

# Ambiguity Resolution by a Virtual Agent Through Its Own World Knowledge

## **Dissertation**

der Mathematisch-Naturwissenschaftlichen Fakultät  
der Eberhard Karls Universität Tübingen  
zur Erlangung des Grades eines  
Doktors der Naturwissenschaften  
(Dr. rer. nat.)

vorgelegt von  
Christian Julius Stegemann-Philipps  
aus Hildesheim

Tübingen  
2020



Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen  
Fakultät der Eberhard Karls Universität Tübingen

Tag der mündlichen Qualifikation:	18.02.2021
Stellvertretender Dekan:	Prof. Dr. József Fortágh
1. Berichtsersteller:	Prof. Dr. Martin V. Butz
2. Berichtserstellerin:	Prof. Dr. Susanne Winkler





*I dedicate this work to my wife Annika, because I can't imagine doing a PhD without her;  
and to those undertaking projects while not knowing the way.*



## **Acknowledgements**

I was lucky to work with wonderful people during my PhD, I always profited from their professional advice, and they have contributed a lot to the positive aspects of this work. Whatever faults may be found with this work, however, are of my own making. I am thankful to Martin V. Butz and Susanne Winkler for their supervision and guidance, to Asya Achimova for all her help and advice and the fun we had, to Dilectiss Liu for stimulating discussions and wonderful guitar play, to my office colleagues from 048, especially Maren Ebert-Rohleder, Elisabeth Schedel, and Anna Schwetz, for their companionship, to the GRK 1808 Ambiguität and all its members for the wonderful setting and their help, especially Inken Armbrust for our shared breaks and Natascha Elxnath and Sarah Metzger for their comments on the linguistic glossary, to the Neuro-Cognitive Modeling Group for their valuable input and the fun we had, to Greg Scontras for sharing his experience and providing guidance, and to all the researchers I have met along the way that listened and gave their advice. I also want to thank my wife and my family for their support, I can only spend at work what I gain at home, and I am happy and lucky to have you.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Understanding Natural Language Is Difficult</b>	<b>3</b>
2.1	Winograd Schemas . . . . .	4
2.2	Basic Definitions for Modeling . . . . .	6
2.3	Deep learning approaches . . . . .	8
2.4	Non Deep Learning Approaches . . . . .	9
2.5	Transparency, Scalability and Cognitive Plausibility . . . . .	10
<b>3</b>	<b>Processing Language and Ambiguity</b>	<b>13</b>
3.1	Meaning and Ambiguity of Isolated Sentences . . . . .	13
3.2	The Problem of Ambiguity . . . . .	15
3.3	Using Context in Comprehension . . . . .	16
3.4	Types of Ambiguity and World Knowledge . . . . .	18
3.5	Information Gaps and the Utility of Ambiguity . . . . .	21
3.6	Modeling Language Processing . . . . .	23
<b>4</b>	<b>Meaning and World Knowledge</b>	<b>25</b>
4.1	Inference and Mental Simulation . . . . .	27
4.2	Embodiment and Grounding . . . . .	30
4.3	Truth Conditions and Scene Reconstructions . . . . .	32
<b>5</b>	<b>Modeling World Knowledge</b>	<b>33</b>
5.1	Predictive Processing Accounts . . . . .	33
5.2	Pre-Defined Structure and Event-Predictive Cognition . . . . .	35
5.3	Emerging Structure . . . . .	36
<b>6</b>	<b>The Model and Main Hypotheses</b>	<b>39</b>
6.1	Main Hypotheses . . . . .	40
<b>7</b>	<b>The Event-Predictive System</b>	<b>42</b>
7.1	Learning and Applying Encodings . . . . .	42
7.2	A Computational Description of the Event Predictive System . . . . .	45
7.3	The Structure of Events . . . . .	46
7.4	Related Models of Event-Dynamics . . . . .	47
<b>8</b>	<b>Implementation of the Event-Predictive System</b>	<b>50</b>
8.1	Sensory Information Available to the System . . . . .	50
8.2	Models, Predictions and Errors . . . . .	52
8.3	Model Updates and Surprise . . . . .	53
8.4	Transition Models . . . . .	54

8.5	Relation Models . . . . .	58
8.6	Summary of the Implementation . . . . .	58
8.7	Implementational Details . . . . .	58
<b>9</b>	<b>Evaluation and Discussion of the Implementation</b>	<b>62</b>
9.1	Overall Error Minimization . . . . .	64
9.2	Learned Models . . . . .	66
9.3	Overcoming Limitations . . . . .	73
9.4	Extending Spatial Relational Encodings . . . . .	75
<b>10</b>	<b>The Language Processing System</b>	<b>78</b>
10.1	Constructing Abstract Scene Representations . . . . .	78
10.2	The Lexicon Function . . . . .	80
10.3	Processing a Sentence . . . . .	81
<b>11</b>	<b>The Inference and Simulation System</b>	<b>85</b>
11.1	Reconstructing a Scene Is Costly . . . . .	86
11.2	The Inference Mechanism . . . . .	89
11.3	Implementation of Simulation . . . . .	90
11.4	Comparing a Simulated Scene to the Requirements . . . . .	93
11.5	Adaptation Heuristics . . . . .	95
11.6	Summary . . . . .	96
<b>12</b>	<b>Evaluation and Discussion of the Implementation</b>	<b>97</b>
12.1	Limitations . . . . .	116
12.2	Resolving Ambiguity . . . . .	117
12.3	Language and Event-Predictive Structures . . . . .	118
<b>13</b>	<b>Experiment: Producing Information Gaps</b>	<b>120</b>
13.1	Language Production and Predictability . . . . .	120
13.2	The Experiment . . . . .	123
13.3	Experimental Setup . . . . .	124
13.4	Participants . . . . .	126
13.5	Annotating Utterances and Analysis . . . . .	126
13.6	Results and Discussion . . . . .	127
13.7	General Discussion . . . . .	129
<b>14</b>	<b>Conclusion</b>	<b>133</b>
	<b>Appendix A Reference to Implemented BrainControl Code</b>	<b>148</b>
	<b>Appendix B Vocabulary in final evaluation</b>	<b>148</b>

<b>Appendix C</b>	<b>Details of Production Experiment</b>	<b>150</b>
<b>Appendix D</b>	<b>Models Learned By Final Implemented System</b>	<b>150</b>





# 1 Introduction

<sup>1</sup>It has been a long-standing goal of cognitive science to develop computer programs that are able to understand natural language, e.g. English, rather than formal programming languages only. There has been considerable advancement over the last decades, however, computer programs are still not communicating with us in plain English as of today. While there are examples where it seems that the machine can understand us, like modern digital assistance software built into smartphones, this comprehension often comes to a sudden and unexpected halt. One of the main problems seems to be that a computer program does not understand our *world*, and thus has limited understanding of our *language*.

Levesque et al. (2012) propose the *Winograd Schema Challenge* to test how well computers actually comprehend language. The basic idea of the challenge is to provide the computer with an *ambiguous* sentence that could be interpreted in two distinct ways. The program then has to decide which interpretation is correct. If the challenge items are well designed, the computer will be required to have an understanding of our world to actually solve the task. Importantly, a human reader will be able to interpret the challenge items in the correct way without much effort, maybe even without consciously thinking about the other interpretation. If humans can understand the challenge items because they have an understanding of how the world works and computer programs are limited in this regard, then this challenge should test the deeper capabilities of computer programs to understand language.

The phenomenon, that a human reader does not notice the ambiguity, but the computer completely misinterprets the utterance, can be linked to a general issue with ambiguity. In everyday conversation, there is much more ambiguity than usually noticed. This is often attributed to humans being able to resolve ambiguity by using context and world knowledge. An important question is thus how ambiguity interacts with world knowledge. We propose that an ambiguity arises when the utterance is missing bits of information that are needed to decide on one of the two possible interpretations. We call this phenomenon an *information gap*. While missing information can lead to ambiguity, it is also efficient to drop information if it is not necessary, creating the need to find an acceptable balance. We show that speakers in our experiment actively manage the amount of information depending on their knowledge of the world.

