in my opinion, its striking similarity with the nesting of lattices that I have described in the previous chapter. This is extremely beneficial to our interpretation: for each level of the recursion, we can access the lattice in UPP that corresponds to that level. This access means that we can see, at that level, what entities are more likely to be used and which ones should be ruled out. As a side remark, this hierarchy also lends itself to the same parallelization strategy that I had suggested in the previous chapter.

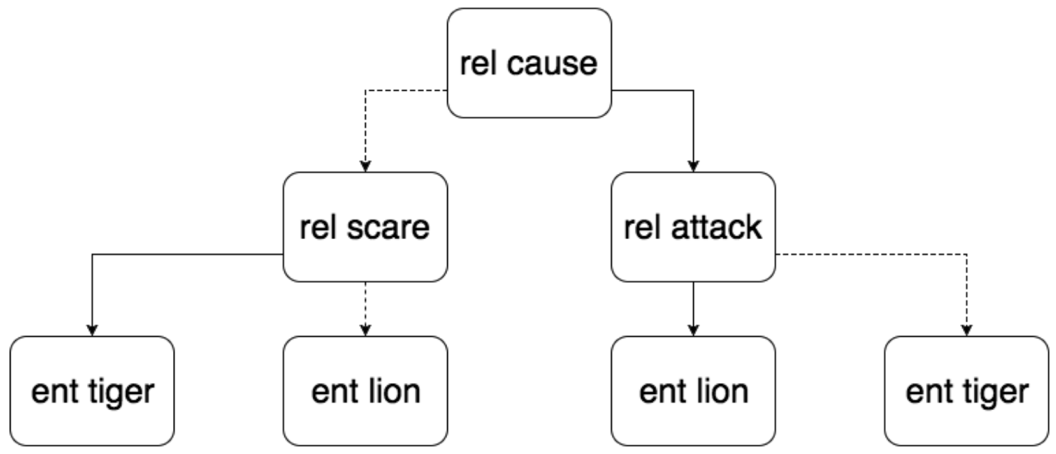

Figure 3-1: The structure of the lion-tiger sentence after it is recursively broken down. The dotted lines indicate the presence of additional nestings that were considered less important, such as “fun object” and “seq roles”.

### 3.2.2 Expectation is what makes humans, and UPP, fast

Having access to the relevant lattice is important because it allows UPP to entertain different possibilities according to what is closer to expectation. In the case of the lion-tiger sentence, this is what allows us to start looking for a causation at the highest level, instead of a subject-verb-object construction where the verb is “to attack”. I argue that heavy usage of expectation is what makes humans particularly fast at this task: before even looking at a sentence, people will know, to some extent, what is coming.

UPP takes advantage of this in a few ways: at every level, a choice needs to be made regarding what type of structure is most plausible and what terms should the structure revolve around. At the highest level, the sentence is most likely to be a Relation and revolve around a verb or a connector such as “because”. The object of a Relation is most likely to be either a Roles Sequence or another Relation. All this information is automatically drawn from experience into the lattices and can then be used to guide interpretation.

Now that all core principles have been exposed, the next section discusses the implementation of Candidate Interpretation Generators, the modules of UPP that are responsible for the task.

## 3.3 Candidate Interpretation Generators use a greedy approach without overcommitment

In UPP, the generation of sentence interpretations is a task carried through by Candidate Interpretation Generators (CIGs). Whenever a sentence is given to UPP as input, UPP creates a CIG responsible for that sentence. The CIG has the task of deciding if the sentence can be interpreted directly or if it should be split into subunits.

A CIG has four different ways of approaching a sentence:

- Apply START to the sentence, which will work correctly if the sentence is grammatically correct, or if it has been broken down to the point of being a single Entity.
- Guess that the sentence is a Relation. Then, guess a verb that could correspond to the relation's thread, and split the sentence in a subject-relation-object form. Two new CIGs are created to approach the subject and the object.
- Guess that the sentence is a Function. Then, because functions typically fit as elements in a roles sequence in a relation, construct a sentence in which the function could fit, and try to approach that.
- Guess that the sentence is a Sequence. There are different methods going from here, distinguishing for example between Roles and Conjuction Sequences. The Sequences are broken into their elements, which are approached one by one by new CIGs.

All guessing methods allow for some amount of guessing and reshuffling, which is what allows the CIGs to identify the sentences even when they are ungrammatical. The main question, now, concerns which approach should be picked, and how far the CIG should go in any direction before backtracking.

Picking approaches, just like every other "guess" described in the method, is always based on expectation. As CIGs are created, they tell each other the relevant point in the lattice structure. This way, each guess comes from either examples (guessing the most common thing given a lattice structure) or other inferences made via the lattice structure. Thanks to their expectation-driven nature, CIGs normally tend to produce the most plausible sentence interpretations first, which speeds up the process. As I had already mentioned, heavy usage of expectation (as opposed to checking all possibilities) is what makes humans fast at this task, and is also what can improve CIGs considerably.

### 3.3.1 Knowing when to stop, and identifying wrong intuitions

CIGs are driven by expectation. What happens, however, when the expectation is wrong? Not all sentences have the same structures, even when the structure at the top is the same. While guessing using the lattices is correct most of the time, occasionally there are fixes to be made. Similarly, when humans try to interpret a sentence, they will occasionally be wrong, recognize their mistakes, and try a different approach.

For CIGs, this isn't an easy process. CIGs are made to analyze ungrammatical sentences, so there are few things that will be recognized as so wrong that no attempts are worthwhile. CIGs are structured with a principle of biological plausibility in mind—no sentence interpretations should be found if a human cannot find one. While this principle has always been satisfied in my tests, CIGs tended to fail in different ways than humans. Humans stop before going down a useless rabbit hole, while CIGs initially attempted to create several implausible possibilities and failing at them. I then decided to design CIGs so that they will analyze a sentence in multiple passes: each CIG has a sequence of steps that they will try, and they will return their output in batches at the end of each of these steps.

The original CIG will be called multiple times until a timer has expired or until a satisfactory interpretation has been produced. This CIG will guess that the sentence is grammatically correct, then guess that the sentence is of the type suggested by the lattices, and finally guess that the sentence is of a different type. Between each of these guesses, the CIG will return its output. All CIGs that are created by this CIG will also be called multiple times—once the original CIG is unsuccessfully done with all its attempts, it will try once more allowing its children CIGs to go deeper. In a sense, the CIG architecture performs a greedy search, but any search path is stopped after a while and restarted if no other path was successful.

Thanks to this structure, CIGs rarely go down deep rabbit holes. They will, however, be stuck if the sentence is so convoluted that no interpretation is possible. In that case, CIGs will progress their slow search until a giving-up clause is enacted, thus reporting that no interpretation could be found.

### 3.3.2 The other side of plausibility: quantifying grammatical error

As CIGs are created and executed, they occasionally perform switches and attempt calls to the START translators. Considering the process as a whole, the sentences translated by CIGs will rest on some atoms translated by START. CIGs keep track of how deeply they had to switch terms in order for START to succeed—in other words, how much START has failed in the attempt of translating the sentence. While arbitrary, this measure allows CIGs to provide a value for the amount of grammatical incorrectness present in any sentence interpretation—which is factored in the final measure of plausibility.