The larger part of this work focuses on developing LEARNA, a computational model that addresses the Winograd Schema Challenge, but which is also able to resolve ambiguity through world knowledge in general. The big question then is how an understanding of the world is acquired, and how it can be used to understand natural language. The computational model proposed and implemented in this work solves

---

<sup>1</sup>Chapter 13 of this work has been reworked and published in Stegemann-Philipps, Butz et al., 2021, Chapters 1-12 have been reworked and published in Stegemann-Philipps and Butz, 2021

the Winograd Schema Challenge in a simplified and restricted environment. The model acquires an understanding of its world by building upon work on cognitive processing of *events*. As will be shown, the structures emerging from this approach are efficient at acquiring world knowledge but also at being used for language comprehension.

This work is structured in a way so that a broad theoretical overview of the different fields is provided first before describing the LEARNA system. Section 2 will start with a general introduction of Winograd Schemas and the problem of understanding natural language. Basic definitions used for modeling throughout this work will be introduced and a review of current modeling approaches to Winograd Schemas will be done. Finally, abstract requirements, that LEARNA should satisfy, will be derived. Section 3 provides an overview of the linguistic theory that motivates this work and is used throughout. A short but general description of language processing in general and ambiguity in particular will be provided. After reviewing more specific linguistic approaches to ambiguity, the notion of information gaps will be introduced, which will motivate one of the main hypotheses of this work.

Section 4 examines theoretical approaches to world knowledge and their role in the meaning of utterances. Mental models will be introduced here, and embodiment and grounding will be discussed briefly. The problem of grounding partly motivates the modeling attempts of this work. Section 5 develops theoretical notions on how to model the acquisition and representation of world knowledge. Event-predictive cognition in particular will be introduced. After this broad theoretical overview, section 6 presents the abstract nature of LEARNA as well as the main hypotheses that this work provides evidence for.

Section 7 switches to the modeling perspective and describes the event-predictive system, one of three main parts of LEARNA, from a computational or functional perspective. Section 8 then describes the implementation of this system. The system will then be evaluated in 9.

As a counterpart to the event-predictive system, section 10 introduces the language processing system, the second of three main parts of LEARNA. A functional description, as well as a detailed description of the chosen implementation, is provided. Section 11 then describes the last of the three main parts of LEARNA: the inference and simulation system. This system will provide a link between event-prediction and language processing. LEARNA will then finally be evaluated in section 12, specifically the range of its language comprehension ability.

Section 13 represents an excursion from the modeling and describes an experiment investigating the influence that world knowledge has on human language production. Section 14 will provide a general conclusion.

## 2 Understanding Natural Language Is Difficult

The main goal of this work is to build an artificial system that understands language to a certain degree. In this work, we define an artificial system as a computer program that exhibits specific interactive behavior. One of the early systems that approached this issue was called *SHRDLU*. Winograd (1972) described the artificial system SHRDLU in a long article titled *Understanding Natural Language*. SHRDLU was able to process natural language expressions to manipulate a simple artificial world. Such simplified artificial environments in which an artificial system is operating will be called *mini worlds* from here on. A mini world is mainly characterized by the space it defines in which objects are simulated. The mini world is often picking out specific objects and behaviors of the real world and simulates those in a simplified manner. In the case of SHRDLU, the mini world consisted of differently shaped objects in a box. The box functions as the general boundary of the mini world, whereas the objects populate it and follow the laws of physics. The behavior of SHRDLU seems quite impressive, as the following short extract from a sample dialogue shows.

'10. can a pyramid be supported by a block?

YES.

The deductive system finds an actual example, so it knows this is possible.

11. can a pyramid support a pyramid?

I DON'T KNOW.

The question makes sense, but there is no example in the scene, so there is no easy way to tell without being able to examine the robot's programs.

12. stack up two pyramids.

I CAN'T.

The robot tries and fails.' (Winograd, 1972, p. 10)

While these results are impressive, SHRDLU is unfortunately fully hard coded, i.e. the behavior it exhibits is fully defined by the original programmer. Adjusting it to encompass additional capabilities is thus very difficult. Even building the artificial system itself is already very time consuming. Given that the mini world, in which SHRDLU operates, is a box populated by pyramids and cubes, the problem of understanding natural language is not actually solved. All the behavior is manually created for this specific mini world, however, it is difficult to *scale up* to more advanced capabilities. This means that SHRDLU is not actually applicable to the general problem it approaches. It was never used in actual applications. In this work, we will try to avoid *hard coding* as much as possible. SHRDLU does serve as a good starting point however, as it comes close to LEARNA in its goal of modeling the process of understanding natural language.

## 2.1 Winograd Schemas

Winograd acknowledged in his work that there are cases in which natural language seems clear to human listeners but very difficult to understand for computers due to the necessity to have a huge amount of knowledge about the world. Winograd proposed (1) as an example of this that would actually lead to difficulties, for instance, in machine translation. To give credit where credit is due, Bar-Hillel (1960) already discussed this problem using example (2).

- (1) a. The city councilmen refused the demonstrators a permit because they feared violence.
- b. The city councilmen refused the demonstrators a permit because they advocated revolution.<sup>2</sup>
  
- (2) a. Little John was looking for his toy box. Finally he found it. The box was in the pen. John was very happy.<sup>3</sup>

Winograd (1972) observed that *they* changes reference in (1), but to be able to assign the correct reference, an artificial system would need to have broad knowledge of the world. Bar-Hillel (1960) argued that *pen* was ambiguous in (2) but to choose *small enclosure for children to play in* over *pencil*, the artificial system would again need to have broad world knowledge.<sup>4</sup> In (1) specifically, a system would need to have a concept of city councils and their members, of demonstrators, of violence, revolution, fear, advocating things, permits and refusals. In addition the system would need to know about how these different things usually work together, e.g. causality. Such knowledge may be called *world knowledge*, the term will be refined over the span of this work.

Example (1) inspired Levesque et al. (2012) to propose *Winograd Schemas (WSs)* as a class of natural language understanding problems that are simple for humans but difficult for computers. Levesque et al. (2012) propose to use pairs of such sentences to test how much world knowledge an artificial system has acquired. Their reasoning is that an artificial system, that is able to assign correct reference to *they* in (1), will also exhibit a substantial amount of world knowledge as well as the ability to do *common sense reasoning*. In this work, we define common sense reasoning as the application of world knowledge to actual problems. It is important to keep *having* world knowledge and *applying* it apart, especially in this work.

---

<sup>2</sup>both (Winograd, 1972, p. 33)

<sup>3</sup>(Bar-Hillel, 1960, Appendix III)

<sup>4</sup>It is interesting that Bar-Hillel claims that ‘no existing or imaginable program will enable an electronic computer to determine that the word pen in the given sentence within the given context has the second of the above meanings, whereas every reader with a sufficient knowledge of English will do this “automatically”.’ At the time of the writing of this work, the translation engine deepl.com actually produced a somewhat correct translation into German by translating *pen* as *Pferch*, which is not perfect but very close to the correct *Laufstall*. When introducing gender into (1), the translation into German was incorrect however. This was done with a simple one time test by the author.

Levesque et al. (2012) collect pairs of sentences into a collection they call the *Winograd Schema Challenge*, see Davis (2019) for an updated collection. Each of those pairs is constructed similar to the pairing in (1), though they introduce a few more restrictions, like ‘the sentences should only differ in one or two words’, to make the data more uniform. Levesque et al. (2012) also try to avoid sentences with simple shortcuts, so they do not allow sentences where simple co-occurrence patterns would already yield the correct interpretation. An example of this would be (3), where *women* and *pregnant* will co-occur in a corpus much more often than *pills* and *pregnant*. Such a co-occurrence pattern is simple to find in a large enough corpus, i.e. a collection of written texts in this case. But finding co-occurrences can be done by a simple program and does not allow more general common sense reasoning.

- (3) a. The women stopped taking the pills because they were pregnant.  
b. The women stopped taking the pills because they were carcinogenic.<sup>5</sup>

To strengthen the case of WSs, let us consider a mini world in which certain regularities are different from the real world. When a person explores the mini world and tries to describe it, they will probably start by using basic categories. If there is an object shaped like a box, they may say: ‘there is a box’. If the object now starts to move around using bipedal motion, they may say: ‘the box walks around’. Something like a WS can be produced if the person says:

- (4) ‘The box falls to the ground and it breaks.’<sup>6</sup>

What *it* refers to in this case remains ambiguous unless one is familiar with this specific mini world.<sup>7</sup> Both, the object shaped like a box and what is seen as the ground, might be programmed to break apart in this case. What is necessary to understand the sentence is *world knowledge*, only this time the world is an artificial one. The rules and regularities of such a mini world may plausibly be turned into WS style sentences, and the knowledge an artificial system has of these regularities can be tested this way. The same reasoning plausibly applies for the real world too. A regularity of the real world that most people are familiar with can be turned into a WS style sentence. Such a sentence might not actually qualify for the Winograd Schema Challenge, e.g. because there is a statistical shortcut, but it is similar in spirit. If this is true, WSs provide a way of probing an artificial system’s knowledge of the world that is very simple for humans to understand.<sup>8</sup>

The overall goal of this work is an artificial system that approaches understanding natural language in a way that allows it to resolve referential ambiguities such as

---

<sup>5</sup>both (Levesque et al., 2012, p. 555)

<sup>6</sup>What would be missing here is a partner sentence.

<sup>7</sup>Of course there is a strong intuition that the box breaks apart here, but programming the mini world so that the ground breaks apart is just as simple as programming it so that the box breaks. This is why a mini world really brings out how WSs rely on knowing how the world behaves.

<sup>8</sup>Coming up with a sentence is often easier than programming a testing routine

WSs. This means the artificial system should acquire and use world knowledge to understand them. Since this task in its generality is extremely complex, we will work towards a manageable problem and an achievable modeling approach, similar in spirit to the system that Winograd (1972) describes while avoiding its shortcomings. Before reviewing current models however, the basic modeling terminology that will be used throughout this work will be introduced.

## 2.2 Basic Definitions for Modeling

Above we already defined artificial system and mini world. As a first restriction, an artificial system in this work is always an *information processing* artificial system, i.e. it is seen as a mechanistic or computational function that maps *input* information to *output* information. In this work, we will disregard the philosophical discussion surrounding these terms and focus on the fact that such an artificial system can be described using formulas. In describing the computations done by information processing artificial systems, it is common to distinguish Marr's three levels of description, see Marr (2010), Rescorla (2020). They can be seen as a hierarchy of descriptions, with the first or top level being the *computational level of description*. This level addresses the questions of what the computation is supposed to achieve and why it is done, i.e. the abstract *function* of the computation and its notable features. The second or middle level is the algorithmic level of description, sometimes called representational level. On this level, the algorithmic nature of the computation is described, i.e. the formalism defining the computation. This specifically includes the representational structures that the computation operates on. The third or bottom level is the implementation level of description, on which the actual physical realization is described. In this work, artificial systems are only described on the first two levels; the physical realization is not important here. Note that we will describe the *implementation* of our artificial system later. But this will actually be on the algorithmic level of description since the exact nature of input and output as well as the mapping functions will be described. Since the artificial system in this work will be complex, it will happen that these levels become blurred. The overall artificial system will be differentiated into parts. This could already be seen as a description on the algorithmic level, even if the parts are only described on the computational level.

Now let us assume we have a description of an artificial system on the algorithmic level. We can say that the input, the output and the internal *states* of the artificial system are *representations*, i.e. informational states that the system operates on and which ideally stand in for a state of the domain in which the system operates. In the case of our system, this will be a mini world. A useful artificial system will usually describe specific dynamics of the domain it operates on. In the case of a mini world these dynamics may be how objects behave or the physical laws that govern the mini world. This we call a model. A more complex artificial system can also contain

many different models, describing different dynamics. We further say that a model *encodes* information about the dynamics it describes. Since an artificial system can also encode information without using models, we define an encoding as information that a system has access to internally. Thus, a model is an encoding, but an encoding is not necessarily a model. Finally, those parts within an artificial system that are not models will be called mechanisms here. Mechanisms operate on models and encodings.

To illustrate these terms on a well known example, consider a small scale digital simulation of our solar system and its internal movement patterns. As a whole, this is a model, because it describes the movement dynamics of the solar system. As an artificial system, it may contain different parts. The current configuration, i.e. the current position of all the objects, is a representation, and there may be distinct models for each planet, describing their movement. Each of these will thus operate on the appropriate planet representation. It is also imaginable that there is only one model describing the movement patterns in general which depends on features of a planet like size or weight. The artificial system may have encodings of these features, e.g. as a list. To predict the movement of one planet representation, the artificial system now needs a mechanism to pick the correct model as well as the necessary encodings.

Recall that SHRDLU, the artificial system described by Winograd (1972), was hard coded. In the solar system example, the most hard coded way of building such a simulation would be to simply encode the trajectories as curves and velocities into the artificial system. It would then be able to simulate the solar system, i.e. it would be a fully functioning model. If, however, we wanted to add in the moon, we would have to implement the moon's trajectory and velocity too, and the same would apply for every new object we ever inserted into the model. In the most general, i.e. least hard coded, way to build such a system, we would encode the laws of physics in a model and let the system compute velocities and trajectories. To add in a new object, we would now only have to encode the necessary features like weight, size and starting position. Note that in this latter case, we would not even need to know the full trajectory of an object because the model would provide it.

An even simpler way of encoding information in an artificial system is to let it *learn*. This is usually done through learning mechanisms that are fed data. If the mechanisms are good, the result will be a model of patterns that the data follows. This process of learning on data is called training. Because systems that are trained can sometimes behave in unforeseeable ways, they are usually *tested* on cases where the ideal behavior is known in order to see how well they perform on these test cases. If there are many similar test cases, a useful measure of a system's performance is *accuracy*, i.e. how many test cases the system solved correctly.

With all of these definitions in place, we can review current approaches to the Winograd Schema Challenge and describe the approach that we will take in this work.

## 2.3 Deep learning approaches

Except for marked exceptions, recent models are usually tested on the collection of WSs in (Davis, 2019). Since the data set only contains WSs that a model can either resolve correctly or fail, models will usually be evaluated by accuracy, i.e. how many examples the model resolved correctly.

The most successful approaches in terms of accuracy on (Davis, 2019) build on the recent advances in training very deep neural network models. Artificial neural networks mimic real world neural networks, meaning that they are networks consisting of nodes, where a node is inspired by a real world neuron. In the network structure, the connections between the nodes have weights that restrict signal flow in the network, and the nodes have functions that determine how the node passes on input signals. These weights can be *trained* on data, which, if done correctly, leads to the network as a whole approximating patterns found within the input data.<sup>9</sup> If the complexity of the patterns that are to be learned increases, an artificial neural network will need a lot of data with correct labels to be able to fine-tune its parameters. One advantage of neural network models is their generality, see e.g. Russell et al. (2010), but this comes at the cost of needing a lot of computing power and data to be trained. Given that the models consist of a large number of interconnected nodes, it is often very difficult to interpret what they *do*. The trained models can become a blackbox, thus being helpful as applied models but uninformative in regards to the patterns they discover.<sup>10</sup>

Recent advances in computing power, data availability and algorithms led to surprisingly strong models in a wide range of fields. Natural language processing is one of those fields, with new results coming in in quick succession. Kocijan et al. (2019), for example, make use of recent advances in pre-training to generate models that reach an accuracy of 71.4% on the collection of WSs by (Davis, 2019). In simple terms, a model is trained on an incredibly large general purpose corpus of written text and then fine-tuned on very specific sets of other WSs, i.e. the model is never actually trained on the WSs in (Davis, 2019). Sakaguchi et al. (2019) further improve this to 90.1% by using a much larger data set for fine-tuning. Given the larger accuracy of their model, they propose to instead use a larger data set which they simultaneously introduce as a new reference test-case. They call this data set Wino-Grande. Since this larger data set is designed to be more difficult than the original collection in (Davis, 2019), accuracy of the models is worse. Sakaguchi et al. (2019) report 79.1% accuracy on Wino-Grande, Lin et al. (2020) even report 84.6% on Wino-Grande.