# Chapter 4

# Results

At this point in the thesis, you should be familiar with how UPP works to interpret sentences. However, you have not seen UPP operate in practice nor you have seen any representation of the lattices that it builds.

The goal of this chapter is to illustrate some of the results of UPP. This chapter will show the lattices that are created after sentences are seen by UPP. Later, this chapter will use UPP's output to describe the decision process that goes in the interpretation of sentences.

After reading this chapter, you should be more familiar with how UPP works in practice. Also, the real examples provided here should make all the previous discussion more concrete and easier to understand. You will also be more convinced that UPP can deal with actual ungrammatical sentences using few correct examples.

## 4.1 UPP can build meaningful lattices with few examples

This section shows the lattices that UPP builds after seeing a few examples. These are the output of UPP's simple GUI, which I made for the purpose and testing and demonstration. In the first case, UPP is given two positive examples, one of which is a simple one-level sentence and one of which is a nested sentence. In the second case, we will consider another simple case that contains a positive and a negative example. Because the GUI is simplified compared to some of the figures that I have shown in this thesis, I first provide an example that should clarify how the two representations are related. Figures 4-1 and 4-2 represent the lattices after seeing the example sentence "The cat eats the mouse."

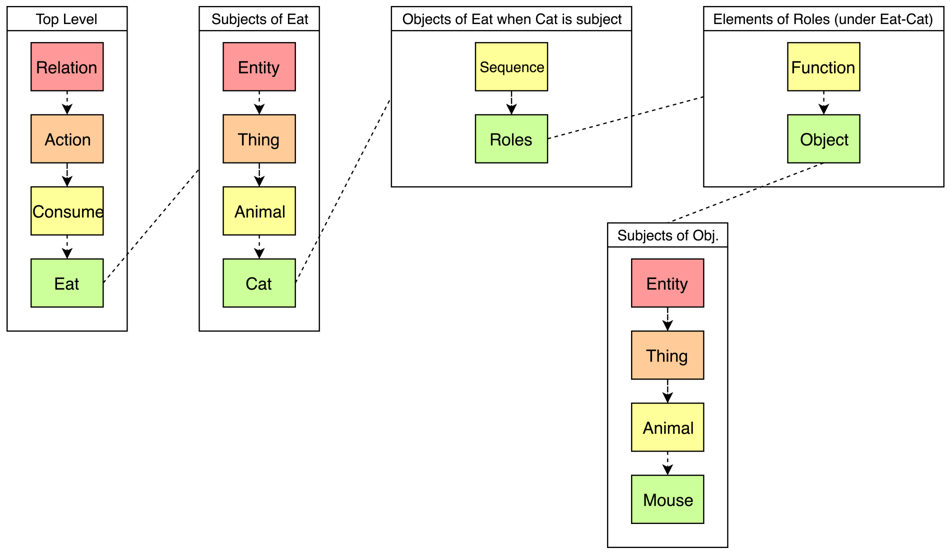

Figure 4-1: A figure that tracks the ones seen before in this thesis. Is shows a nesting of lattices, and it shows the nestings explicitly with dotted lines.

### 4.1.1 The first case: only positive examples

In this case, I trained UPP using the two following sentences:

*John eats an apple.*

*The wolf eats the fish because the wolf wants food.*

Both of these examples are assumed to be taken out of a context that is called the *real-world* context. For the sake of example, however, I added another context to the map, which I show in Figure 4-3.

As the second chapter of this thesis explained in greater detail, the context map can be treated as one extra layer of lattice nesting. We now consider the lattice nested under the *real-world* context node, which is the one directly affected by the examples