Even more recent, Brown et al. (2020) describe a model that is once again much larger, in terms of parameters (weights that are trained) and in terms of training data,

---

<sup>9</sup>Since artificial neural networks are not used further in this work, the description remains limited. See e.g. Russell et al. (2010) for an in depth introduction.

<sup>10</sup>It is generally assumed that neural network models are difficult to interpret. In how far this is a problem and how good this can be solved is a controversial issue into which we will not go here.



than preceding models. They report around 80%<sup>11</sup>, which is worse than the models above, but their model is not fine tuned at all. This means the model is only trained on written text found on the internet and then tested on the WS data set. At the time this dissertation was written, the model by Brown et al. (2020) was among the most recent.

While these very recent results are impressive, interpreting them is not as easy as it seems. The original idea of WSs was to create a problem that would require common sense reasoning to solve. It is not at all clear that the models presented above are indeed able to use common sense reasoning<sup>12</sup>. These results are often achieved by using a very closely related data set to fine-tune the models, raising the question of whether their might be redundancy in the data to some extent. Lin et al. (2020) are indeed careful in interpreting the implications of the power of their model. They conclude that the results could either be explained by biases in the data which the models exploit or by an actual progress towards something like common sense reasoning. The latter may also be because the common sense knowledge simply appears in the data. As they put it:

Perhaps it is the case that in a humongous corpus of natural language text, someone really has written about trying to stuff a tuba in a backpack? (Lin et al., 2020, p. 3)

Whether such models get closer to common sense reasoning is a huge topic of its own. For now, we can conclude that in terms of the WSs, there are now models that achieve impressive accuracy on the original collection in (Davis, 2019). However, as of now, we are unable to tell what patterns the models discover and use, making them good for application but not very useful for understanding underlying questions about acquiring and processing common sense reasoning.

## 2.4 Non Deep Learning Approaches

Other approaches to WSs have proposed to insert more pre-defined structure into the model. Peng et al. (2015) introduce *predicate schemas* that capture subject-predicate-object relationships found in the real world. A basic example they give is a schema  $pred_m(m, a)$  that could be instantiated as *have*( $m = flower, a = pollen$ ). They then translate the interpretations of WSs into instances of such predicate schemas and score their probability. Their example would be ‘the bee landed on the flower because it had pollen’, where their model ideally would score the flower having pollen higher than the bee having pollen. They use different methods to learn scoring such predicate schemas, for example they extract instances from a huge corpus, of text or they use a web search and count the returned results. All in all, this method is based on the same

---

<sup>11</sup>They report slightly more than 80% but then discuss a mistake in training, decreasing their performance to slightly less than 80% on a filtered version of the collection in (Davis, 2019).

<sup>12</sup>It is not clear that they are not able to do so either

resources as the neural network models in section 2.3, i.e. a huge corpus of raw text, but it applies an intermediary step to make it easier to assign probabilities.

Schüller (2014) proposes translating WSs into graphs, a representation consisting of entities and relations between them. The solution is then computed by analyzing these graph representations. The drawback of their approach is that they have to manually build the graph structure, translating each WS and the appropriate world knowledge into their representational format. This is impossible to scale efficiently to larger data sets.

Sharma et al. (2015) propose a formalization of WSs in terms of two specific structural types they identify. They then use the formalization to search the web<sup>13</sup> for examples containing the necessary world knowledge. The drawback of this approach is the need to search a large text corpus *after* getting the WS as input. The world knowledge they extract will thus be very redundant in the long run because for each new example, the relevant knowledge is searched anew within the corpus.

Liu et al. (2017) try to learn embeddings with the help of databases that store dependencies between words or concepts. An embedding is a function that assigns vectors to input units, such as words or sentences, so that the vectors capture some of the structure of the input. Using the data within the data bases, they compute similarities between the embedding of the pronoun in the WS and its possible referents. While their model indeed scales to the original small set of WSs in Davis (2019), they report a rather low accuracy of 52.8% on a similar data set which was provided to participants of a formally held Winograd Schema Challenge, see Levesque et al. (2012).

What unifies the approaches presented above is that they introduce handcrafted structures to encode world knowledge and then populate these structures, i.e. learn world knowledge, from data. The basic sources to learn from are either huge corpora of written text or hand crafted sources, e.g. existing data bases or sources created specifically for the task. While hand crafting works well, it is very time consuming, and scaling is extremely difficult, something that was already discussed for the system described by Winograd (1972). Ideally, a model should be able to function without extensive hand crafting, i.e. the need for hand crafting should not increase when scaling the model to larger domains. In terms of raw accuracy on the collection in (Davis, 2019), the models were overpowered by sheer corpus size and computational power as described in 2.3. It is unclear whether this is because of the inability to scale or the proposed structures being faulty.

## 2.5 Transparency, Scalability and Cognitive Plausibility

In the following, we will derive what path this work will take in modeling. So far, we have seen that hand crafting or hard coding are problematic because the need for introducing structure is difficult to satisfy when scaling. We also saw that deep

---

<sup>13</sup>The web can basically be considered a text corpus.

learning models, while achieving high accuracy when tested on data sets, are obscure as to their inner workings, not revealing learned patterns.

At their core, WSs are about understanding the world. The general assumption is that understanding WSs requires world knowledge, so the models reviewed so far all extract and apply world knowledge to solve WSs. This can even be assumed for the obscure neural network models where it is unclear how the model works internally, see e.g. the discussion on the presence or absence of common sense in these models by Lin et al. (2020). The efforts to generate these models are part of a broader attempt to produce models with deeper language understanding that are not confined to a specific task. Ideally, a model that can solve WSs has an understanding of language and is also applicable to related tasks. Reversely, models that succeed at those other tasks should be able to solve WSs. The work on such deeper understanding has a long history and there are many more contributions than the WS models that are mentioned here. Storks et al. (2019) provide a very recent review of these modeling attempts. Judging by their review, it seems that the models above provide appropriate insight into the range of approaches used in general: neural networks seem to be the most active approach, but there are also other statistical models as well as handcrafted reasoning models. Storks et al. (2019) even identify the same data sources as above: written text and established data bases. Even so, the most accurate models discard established data bases and only rely on written text before fine-tuning on specialized selections of winograd schemas or similar sources.

Focusing on test set accuracy, the models described above do not pursue similarity to human cognition. This also means that those models interact very little with research on human cognition. To bridge the gap between models and research on cognition, this work chooses a different approach. We will propose a model that is inspired by theories of cognition and thus more similar to humans cognition. This has three main advantages. We may uncover structural features that can be used to improve models which optimize for accuracy. It also allows to test theoretical models of parts of cognition. If we are able to implement a model of cognition and the model is successful, this makes the theoretical model more plausible. It will not prove that the theoretical model is right, but it proves that the model is feasible. Finally, an implementation will force every little detail either to be spelled out or to be part of the assumptions, thus uncovering weak spots of theoretical models and opening ways for refinement. Such a *plausibility check* through implementation is all the more useful in cases where details of cognition are hard to prove through data. This is often the case for underlying mechanisms of cognition, where only behavior and brain-related data is accessible.

Because cognition is so complex, models are usually implemented in a reduced fashion, picking out specific cases or relying on certain assumptions. Such partial implementations strengthen the theoretical models if successful; they may provide useful structural insights for further models and demonstrate the need for refinement where problems emerge. For models that implement theoretical models of cognition

and their structure, the main advantages hold even the implementation is restricted or reduced to only implement parts.

As a first summary, we can note requirements that the model in this work should satisfy, as described below.

1. Hard coding should remain minimal. In the case it does occur, it should not require more hard coding when scaling the model, i.e. hand crafting parts of the system should only be done once.
2. The model should be inspired by and mirror plausible theoretical explanations of human cognition. It suffices if this is restricted to the reduced domain that we focus on, namely ambiguity resolution through world knowledge as exemplified by WSs.
3. The structure of the model should be transparent and learned patterns should be accessible.

To resolve all possible kinds of WSs, a full model would need to encompass a complete language faculty as well as world knowledge and the ability to process such knowledge. This is an extremely challenging task, which was the original point made by Bar-Hillel (1960) and that Winograd (1972) reiterated. Such a full model is far beyond the scope of this work, but a partial implementation is feasible. As discussed above, the model should mirror theories of human cognition. So, the next sections will introduce the theoretical scaffolding that will be needed to build the model, starting with language processing.

### 3 Processing Language and Ambiguity

This section will develop the basic theoretical background that will inspire our model for it to be able to resolve the reference in examples like (4). Besides an overview of how language can be interpreted in general, i.e. the basic structures that the model should implement, a more general discussion of ambiguity and its resolution will be included. As before, we will start with basic definitions.

In this work, we restrict the definition of communication to a speaker sending an utterance to a listener. Although we say speaker and listener, it does necessitate spoken communication. Our model will process written sentences. Before the listener can retrieve the meaning of a sentence, it first has to enter the mind, e.g. through reading or hearing. Thus, the first system that is activated in comprehension is an auditory or other perceptual faculty. This system will provide an accessible representation (instead of sound waves). In this work, we will disregard this first step and assume in our model that the input is already accessible to the underlying systems.

We can say that the speaker *produces* the utterance and the listener *comprehends* it. We will assume that both the speaker and the listener are *cooperative*, i.e. both use the resources they have available to be understood and to understand. A traditional approach to define *understanding* is to say that a listener understands an utterance if they know the conditions under which it is true, see e.g. Löbner (2012). Knowing these *truth conditions* is equal to knowing which states of the real world are denoted by the utterance. We will stick with this approach in this work, but the details of how to model this will be worked out later.

Finally, we restrict ourselves to utterances that describe real world situations, so we ignore questions or implicit meanings, for example. This also means that the meaning of an utterance is always a specific situation.

#### 3.1 Meaning and Ambiguity of Isolated Sentences

When a listener is presented with a simple isolated sentence like (4), there are traditionally three basic things they will have to do to ascertain the meaning of the sentence. They will need to assign a meaning to each of the words that make up the sentence, i.e. they will have to know lexical semantics. In this case, this means they will have to know what a box is, what the ground is, what falling is and what 'to' means. We can assume that this meaning is stored in memory and activated when necessary. This meaning storage is often called the *mental lexicon*, although this is a rather theoretical construct, see e.g. Löbner (2012). Without diving too deep into the debate about the mental lexicon, we will assume that the meaning of non-compositional concepts are stored in memory. The basic unit of lexical meaning will be regarded as *words* or *lexemes* in this work.

The listener will also need to work out the grammatical structure of the sentence, i.e. they will have to know syntax and morphology. In this work, we ignore morphology,

e.g. the *s* in *breaks*, since it is too removed from our modeling focus. Syntactic analysis of (4) would tell the listener that there are two parts within the sentence, that *falls* has the box as its subject and that it is extended by the prepositional phrase *to the ground*. Further, *Breaks* has a pronoun as the subject which could refer to either the box or the ground. In terms of human language processing, there is evidence that syntactic processing is somewhat distinguishable from other steps, see e.g. Ferreira (2019) for production or Vonk et al. (2018) for comprehension. This does not mean that syntax is necessarily processed in isolation, but there is some evidence that there is a special mechanism for putting elements together.

After analyzing word meaning and syntactic structure, the listener will have to use both and assign meaning to groups of words until ending up with the full sentence with meaning, i.e. they will have to know compositional semantics. This means they will form the meaning of *the box falls* and *to the ground*, then of *the box falls to the ground* and finally of the complete sentence. From a processing perspective, this step is less clear and we will develop details of this process over the following sections.

(4) The box falls to the ground and it breaks.

It has to be noted that these theoretical distinctions are subject to grand controversies, especially in regards to if and how the different types of analysis are intertwined. Jackendoff (2002) discusses this issue at length and says at one point that ‘it is a major research problem, debated for the past forty years, to determine how much of meaning is directly signaled by syntax.’<sup>14</sup>

Note that in the case of (4), the listener will have to assign a reference to the pronoun *it*. This is exactly the main problem fueling this entire work, that it is impossible through syntactic and semantic analysis alone to assign a reference to the pronoun. But first, it should be made clear that this is actually the case. If a reference would be assigned to *it* in terms of strictly syntactical analysis that does not depend on word meanings, this assignment would on average be wrong for about every second WS. Recall that WSs are designed so that switching one word or a small group of words, e.g. a noun phrase or a prepositional phrase, changes the reference of the pronoun. In (4), we can imagine a mini world where the ground *shakes* when something hits it. In this case, we would get a WS as in (5), where both sentences have the same syntactic analysis but the referent of *it* is different.

(5) a. The box falls to the ground and it breaks.  
b. The box falls to the ground and it shakes.

(6) I took the tomato off the shelf and it was ripe.

It could be the case that Semantic Knowledge can be used to assign reference to *it*, as in the case of (6), where the reasoning would be that only tomato can be combined

---

<sup>14</sup>Jackendoff (2002), p. 270

with ripe. The reference would be determined through semantic composition rules. This reasoning does not apply to (5), however, because it seems that in general, boxes and ground can shake or break alike. By knowing the specific mini world, we can come to *know* that the box breaks, but it is not part of the rules of compositional semantics. This issue of differentiating world knowledge and semantic knowledge is also rather controversial, see e.g. Löbner (2012). Note that in the original example (1), both demonstrators and councilmen can in general fear violence or advocate revolution, no combination is semantically wrong. This means that semantic knowledge may solve the problem of assigning reference to the pronoun in WSs sometimes, but WSs should resist this solution by design to a certain extent. We thus ignore this aspect of semantic knowledge in this work.

If the listener can not assign reference to the pronoun, the meaning of the overall sentence remains obscure if we assume that there is an important difference between the box or the ground breaking. This leads us to the definition of ambiguity as we will use it in this work:

**Ambiguity** We say that a sentence or an utterance is *ambiguous* if at least two distinct meanings can be assigned to it.

This definition thus allows us to distinguish ambiguity from underspecification, which we use when an utterance does not specify a detail, i.e. something is left open to interpretation. This distinction is important because using these definitions, ‘the box breaks’ is not ambiguous but may be under-specified. This will resurface in later discussion, for now it is important that ambiguity denotes several *distinct* available meanings.

### 3.2 The Problem of Ambiguity

Ambiguity has often been seen as a problem that should be avoided because it leads to misunderstanding. If an utterance is ambiguous, a listener might choose the wrong meaning and might not understand correctly. Communication would have failed. Since we assume that the speaker wants to be understood, a more precise utterance should be preferred over an ambiguous one from this perspective. Trying to build a perfect logical language, Frege (1879) banned any form of assigning multiple meanings to a sign. Much later, Chomsky (2002) still found that ambiguity makes communication more difficult. Grice (1967) prescribes that a speaker should try to avoid ambiguity in conversation as one of his maxims. This is especially apparent in formal language, Sennet (2016) remarks that ‘we use formal languages precisely so that we can disambiguate otherwise ambiguous sentences’.

While problems of misunderstanding should be obvious, there are also famous examples of how ambiguity simply makes an utterance difficult to understand. This happens with garden path sentences like (7).