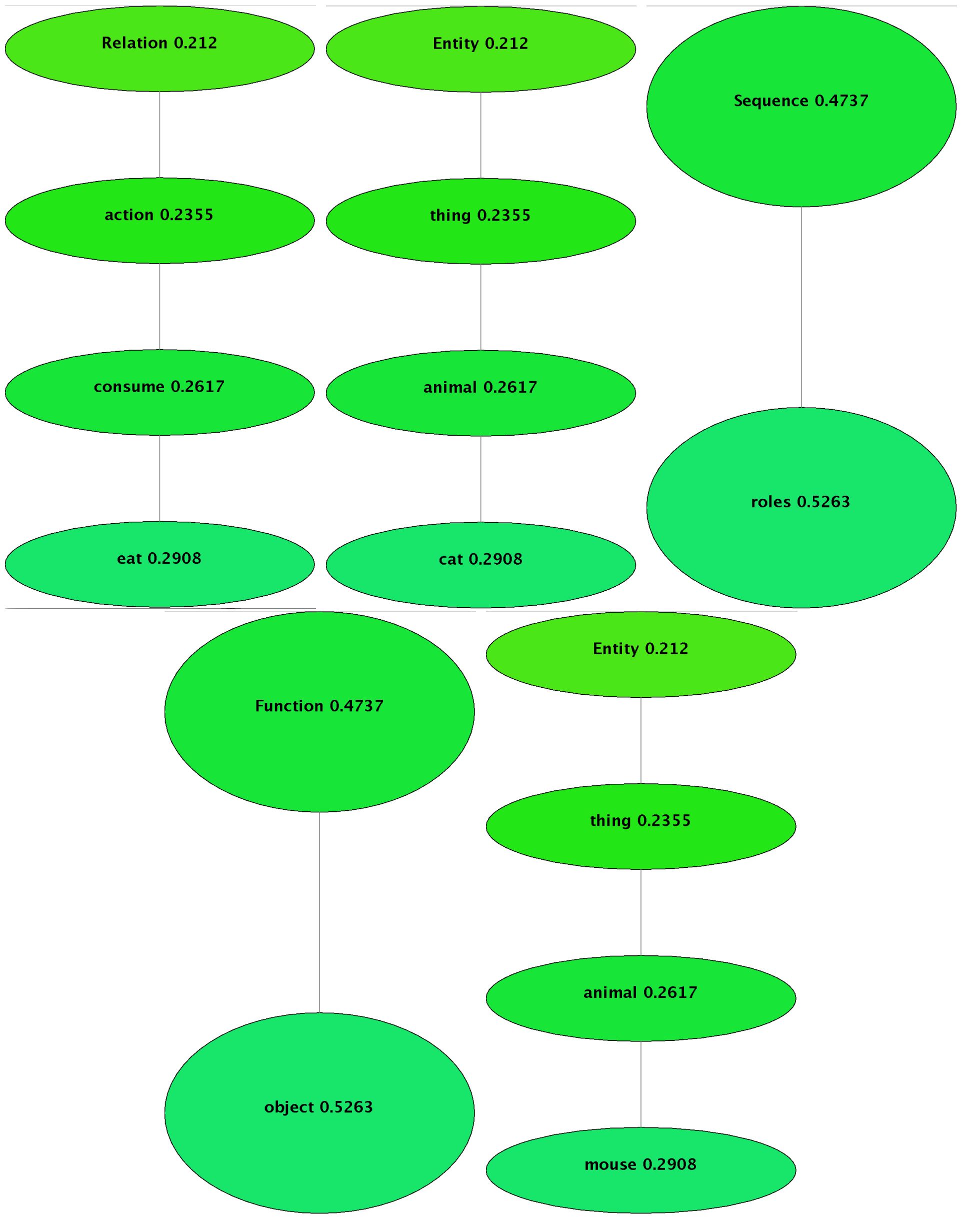

Figure 4-2: A figure that shows the lattices as they are represented by the GUI. The first diagram is the lattice at the top level, showing that the only sentence we have seen is an eating relation. By clicking on any of the nodes, the GUI shows the lattice nested underneath it: for example, if we click on the "eat" node, we see the lattice of the subjects of eating in the second diagram. By clicking on "cat" we see the objects of eating when "cat" is the subject: that is, a roles sequence which in turn contains an object function whose subject is then shown in the last diagram. Note how this structure is just the same as the one shown in Figure 4-1, just without the explicit labels and links. In the rest of this chapter, there will be many figures like this one, and I often represent only a part of the lattices: for example, I often skip the middle lattices corresponding to "sequence roles" or "function object."

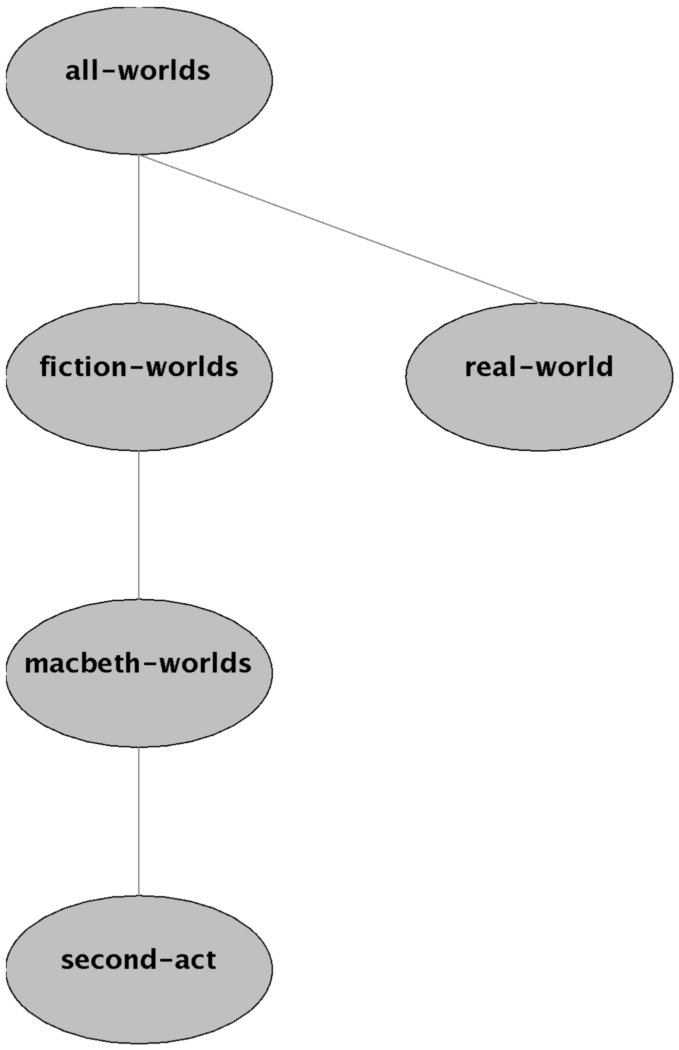

Figure 4-3: The context map. Here there are two contexts, one real and one imaginary, and the examples that we are considering both belong to the real world.

that we learned. Note, however, that anything we see in this lattice will appear, with smaller numbers, in the lattice nested under the *all-worlds* context node. Because of that, the examples that we are learning affect our knowledge of what happens in the second act of Macbeth as well, just in a less direct way. Similarly, anything that we see in Macbeth will change our knowledge of fictionary worlds, which in turns affects our expectation of reality. Figure 4-4 shows the highest level of the *real-world* lattice.

While the numbers seem arbitrary, looking at them more closely reveals something about how the lattices tell us about sentence plausibility. The lattice shown in Picture 4-4 is the one whose goal is to measure the plausibility of what is the highest-level

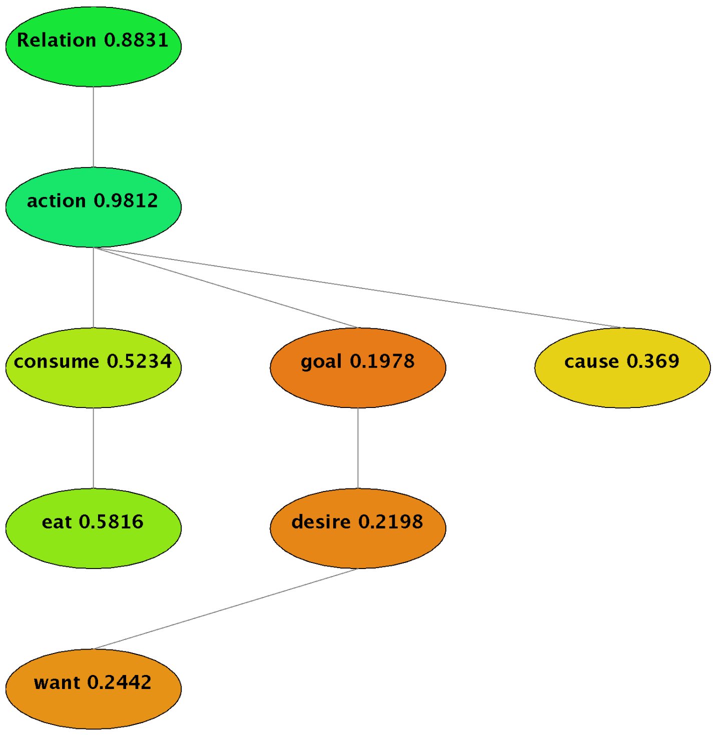

Figure 4-4: The highest level of the *real-world* lattice. As we see, there is only one root, which corresponds to the "Relation" node. This reflects the fact that all the complete sentences that we have seen are Relations. This will inform our decision when we attempt to interpret a sentence: the complete sentence is likely to be a Relation. There are three different branches that descend from this root, corresponding to the actions of eating, causing and wanting. These are the three things that we have seen (note that wanting appears in a subsentence, but that subsentence might exist as a complete sentence, and that eating has a higher score because it appears twice). This figure and the ones that follow should mirror exactly the scheme I showed in Figure 2-8.

structure of a sentence. Note the high score at the "action" node: this means that it is very plausible for the highest-level structure of a sentence to be an action. In the two examples that we have seen, the action of eating appears twice: this is why its node is so highly ranked. It doesn't matter as much that one of the occurrences of eating is not at the top level of a sentence—being a Relation, that subsentence could be a top-level sentence if it hadn't been merged with another. We further investigate the lattice by opening what is nested at the "cause" node. Figure 4-5 shows a sequence of lattices that are nested there.