- (7) The horse raced past the barn fell.<sup>15</sup>

*Raced* by itself could be analyzed as intransitive or transitive, i.e. as *the horse is racing* or *the horse is being raced by someone*. If *raced* is read intransitive, *fell* does not fit into the sentence, producing the garden path effect. The *problem* seems to be that people parse the sentence in an incremental fashion, reading *raced* intransitive first, not recognizing the ambiguity. This way they are then surprised by *fell*. This is a case of *temporary ambiguity*, because *raced* is only ambiguous temporarily if the sentence is parsed incrementally from left to right. In the full sentence, there is only one grammatically correct interpretation, i.e. there is no open ambiguity after all.

From this perspective, ambiguity can be seen as undesirable because it makes communication difficult or fail.

### 3.3 Using Context in Comprehension

If (4) is not presented in isolation but occurs embedded in *discourse*, the listener can make use of the situational context to understand the utterance. Contextual cues can include, for example, the immediate situational context or the manner of production of the utterance. If (4) is the final line of an ongoing description of the box as in (8), the listener will take this as a sign that it was the box that broke, not the ground.

- (8) A There is a box that sits on a platform and now a ball rolls in from the right.  
B Aha.  
A Now the box is pushed to the left and - oh - the box falls to the ground and it breaks.

Using such contextual cues for interpretation goes beyond syntactic and semantic analysis and is studied in the field of pragmatics. As an example for the study of pronoun reference, Grosz et al. (1995) propose *centering theory*, which contains rules of using pronouns in discourse according to which objects are *in focus*. The basic idea could be summarized as follows: For each utterance within a discourse, there is a discourse entity in focus that is already established, the *backward-looking center*, as well as a number of discourse entities that can come into focus next, the *forward-looking centers*. The backward-looking center then connects the utterance back to the last one and the reference of the pronoun is established through focus. The list of forward-looking centers is also ordered by how likely they will come into focus. One of the examples they discuss is shown in (9), where *Susan* is probably the backward-looking center in the second and third utterance.

- (9) a. Susan gave Betsy a pet hamster.  
b. She reminded her that such hamsters were quite shy.

---

<sup>15</sup>Often attributed to Bever (1970), p.316



- c. She asked Betsy whether she liked the gift.<sup>16</sup>

Grosz et al. (1995) note that given only (9a) and (9b), one could assume that the reference of the pronoun does not favor Susan or Betsy. They argue that this is wrong and Susan is indeed favored. As evidence, they provide examples that introduce changes in focus, i.e. necessitate to resolve the initial pronoun to Betsy instead of Susan. The example they present as most unintuitive is shown in (10), where the backward-looking center in (10c) is Betsy. It is important to note that Grosz et al. (1995) state that discourse like that in (10) is not *wrong*, but seems *less coherent*. This means (10) simply requires more *inference* on the part of the listener than (9).

- (10) a. Susan gave Betsy a pet hamster.  
b. She reminded her that such hamsters were quite shy.  
c. She told Susan that she really liked the gift.<sup>17</sup>

Note that in (8), resolving the reference to *the box* also seems intuitive because the box is the main object of the story. This *standing out* of a discourse entity that Grosz et al. (1995) call being *in focus* has many names in the literature. Rosa et al. (2017) for example say that the discourse entity is *salient*. The listener can use this to form an opinion on the reference of the pronoun, but it will only be a heuristic shift in probability. For example, if a word is in subject position, it is usually more salient than other words unless there is a shift specific to the situation. However, there are more factors involved, Garvey et al. (1974) for example found that certain verbs can produce a strong tendency to resolve a following pronoun to the subject, e.g. *approach*, while other verbs do the same for the object, e.g. *praise*, as demonstrated in (11). Hartshorne et al. (2015) even provide the mirrored example shown in (12), where only the verb changes but the pronoun reference seems to change as well. Note that (12) is very similar to WSs too. The overall phenomenon is often called *implicit causality*, since the examples rely on sentences using *because*.

- (11) a. Peter approached Paul because he needed water.  
b. Mary praised Jane because she was being thoughtful.
- (12) a. Archibald angered Bartholomew because he was reckless.  
b. Archibald criticized Bartholomew because he was reckless.<sup>18</sup>

In summary, the reference of ambiguous pronouns can often be resolved on the basis of cues like salience in discourse or heuristics like implicit causality. Note that in cases like (12), while there is a bias induced by the verb, the pronoun is still strictly ambiguous. If Bartholomew is known for violence, Archibald could be reckless to

---

<sup>16</sup>(Grosz et al., 1995, p. 211)

<sup>17</sup>(Grosz et al., 1995, p. 212)

<sup>18</sup>(Hartshorne et al., 2015, p. 1)

criticize Bartholomew, the same is true for (11). Nonetheless, a listener will use the available cues to resolve the ambiguity if possible. It is a common finding that listeners are really good at resolving ambiguities, not only for pronouns but in general. H. H. Clark et al. (1977) even propose an *ambiguity paradox* because there is a lot of ambiguity in everyday utterances, but it is rarely noticed. In light of the problems of ambiguity outlined above, Wasow et al. (2005) ask: 'If ambiguity is undesirable however, why is there so much ambiguity in everyday language use?'. As we have seen for pronouns above, the common answer is that ambiguity is resolved in discourse or even left open, see Winter-Froemel et al. (2015). If listeners are really good at resolving ambiguities, the previous assessment has to be reconsidered: ambiguity is not really undesirable but rather harmless, as it does not actually make communication fail. 'Ambiguity avoidance is overrated', as Wasow (2015) puts it. This is because listeners are well equipped to handle utterances that are ambiguous.

### 3.4 Types of Ambiguity and World Knowledge

Given the discussion above, WSs present a curious case, as the pronoun is ambiguous, i.e. even after applying syntactic and semantic analysis it can be assigned different referents. Further, a listener will have no problem understanding a WS even though there is no further situational context. Discourse salience, as in the centering theory, or heuristics, e.g. implicit causality, do not help. Instead, WSs rely on world knowledge. In the following, we want to look at ambiguity from the perspective of how it is caused and how it is resolved.

As a starting point, we follow the discussion of ambiguity by Winkler (2015) and Winter-Froemel et al. (2015). They distinguish *ambiguity in the language system* from *ambiguity in discourse*. Note that the term *system* here refers to the system of language and is different from artificial systems as used in this work at other times. Ambiguity in the language system refers to cases of ambiguity, as introduced above, where a linguistic item, e.g. a sentence or a word, can be assigned different distinct linguistic interpretations after applying linguistic knowledge. In addition to ambiguous pronouns discussed above, a classic example of this is *lexical ambiguity*, where a lexical item has distinct meanings, e.g. the word *bank*.

Ambiguity in discourse on the other hand refers to cases where an utterance is embedded in an actual discourse situation and can be plausibly assigned several distinct meanings arising from the context. The utterance in (13) is a simple example, but there are more complex cases too. Assume the utterance in (14) is produced in a situation where it is clear to the interlocutors that one of Thom's sisters has a very difficult relationship with Thom. This generates two distinct interpretations, i.e. that Thom either visited that specific sister, or he visited another sister, which could make a huge difference to the listener. This would be an ambiguity in discourse.

(13) There is a bank now on the corner of our street.

(14) Thom visited his sister.

It is important to note that (13) as an utterance in discourse is an example of ambiguity in discourse, but as an isolated sentence is also an example of ambiguity in the language system. The utterance in (14) is also an example of ambiguity in discourse given the situation described above. But as an isolated sentence it is *not* an example of ambiguity in the language system; there is only one linguistic interpretation of (14). Sennet (2016) even argues that a sentence like (14) may have distinct interpretations in conversation, but is *not* a case of ambiguity, restricting the meaning of ambiguity to *ambiguity in the language system*. Here, we follow the alternative approach and assume that (14) *is* a case of ambiguity, but in discourse only, not in the language system. The central condition is that the listener can assign distinct interpretations to the utterance.

An important difference between (13) and (14) lies in the fact that the ambiguity is caused by the language system in (13) but by the situational setup in (14). To categorize both examples in this work, we can say that (13) is an ambiguity in the language system but may be ambiguous in discourse, i.e. the ambiguity is *open*. If the listener just asked about the closest ATM to withdraw money from, they will resolve the ambiguity in (13), because bank should mean financial institute in this case. That listeners often resolve ambiguities during discourse is a common finding and will be discussed in more detail below. Here we only need to distinguish between an ambiguity being *open* and being *resolved*.