Then, we investigate the lattice by opening what is nested at the "eat" node.

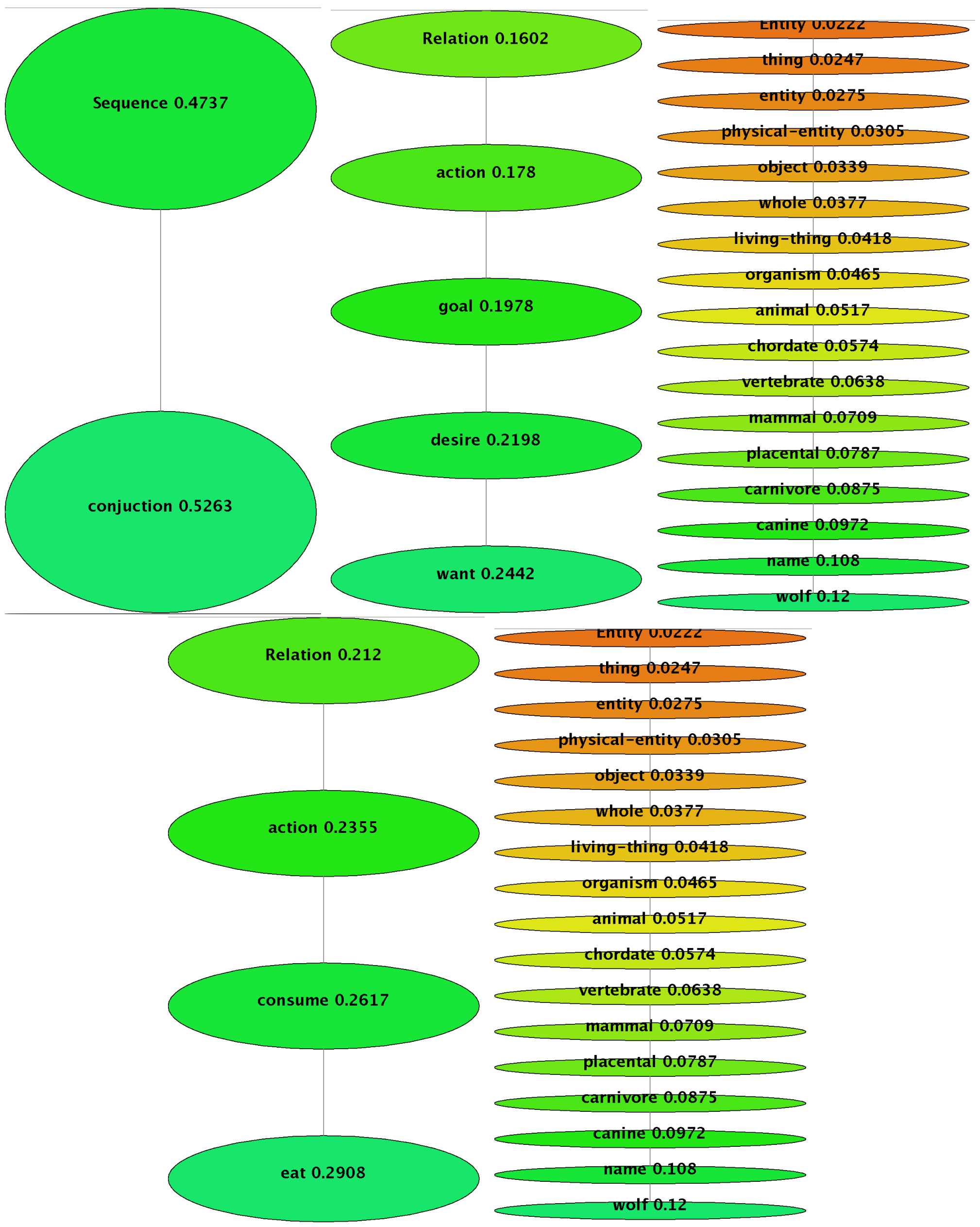

Figure 4-5: This figure shows the path of lattice nestings. Under “cause” (shown in Figure 4-4), there is a nesting of two lattices, the one corresponding to the subject of the action of causing and one corresponding to the object of the action of causing. The subject of the action of causing is a conjuction sequence (lattice shown in the top left), which is indeed directly nested under “cause”. This sequence contains one element, the Relation of wanting (top center): hence, this is nested directly underneath it. As Chapter 2 explained, to reduce complexity I made it such that nested relations do not cause additional double nestings: for this reason, the only lattice nested under the “want” node is its subject “wolf” (top right). Nested under wolf, is the object of “cause”, which has not yet been added: the action of eating (bottom left). Under that is the “wolf” subject (bottom right).

Figure 4-6 shows a sequence of lattices that are nested there.

A few interesting things are noticed as we look at this second lattice nesting and even more so as we compare it with the one shown in Figure 4-5. Strangely, the John node's parent is a "toilet" node, because according to WordNet the primed thread assigned to John is a toilet. This shows the need for Genesis to perform good disambiguation, which UPP could do given some proper training. On the other hand, it shows a weakness of UPP, which needs to be fed positive examples with carefully set primed threads.

Another thing that we notice involves the fact, which I also describe in the figure's caption, that the lattices nested underneath the "whole" node and the one underneath the "John" node are different. According to the lattices, John is able to eat a fish—because he is a whole and wholes are currently considered able to eat fish. However, he is "much more able" to eat an apple, because he can do that by virtue of being a whole as well as by virtue of being a John.

A similar behavior appears when we look at the two occurrences of the action of eating. An "eat" node appears both at the top level of the lattice and nested as an object of the action of causing. Eats appears differently, however, in the two locations, which is why the lattices nested underneath these nodes are different.

### 4.1.2 The second case: positive and negative examples

In this case, I trained UPP using the two following sentences:

*John eats an apple.*
*A fish eats paper. (negative)*

The negative example is to say that it cannot be the case that a fish eats paper—it doesn't necessarily mean that paper cannot be eaten or that fish do not eat. Let us investigate what the lattice looks like in this case, in Figure 4-7.

The figures shown in this section are true to the statements of Chapter 2—plausibility is a question that can be answered by thinking in terms of "in virtue of what is this possible". As we see in the figure, if we walk down the lattices looking

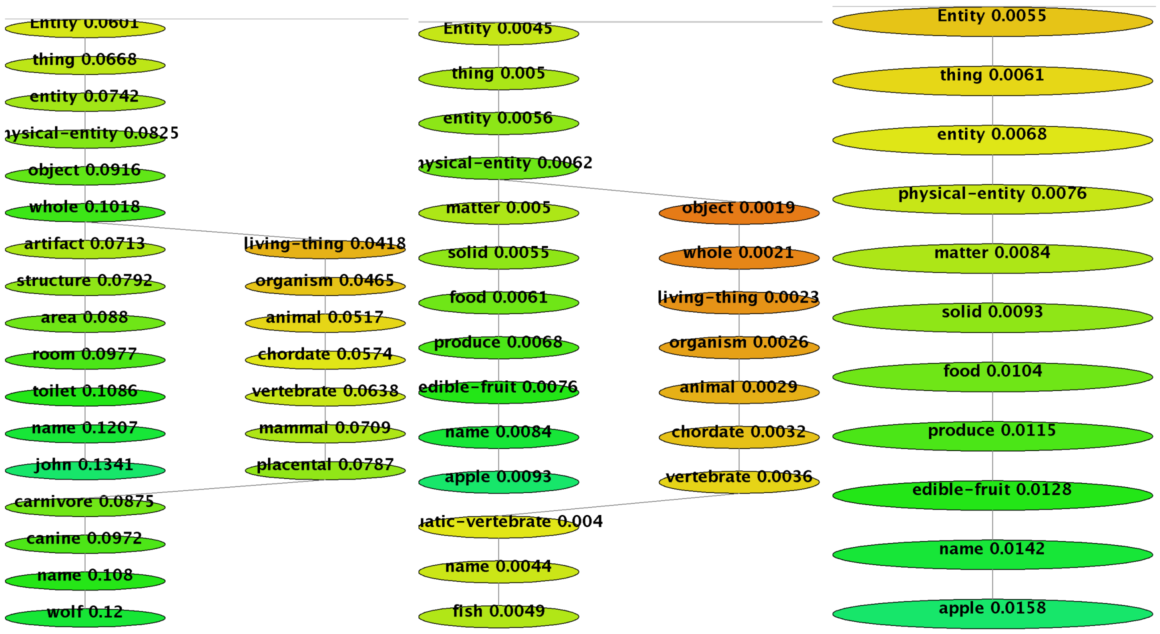

Figure 4-6: This figure also shows the path of lattice nestings. Under "eat" (shown in Figure 4-4), there is a nesting of two lattices: one corresponds to the subjects of eating and one corresponds to the objects of eating. Directly under the "eat" node is the lattice that corresponds to the possible eaters (left). John and the wolf are both "wholes", so under the "whole" node we see a nested lattice (center) which contains both apples and fish: these are the things we know "wholes" can eat. On the other hand, under the "John" node (in the lattice on the left) is only an apple thread (right), because that is the only thing that we are aware of John being able to eat. The figure skips all lattices that are less relevant, such as "function object" and so on.

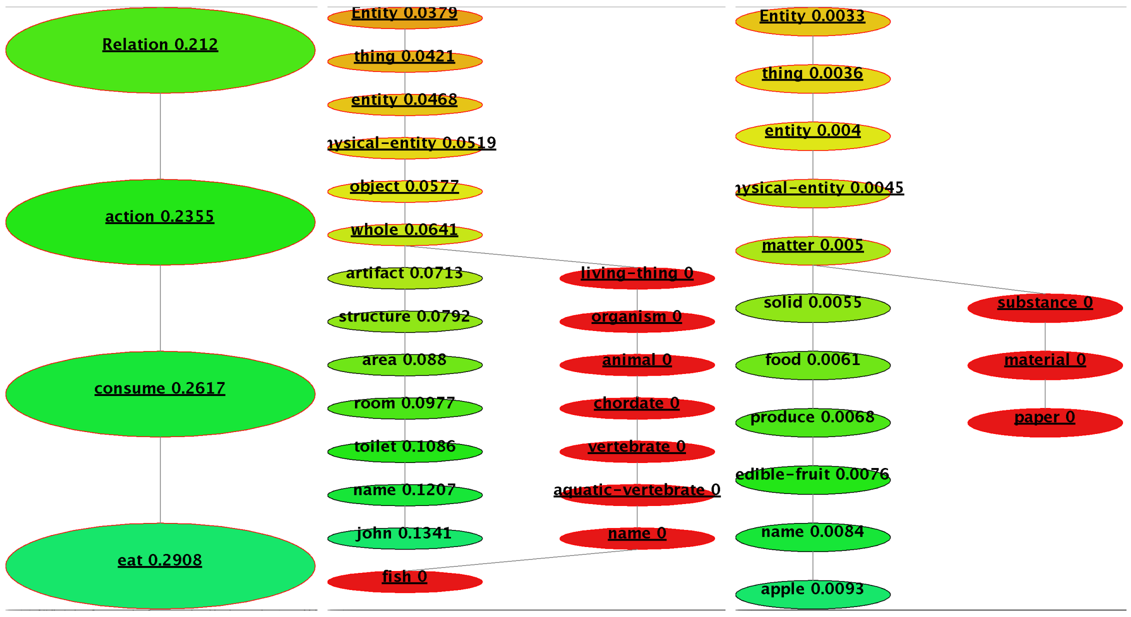

Figure 4-7: This figure shows the path of lattice nestings in a case in which we use negative examples. Oval borders marked in red and underlined text represent a might-not-be-plausible encoding. This is an important detail that the reader should be well aware of: the coloring inside the oval is a function of the quantity of positive examples, while the coloring of the borders (and the underlining) depends on positive and negative examples. In other words, the borders encode the inference of traditional lattice learning. In the description of this figure, with “marked as negative” I refer to these borders and to the underlining. While this representation is slightly different than the one seen in Figure 2-4, the decision process is identical. We have received an example of an eating Relation that is marked as negative: this means that a sentence whose main action is to consume or eat might not be plausible (consequently, in the drawing on the left the all the nodes are marked as negative, even if the insides are green because of the presence of positive examples). If we open the lattice nested at “eat”, we see the possible subjects of the action of eating (center). John is not marked as negative—as far as we know, all the examples of John eating are positive. On the other hand, “whole” is marked as negative because we have a negative example of an eating whole. In the lattice underneath “whole” (right), the “apple” branch is not marked in black as there are no rules that would affect the plausibility of apple-eating wholes. However, the “paper” branch is all negative because there is a negative rule derived from the fish sentence.

for whole beings performing the action of eating, we see that there is some plausibility if the whole being eats an apple but not if it eats paper (because we have positive examples of wholes eating apples and negative evidence against wholes eating paper).

## 4.2 UPP can use the information stored to interpret ungrammatical sentences

The previous section explains in practice how the lattice looks after UPP sees a small number of examples. In this section, I use UPP to interpret two ungrammatical sentences. When interpreting sentences, UPP produces understandable output which can be used to follow UPP's reasoning. For sake of clarity, I describe them in words instead of attaching them.

In both of the examples, UPP had been trained with just three positive examples:

*The cat likes the dove.*
*The wolf eats the fish because the wolf wants food.*
*John marries Mary because he loves her.*

### 4.2.1 UPP interprets a simple ungrammatical sentence with only one verb

The first example that I provide is the following sentence:

*The pigeon the fox eats.*

First of all, note that there are two grammatical interpretations of this sentence that jump to mind:

*The pigeon eats the fox.*
*The fox eats the pigeon.*

Note, also, that fox and pigeons never appeared in the examples given to UPP, showing the generalization abilities. The reasoning works as follows:

1. Approach the sentence: at the top level, what is the most likely thing that can occur?
    - A Relation, so UPP approaches the sentence as a Relation.

2. Guess that the sentence is a very simple relation. Among verbs in the sentence, which one is most likely to be the verb that the relation revolves around?