Returning to WSSs, the ambiguity there is resolved by world knowledge. Following the discussion above, we can also say that an ambiguity is *caused* by world knowledge. An example of this could be a box that has a 0.5 chance of exploding upon impact on the ground when falling. If the explosion seriously alters the landscape, this difference would be relevant. Then the utterance in (15) would then be ambiguous because the landscape could be altered or stay the same. Note that calling (15) ambiguous requires an understanding of ambiguity as in ambiguity in discourse. The reason to differentiate such world knowledge related ambiguities and ambiguity in discourse is that world knowledge is not situational, i.e. world knowledge related ambiguities are cases that are ambiguous in isolation but not because of an ambiguity in the language system.

(15) The explosive box falls and hits the ground.

It may be argued that the world knowledge described above refers to a specific mini world and is thus situational, in the sense that we can ask: ‘which mini world are we talking about?’ This misses the point, however. We introduced mini worlds as *absolute* frames of reference and the agents that inhabit the world cannot leave that frame of reference. As an utterance *in* the mini world, (15) is resolved through world knowledge, and this is what we are focusing on here. In the following we thus assume that for utterances that refer to object in a mini world, general knowledge about that mini world is world knowledge and what is happening in the mini world at any point

in time is discourse context. This mirrors the distinction between world knowledge and discourse context in the real world.

There are borderline cases where it is difficult to decide whether some piece of knowledge is situational or general world knowledge, e.g. who the leader of a specific country is at a specific time. At other times however, the distinction is clear, as in the case of Ws, which are not dependent on context. Note that the ambiguity in Ws is not *caused* by world knowledge but by the language system, it is only *resolved* by world knowledge. A summary of the discussion above is given in examples (16) to (25). Examples (16), (17), (18) and (19) are caused by the language system. (16) is a temporary ambiguity that is resolved by the language system, (17) is the classic WS example and is resolved by world knowledge. (18) is a case of lexical ambiguity that can be resolved by discourse information and (19) is unresolved even in discourse.

- (16) The horse raced past the barn fell.
- (17) The city councilmen denied the demonstrators a permit because they feared violence.
- (18) A I'll need to get some cash before we go.  
B There is a bank now on the corner of our street!
- (19) A The view from your window is much nicer now!  
B Yes, they redid the whole block, planted a lot of trees and renovated some of the buildings.  
B It's much nicer to look at, and there is a bank now on the corner of our street!

Examples (20), (21) and (22) are *caused* by world knowledge. As described above, the examples assume world knowledge about a mini world where an explosive box has a 50% chance of exploding, heavily altering the mini world. (20) can be resolved by world knowledge without further context. (21) is resolved by discourse, while (22) is unresolved.

- (20) The explosive box fell and hit the ground but everything is in order.
- (21) A Wow what happened to that part of the game world there.  
B The explosive box fell and hit the ground.
- (22) A So what is happening right now?  
B The explosive box fell and hit the ground.

Finally, examples (23), (24) and (25) are *caused* by discourse context, showing the difference to world knowledge which is context independent. The ambiguity in (23) can be resolved by gender, i.e. the language system, whereas the ambiguity in (24) is

resolved through discourse knowledge. Finally, the ambiguity in (25) is unresolved, even if there may be a bias that Thom rather visited the sibling he gets along with. Note that the example discourse in these examples may seem slightly unnatural, using *sibling* seems unintuitive. Since the examples only serve the purpose of distinguishing types of ambiguity, this is acceptable in this case.

- (23) a. Thom has a brother in Berlin that he gets along with really well but also a sister that he always fights with when they see each other.  
b. Last weekend, Thom visited his sibling and she was in a good mood.
- (24) a. Thom has a brother in Berlin that he gets along with really well but also a sister that he always fights with when they see each other.  
b. Last weekend, Thom visited his sibling in Berlin.
- (25) a. Thom has a brother in Berlin that he gets along with really well but also a sister that he always fights with when they see each other.  
b. Last weekend, Thom visited his sibling.

### 3.5 Information Gaps and the Utility of Ambiguity

There is a common feature of the examples above, namely that in cases where an ambiguity can be resolved, it can be resolved because there are *enough* pieces of information about the meaning of the utterance that are available to the listener. In cases where the ambiguity remains open, there is missing information. Note that the information is missing *in the utterance* if there is ambiguity, but is provided either later in the utterance or through world knowledge or context if the ambiguity is resolved. In the case of Ws, the information that is needed to resolve the ambiguity is part of general world knowledge. We call this phenomenon an *information gap*.

**Information Gap** A part of the meaning of an utterance that is not given through compositional structure but is to be inferred by the listener.

Because information gaps are bound to the meaning of the utterance, they avoid being trivial underspecifications. If information is missing from an utterance which is not relevant to the utterances meaning, there is not necessarily an information gap. In the case of (4), repeated below, the ambiguity arises because there are two possible referents of *it*. The information gap arises because the information about *what breaks* is missing in the utterance. In this case, the ambiguity is supposed to be resolved through world knowledge. So we can say that the information gap has to be *filled* by applying world knowledge.

- (4) The box falls to the ground and it breaks.

The exact nature of the inference that is necessary here will be the subject of section 4. When talking about resolved and open ambiguity, the question really seems to be whether there are information gaps that can not be filled, something we call *destructive information gaps*. A destructive information gap occurs when there is not enough information *overall* to infer the correct meaning.

Recall examples (23), (24) and (25). In discourse, there is information provided that characterizes each sibling of Thom. Right after *sibling* in the second utterance of each example, there is the temporary ambiguity of which sibling Thom visited, which is the information gap. In (23), there is extra information about the gender of the sibling, thus allowing for inferring which sibling is meant. It is important to note that the additional cue does not *fill* the information gap, but only provides additional information. The filling of the gap still has to be done through inference. This becomes especially clear in (24), which exhibits the exact same information gap but provides a different cue, namely the location of the sibling. This again enables the listener to fill the information gap. In (25), there is not enough information, the information gap cannot be filled and is actually destructive.

World knowledge is again a curious case, because it is by definition neither part of the discourse nor of the utterance. Thus if an information gap is filled by world knowledge, that means that there *are* informational cues in the utterance or in discourse, but the listener still necessarily needs to draw upon world knowledge for *additional* cues to be able to do the inference. These additional cues are context independent and often difficult to discuss because they are never made explicit. WSs are examples where it becomes very clear that world knowledge is necessary, but what kind of world knowledge is needed remains elusive.

Since human listeners are often very good at resolving ambiguity and inferring the meaning of an utterance, it seems that destructive information gaps are relatively rare in natural discourse. It can even be argued that information gaps have a function, because using language comes at a cost, i.e. producing and understanding utterances binds cognitive resources. A speaker should avoid unnecessary detail and produce utterances that are only as long as they need to be. This was, for example, noted by Grice as a part of his maxim of quantity (Grice, 1967). Unnecessary information is confusing to the listener but also inefficient for the speaker. An utterance should be kept as short as possible. If we assume that listeners can fill information gaps, this can be used to make utterances shorter.

The argument about efficiency, focusing on lexical ambiguity and the length of words, has been discussed in the literature to some extent, see e.g. Levinson (2000), Piantadosi et al. (2012), Zipf (1950). The main point is that shorter words are more often ambiguous in their meaning to allow reuse, i.e. use the supposedly limited number of short words in a vocabulary in as many situations as possible. Shorter words lead to shorter utterances and reuse allows more utterances to be made up of these shorter words. For example, Piantadosi et al. (2012) argue that overloading short

words makes language more efficient overall: as long as the meaning can be inferred, short words *should* be overloaded as much as possible. Note that there is criticism to this, e.g. Wasow (2015) argues that there are still many short potential words that are not part of the English vocabulary, so why overload some but dismiss others?