    - The verb eats, so now UPP approaches the sentence as if it was an eating relation.

3. Given that we are interpreting an eating relation, how could the sentence be split so that there is a subject and an object of the action of eating?

    - In one option, "The" is a sub-phrase and "pigeon the fox" is another sub-phrase.
    - In one option, "The pigeon" is a sub-phrase and "the fox" is another sub-phrase.
    - In one option, "The pigeon the" is a sub-phrase and "fox" is another sub-phrase.
    - All possible sub-phrases fail to be interpretable phrases, except for "The pigeon" and "the fox". Those are interpreted as entities that can be either subjects or objects.

4. Two sentence options are produced, "The pigeon eats the fox" and "The fox eats the pigeon."

5. Out of all sentence options, all those that satisfy a minimal degree of plausibility are selected as output. This includes both sentence options. Because the number of options produced is nonzero, the Candidate Interpretation Generators have completed their tasks.

6. The two sentence options are compared for plausibility. They have the same degree of grammatical distance from the original sentence, and "The fox eats the pigeon" has a greater lattice plausibility than "The pigeon eats the fox."

7. UPP returns "The fox eats the pigeon" as the best interpretation.

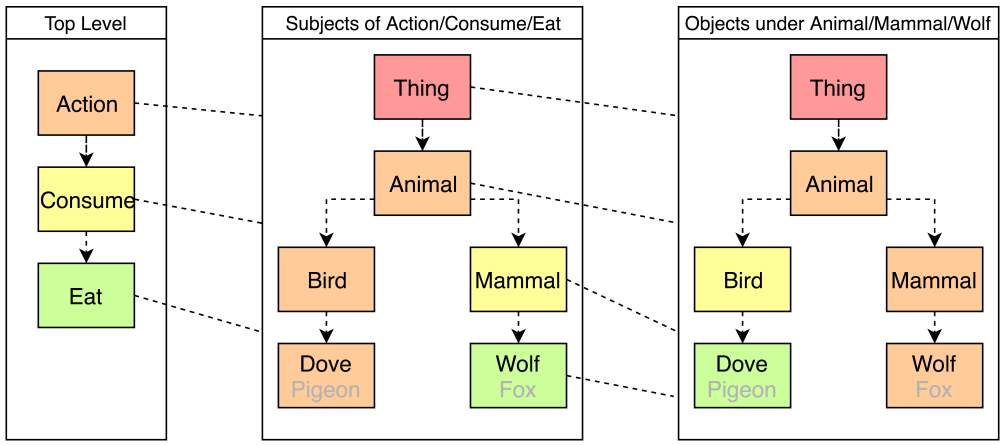

Figure 4-8: A simplified version of the lattices that are relevant to this explanation. At the top level, the only relevant action is the one of eating. Among the subject of acting, consuming, or eating, mammals (especially wolves) are more plausible than birds. Among the possible objects of “being eaten by a mammal”, birds are more plausible than mammals.

The sixth item of this list deserves some attention in itself. To clarify what happens, we can look at Figure 4-8, and follow UPP as it walks down the lattices to establish the plausibility of the two sentences. At the top level, both sentences involve the action of eating, so they are identical. UPP opens the lattices nested under “action”, “consume” and “eat”. In these nested lattices, UPP finds the possible subjects of the actions. From our examples, we have some evidence that wolves eat, which translates to the fact that mammals probably eat. A sentence that involves an eating fox will, then, be scored higher than one with an eating pigeon, because the fox is a mammal. It does not, however, end here: the sentence is not finished. Nested under “mammal” there is a lattice of the possible objects of eating-by-mammal. Here, we see that birds are a fairly likely target. On the other hand, mammals are not a plausible object of eating: they only are in virtue of the fact that animals can be eaten.

### 4.2.2 UPP interprets a more complex ungrammatical sentence with a causation and multiple actions

The first example that I provide is the following sentence:

*The dog the snake kills because the dog is strong.*

This example, while on the surface is similar to the simpler one, is interesting because the sentence "The dog the snake kills" is a more difficult one to parse with the examples given. In fact, even to us the sentence is difficult to parse on its own, as a dog and a snake could have a fight with either result. However, as we see the sentence in its entirety, we no longer have doubts: we are saying something about the dog that would qualify it to kill the snake, so the interpretation where the dog kills the snake is indeed more plausible.

This is the same principle used by UPP, even if in a fairly unsophisticated way. UPP has seen a wolf take an action because of another event that the wolf was subject of. This is enough for UPP to draw some conclusions about the fact that a dog is likely to do something because of some other event that the dog is a subject of. This lack of sophistication is mostly due to the fact that not many examples were given: with a larger set of examples, UPP will become more nuanced and be able to draw more complex distinctions. The careful reader might notice here that UPP could draw the conclusion just because of the fact that there are examples about animals that are evolutionarily closer to the dog than to the snake. I noted this subsequently, and repeated the experiment adding a sentence where another reptile performs some action, and the same result was obtained.

In more detail, UPP's reasoning worked as follows:

1. Approach the sentence: at the top level, what is the most likely thing that can occur?

    - A Relation, so UPP approaches the sentence as a Relation.

2. Guess that the sentence is a very simple relation. Among verbs in the sentence, which one is most likely to be the verb that the relation revolves around?

    - The verb dog, so now UPP approaches the sentence as if it was an "dogging" relation.

3. All simple interpretations of sentences that use “dog” as the main verb fail to be understandable.
4. UPP approaches the sentence as something simple that is not a relation.
5. UPP fails to correctly interpret the sentence as anything that is simple and is not a relation.
6. UPP approaches the sentence as something that is a Relation but is potentially more complex. Among single words in the sentence, how can the sentence be broken around words creating a relation?
    - The word “because”, so now UPP splits the sentence as a Relation of “the dog is strong” causing “The dog the snake kills” (or vice versa).
7. If “the dog is strong” is to be interpreted as a subject of a causation, what is it most likely to be?
    - A conjuction sequence that contains a relation. So, the sentence is understood as that, and it is grammatically correct.
8. If “the dog is strong” is to be interpreted as an object of a causation, what is it most likely to be?
    - A relation. So, the sentence is understood as that, and it is grammatically correct.
9. If “the dog the snake kills” is to be interpreted as a subject of a causation, what is it most likely to be?
    - A conjuction sequence that contains a relation. UPP works recursively on this ungrammatical sentence, giving rise to an interpretation where the dog kills the snake and one where the snake kills the dog.
10. If “the dog the snake kills” is to be interpreted as an object of a causation, what is it most likely to be?

- A relation. UPP works recursively on this ungrammatical sentence, giving rise to an interpretation where the dog kills the snake and one where the snake kills the dog.

11. A few sentence interpretations are created, and all of them are minimally plausible:
    - "The dog kills the snake because the dog is strong."
    - "The snake kills the dog because the dog is strong."
    - "The dog is strong because the dog kills the snake."
    - "The dog is strong because the snake kills the dog."
12. UPP judges the most plausible option to be "The dog kills the snake because the dog is strong."

Given an appropriate set of examples, UPP can interpret ungrammatical sentences like the ones that I just showed. While UPP worked well on the examples that I tested it with, the system still lacks the large scale testing that would be necessary for scaling up. I believe, however, that UPP in its current form is already a very large step towards this scaling.

## 4.3 UPP's complexity might grow unwieldy if more parameters are added

This chapter showed a few examples that UPP correctly classified, and that give some hope for UPP allowing Genesis to scale up towards more complex sentences. However, the dog-snake example that I just discussed raises an interesting point: what if we were overwhelmed with examples of snakes killing other animals? In that case, which is quite realistic because snakes are tendentially considered more deadly than dogs, UPP could be pushed to make a different call about the sentence.

This point illustrate what perhaps is a larger problem: UPP's decisions are based on a number of parameters that trade against each other in the attempt to make a system that is realistic. There is only so much, however, that parameters can do to make UPP more human-like, and adding more parameters can make a system more accurate but also more arbitrary. Table 4.1 illustrates the parameters used by UPP, by briefly stating their function and by giving the values that I used in most of my experimentation.

| UPP Parameters | | |
|---|---|---|
| Parameter name | Function | Value in Experiments |
| defaultDecay | The rate at which plausibility decays in lattices | 0.9 |
| maxCalls | Maximum number of calls to START | 150 |
| contextDecay | Rate of decay in the context map | 0.5 |
| decayExceptions | Any lattice-specific exceptions to decay | None |
| failuresPenalty | A penalty for more ungrammatical sentences | 0.9 |
| stoppingThreshold | Minimal plausibility to accept sentences | 0 |
| negativeRulePenalty | A multiplier applied for broken rules | 0 |

Table 4.1: Table of parameters in UPP.

While these may seem like many parameters, one fact is reassuring: in the cases that are most obvious to humans, UPP performs equally as long as the parameter choices are remotely sensible. Unfortunately, it is very possible to construct cases where UPP does not reason like a human would. Besides these parameters, what is most important is the set of examples that UPP can read. UPP can access anything that Genesis already understands, so as long as Genesis reads realistic examples UPP's decisions will also be realistic. Building a knowledge base of truly relevant examples that can address many areas of language, however, would be a very interesting and difficult endeavor of its own.

This section described some of UPP's successes with lattice creation and sentence interpretation, and also illustrated a potential weakness with the system. The rest of the thesis will introduce possible extensions to UPP and then summarize how this system constitutes an important contribution to Genesis.

# Chapter 5

# Extensions

At this point in the thesis, you should be familiar with UPP's design and implementation, and with some of the results that it achieved.

This chapter looks at the future: here, I talk about how the ideas in UPP are applicable to issues beyond sentence parsing. As part of this, I touch on the problem of disambiguation, on UPP as a story understanding tool and on other extensions that would make UPP more effective at what it does.

After reading this chapter, you will know of ideas for interesting future projects that improve on UPP or use it for something new. You will also know more about how the framework of plausibility can fit in the greater scheme of things within Genesis beyond UPP.

## 5.1 Beyond parsing: plausibility solves the ambiguities of language

As I said at the end of Chapter 2, the description of Frame Nested Learning and of the Context Map had very little mention of how plausibility is useful to the problem of parsing. Much of that is clearer at this point, after Chapter 3 and Chapter 4 described the rest of UPP and some of what it can do. There is, however, one important reason why I refrained from framing the question of plausibility evaluation specifically as a piece of the parsing problem. As I worked on designing UPP, I realized more and more how powerful a framework plausibility is, and how much plausibility can give us in terms of general story understanding tools.

This section and the next one discuss how plausibility is useful beyond parsing. This section, specifically, argues how UPP can essentially solve the problem of the ambiguities of language.

### 5.1.1 UPP's disambiguating capabilities go beyond the single word

In Section 2.1.1, I described Eli Stickgold's DISAMBIGUATOR and explained how it uses lattice learning to understand which of the possible meanings of a word are acceptable given a set of previous examples and rules. UPP can do the same: given any sentence, we can shuffle through the possible meaning of any word, and evaluate the sentence's plausibility to figure out which meaning is most likely to be correct. In this respect, UPP can provide what in my opinion is a more complete result than DISAMBIGUATOR: it has ways to take the entire context into account, and it can compare between many acceptable meanings by using plausibility on a scale. Also, UPP does not rely as heavily on negative rules.

Let us consider a comparison between two different sentences: the first one is "The wolf chases the rabbit because the wolf is hungry" and the second one is "The wolf chases the rabbit because the rabbit is hungry." If properly trained, UPP will understand that being hungry is a better motivation for chasing than for being chased, and will notice that the first sentence is more likely than the second one. Used this way, UPP can help us disambiguate the meaning of sentences such as "The wolf chases the rabbit because he is hungry."

Besides this point, I believe that UPP's disambiguating capabilities are remarkable because they go beyond the meaning of a single word. All the meanings of the words of a sentence can be shuffled through at the same time, and UPP will be able to understand which of the arrangements of meanings is the most plausible. This will help us decide how a sentence should be interpreted when the ambiguity goes beyond the single word.

## 5.2 Plausibility in UPP can be used as a story understanding tool

To humans, plausibility is not only used as a measure to discern meanings. As a matter of fact, I would argue that plausibility is primarily important when people reason about the world, when they evaluate events, and when they try to predict the future. The functionalities described in this section are speculations—I did not directly implement them. They could, instead, be thought of as short proposal for research projects which I believe could lead to interesting results. To facilitate any endeavor of this kind, I implemented UPP with some care so that it could be used as a routine by programs that tackle the problems that I am about to outline.

### 5.2.1 What does it mean when something is not plausible?

Much of the attention so far was given to the task of finding the most plausible interpretation of a sentence. However, I think that UPP can be most valuable in the opposite sense as well: that is, UPP can detect when events conflict with expectation.

#### UPP can detect major changes of scenario

If we abandon for a moment the scenario of complex sentences, encountering an implausible sentence might be puzzling. With no alternative interpretations to look at, the implausible interpretation needs to be considered for what it is: a mark of the fact that expectations are violated. This happens often in stories, and it typically signifies a major change in the scenario, or more generally an alteration of the rules of reality in the story. I think that these moments are moments of particular impact on the reader, and giving Genesis a tool to detect them is most valuable.

**UPP can tell us something about the reasoning of characters**

Just like the reader is occasionally surprised by a plot twist or an event that violates expectation, the same can happen to characters. If Genesis could truly understand a story from the point of view of a character, that would be an incredible accomplishment already—and I think that if that was possible then UPP would have a good deal to add. By analyzing the sentences that a character sees, UPP would be able to tell which events are expected by the characters, and which are perceived as surprising.

### 5.2.2 Plausibility can represent constraints, and help us predict actions

UPP's evaluation of sentence plausibility is, after all, an analysis of whether some event follows the rules of reality. In this sense, UPP represents what is possible, or at least what is likely to happen, in a world that obeys some rules and behaves according to some examples.

This view of plausibility lends itself easily to a broader topic: given some description of the world, what is possible in this world and what is not? If we can describe the world via rules and examples, then UPP can act as a "constraint checker": that is, a tool that verifies whether some event obeys the world's constraints. To see how this is useful, I want first to bring up two other essential pieces: Gyorgy Gergely's paper on learning as infants [2] and Bryan Williams's ASPIRE system, part of the same Master's Thesis that established the connection of Genesis with ConceptNet.

Briefly speaking, Gergely's vision is that even infants, who act mostly on the basis of emulation, perform an action selection process that can be mostly reduced to finding the simplest action within the world's constraints that gets them to their goal. Gergely's teleological framework is very intuitive, and it could be argued that even for adults most actions can be explained given goals and constraints. In a world whose complexity is small enough to be reduced to rules and examples, if we had access to a list of possible actions that achieve a goal, UPP could predict what the character would do. This list is where ASPIRE becomes important: given a story, ASPIRE infers characters' possible goals, and can list actions, via ConceptNet, that could be motivated by or fulfill those goals.

Combining UPP's constraint checking with ASPIRE's goal identification inside of Gergely's framework could give Genesis extremely interesting predictive power, which I think would be a important achievement.

## 5.3 Frame Nested Learning could benefit from the introduction of ConceptNet

This last section discusses what I think is the most important weakness of the plausibility-evaluating side of UPP, and proposes a way to strongly mitigate this weakness.

### 5.3.1 WordNet's hierarchical structure fails to capture connections

The entire lattice learning system rests on the hierarchical structure of meaning suggested by WordNet threads. While this hierarchy is the reason that allowed many of the valuable properties of UPP, it is also unfortunately true that it fails to capture some of the properties of meaning. Let's consider an example to clarify this point. A dog is a canine mammal, a vertebrate and so on. A parrot is a bird, which is quite far in the lattice from the dog. Both animals, however, can be pets to be kept at home—wolves and foxes, on the other hand, cannot. We might learn an example in which John, our character, buys a dog in a store. As humans, we infer from that that there exist pet stores—plausibly, one could buy a parrot in such a store as well. According to UPP, however, a dog is very similar to a wolf and not as much to a parrot: then, UPP would confidently say that buying wolves in stores is within norm.

Unfortunately, there isn't much to be done about this kind of problem within the current structure of UPP. This is because there is simply no information inside UPP according to which wolves and dogs would be that dissimilar, or according to which dogs and parrots should be similar. Ideally, it would be possible for UPP to learn these hidden links, but after some experimentation it appeared that such a possibility would open too many degrees of freedom and hamper UPP's generalization capability.

### 5.3.2 ConceptNet could supply the missing links between threads

Luckily, the solution to the problem presented itself when Bryan Williams connected Genesis to ConceptNet. ConceptNet connects words with using links coming from commonsense knowledge. It has in-built measures of similarity, which it can use, for example, to connect a dog with a parrot. It is not constrained by any sort of structure, which allows this links to be freely obtainable and flexible. I think that ConceptNet could easily supply the hidden links between meanings that could solve the problem that currently plagues UPP. Each node in the lattice would have a few neighboring nodes, and this relationship would come directly from ConceptNet without affecting the lattice structure. Then, anything that we learn about dogs could ripple to neighboring pets.

While ConceptNet is powerful and it seems to fill the gap that UPP leaves open, there is one issue that needs to be considered. I already mentioned at the end of Chapter 4 how UPP is potentially vulnerable to issues of parameter tuning, and ConceptNet comes with many values of its own which are not designed to work with the numbers in UPP. I do not think that this should rule out ConceptNet from UPP, but it certainly represents a potential issue which needs attention.

# Chapter 6

# Contributions

In this thesis I described how UPP works and how it achieves its goal of interpreting ungrammatical and complex sentences. My goal now is to explain how my design and implementation of UPP was a contribution to research in and out of Genesis.

Without UPP, Genesis struggles to understand sentences that are made by humans for humans, because its reliance on the START parser makes it rely very heavily on a strict grammatical sentence structure. Genesis has the goal of developing a computational account of human intelligence by relying on story understanding. If we are to achieve such a goal, then it is crucial that Genesis can read sentences and stories that are truly made for humans. For this reason, I have designed and implemented UPP, which constitutes an important leap towards reading sentences made for humans. Thus, my contributions are the following:

- I designed a powerful framework for the representation of plausibility. I called it Frame Nested Learning, and it relies on the Lattice Learning method to represent to which extent any event can be considered plausible in a world described by examples and rules. Frame Nested Learning is based on expectation, and shows the same ability to generalize from small sets of examples that humans do. Compared to Lattice Learning, Frame Nested Learning is considerably more suitable to representing the world, because it has little reliance on negative rules, it represents plausibility on a scale and it is robust in the face of changing sentence contexts.

- I designed a system that generates interpretations for sentences in a humanly plausible manner. This system is made by modules that I called the Candidate Interpretation Generators, and it uses nested lattices to generate sentence interpretations in a way that is guided by expectation.

- I combined these designs into a single system that interprets ungrammatical and complex sentences, and implemented that system. I called the system Unified Plausibility Parser, because it generates sentence interpretations and then picks the best one using a metric of plausibility that unites grammatical plausibility and plausibility of meaning.

- I proposed several extensions to the idea of plausibility evaluation, which would make UPP useful beyond the issue of sentence parsing. I have leveraged UPP's plausibility evaluation tools towards the problem of word sense disambiguation, which UPP can approach in a more flexible way than existing systems. I have discussed how UPP can be used in conjunction with ASPIRE to make an action prediction system. I have also explored the idea of leveraging the common sense information in ConceptNet to broaden the scope of the ideas in UPP. Beyond the conceptual extensions, I have also built UPP in a way that lends itself easily to parallelization, which could significantly improve its performance. I strongly believe that the frameworks that I presented in this thesis are very flexible and could be applied to a variety of problems inside and outside of Genesis, and for this reason I have implemented UPP in a way that is easily understandable and ready for adaptation.

# Bibliography

[1] Noam Chomsky and Robert Berwick. *Why Only Us*. MIT Press, 2015.

[2] Gyorgy Gergely. What should a robot learn from an infant? mechanisms of action interpretation and observational learning in infancy. *Connection Science*, 2003.

[3] Josh Haimson. Language interpretation guided by expectation. Superurop final paper, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2016.

[4] Boris Katz. Annotating the world wide web using natural language. In *Proceedings of the 5th RIAO*, Conference on Computer Assisted Information Searching on the Internet, pages 136–159, 1997.

[5] Hugo Liu and Push Singh. Conceptnet - a practical commonsense reasoning tool-kit. *BT Technology Journal*, 2004.

[6] George A. Miller. Wordnet 3.0. *www.wordnet.princeton.edu*, 2006.

[7] Eli Stickgold. Word sense disambiguation through lattice learning. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2011.

[8] Lucia M. Vaina and Richard D. Greenblatt. The use of thread memory in amnesic aphasia and concept learning. *Massachusetts Institute of Technology, Artificial Intelligence Laboratory*, 1979.

[9] Bryan Williams. A commonsense approach to story understanding. Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2017.

[10] Patrick Henry Winston. The genesis story understanding and story telling system: a 21st century step toward artificial intelligence. *CBMM*, 2014.