In summary, we can say that listeners are able to fill information gaps and thus resolve ambiguity if there are enough cues. This can be used to produce shorter utterances. We hypothesize that human speakers will also *produce* information gaps if world knowledge provides enough information to fill them. Recall that the inference needed for filling information gaps is a gradual affair, i.e. the cognitive cost of inference for the listener can vary, as in examples (9) and (10) that Grosz et al. (1995) discuss. As opposed to ambiguity in the language system, where an example is often either strictly ambiguous or not, an information gap can thus have varying *cost*. Speakers should ideally manage their production of information gaps accordingly. We will provide evidence of this in section 13.

### 3.6 Modeling Language Processing

This concludes the discussion of language related theory. In summary, a model of language processing should implement functions to analyze the syntax of an utterance and retrieve the semantic meaning of all lexemes. It should then combine these to construct the meaning of the utterance and infer additional information in case of information gaps. In this work, we will ignore discourse context and processing heuristics such as implicit causality biases or those proposed by centering theory. This leaves us with three basic requirements to the model.

1. Each lexeme has its meaning stored in memory; the model has to build and implement such a storage.
2. There is a module that encodes rules as to which role a word can play in a sentence.
3. The meanings are combined with their roles to form the meaning of the sentence. If there are information gaps, additional information has to be inferred from world knowledge. The model has to implement functions to build combined meanings and to do inference.

This however raises several questions that the model must answer.

1. What is the meaning of a lexeme, i.e. what kind of encoding is stored in the mental lexicon?
2. What are the roles that are assigned by the grammatical system, i.e. what kind of representation is produced by the syntactic analysis function?

3. How does the model satisfy *knowing truth conditions*? How does the model do inference?

In the following, we will discuss inference and develop a model. Since our artificial system focuses on world knowledge, this will be our main concern. The focus on world knowledge will also lead to a model of *understanding*.



## 4 Meaning and World Knowledge

Consider once more the original example proposed by Winograd:

- (1) a. The city councilmen refused the demonstrators a permit because they feared violence.
- b. The city councilmen refused the demonstrators a permit because they advocated revolution.<sup>19</sup>

In the last section we have discussed how the linguistic structures of (1) would be analyzed and what role the ambiguity of *they* plays. The main message was that (1) contains an information gap that a listener has to fill through inference, in this case by using world knowledge. In this section, we want to discuss theories concerning this process.

Let us try to approximate what kind of knowledge is necessary (1 a) and let us, for now, conceive of knowledge as *propositional knowledge*<sup>20</sup>. In this work, we can simplify and assume that propositional knowledge is knowledge that can be described by simple sentences. The simplest kind of knowledge that would allow to assign the correct reference in (1 a) would be 'If city councilmen fear violence, the city councilmen will refuse demonstrators a permit and if not, it does not really matter whether the demonstrators fear violence or not.' It seems implausible to say, however, that this is what people *know* in some explicit sense, i.e. have memorized. If this was the case, people would need to store incredible amounts of explicit knowledge in their memory, especially all kinds of inferences. It seems more plausible to assume that world knowledge is more general as in the following. Using such knowledge, the listener could infer the correct reference in (1) using logic.

1. Demonstrators may sometimes turn to violence while demonstrating.
2. Violence during demonstrations is against the law.
3. Breaking the law is undesirable for governing bodies.
4. A city council is a governing body.
5. A demonstration has to have a permit from the local governing body.
6. ...

Propositional knowledge provides an intuitive description of how knowledge is stored and is operated on. Introspectively, it often seems to be the case that thoughts are built as natural language sentences and the process of thinking is an inner monologue.

---

<sup>19</sup>both Winograd (1972, p. 33)

<sup>20</sup>This explicit propositional representation of knowledge is used for easier analysis here.

This is problematic in several ways, however. First, it is generally established that there are certain skills that the human mind possesses and that we attribute to cognition where a description in natural language terms seems counter intuitive. Riding a bike (or other entrained motor skills) is an example of this, and riding a bike clearly makes use of cognitive processing. This first problem could be dismissed because it is not clear (yet) that the human mind would need skills on the level of bike riding when doing inference in language comprehension. We have to be careful though not to restrict a model of inference to conscious or explicit thought. It seems clear that in resolving (1), not all details of inference are spelled out consciously during comprehension, so the lines are blurry in this case. In this work, we will not differentiate different *types* of thought but talk about *cognitive processing*, which is the focus of our model.

The second problem with the thought as natural language intuition is that this leads directly to the strongest version of what is often called the *Sapir-Whorf hypothesis*, see Scholz et al. (2020) for a discussion. If thoughts are natural language sentences then thought can only ever encompass anything that the mind could describe in natural language. Humans could only think when they knew the specific words. This has been severely criticized and remains a very controversial topic. The last and most severe problem is that if thoughts consisted of natural language, there would be a regress. In order to interpret a thought, the human mind would need a dedicated language faculty to do the necessary inference, which would once again happen in natural language thoughts, necessitating a language faculty again and so on.

More refined but intuitively related theories posit propositional knowledge where the propositions are not constructed in natural language but in a mental language, a *language of thought*, see e.g. Fodor (1979) or Rescorla (2019) for an overview. The regress is then avoided because the language of thought is supposed to *be* the mode of operation of the mind and therefore does not need interpretation. The language of thought becomes a functional description of what the mind does. Severing the connection to natural language of course raises the question of how the mental language is constructed and how it can be used in modeling.

Using natural language sentences to describe thought and inference is very productive nonetheless. Describing the hypothesized inferences in section 4 would have been difficult if not for natural language propositions. Note that the hypothesized description was given on a *functional* level, however, i.e. it does not necessitate that the process is also *implemented* in the mind this way. Recall that we are so far operating on the computational level of description, not on the algorithmic level of description. Indeed, many approaches to describing how world knowledge figures in thought resort to natural language descriptions and what one would usually call symbolic logic. Symbolic logic is used here in the sense of a set of rules or relations between propositions. If we treat propositions as simple entities, as it would also be required by a language of thought approach, an intuitive way of explaining how the mind operates on world knowledge is to propose explicit representations of such relations. This could mean, for

example, that the concept of a tree would have something like a ‘have-relation’ to leaves, an ‘is-made-of-relation’ to wood etc. The concept of falling could have a ‘from-relation’ to a former place, a ‘to-relation’ to a future place, a ‘at-speed-relation’ to a velocity and so on. This way, propositions are enriched with relations to other propositions to form a *network*. A popular variant of this approach are *frames*, see e.g. the discussion by Löbner (2012). An older variant is that of *scripts*, see Schank et al. (1977). Scripts are somewhat more specific and describe how certain events in the world will usually unfold, a classic example being the restaurant script that spells out the steps of going to a restaurant. They represent something like ‘happens-after-relations’ between more complex combinations of meaning units.

These approaches assume that the rules captured in frames or scripts are sufficient to do the necessary inference in comprehension. Recall example (1 a) and the inference described above. ‘City councilmen’, as an entity in the network, could have an ‘avoid-relation’ to ‘breaking the law’ and an ‘give-permit-relation’ to ‘demonstrators’. The ‘give-permit-relation’ would probably need to be combined from simpler parts. ‘Demonstrators’ could have a ‘possibly-cause-relation’ to ‘violence’ which in turn has a ‘does-relation’ to ‘breaking the law’. Note that this is a translation of the propositions hypothesized above into a network of relations that roughly approximates the idea of a network of frames.<sup>21</sup> Once the network is established, the inference mechanism has to compute that the sensible interpretation of (1 a) is that the city councilmen are afraid of the demonstrators causing violence.

The main problem with *building* such a network is that it would need to encode implications not only of simple configurations, but also of complex combinations. There are however also other reasons that make such a network seem implausible.

#### 4.1 Inference and Mental Simulation

Assuming we can construct a network of relations as described above, it seems that some WSs can be solved. Consider, however, the following two WSs, both relying on spatial configurations:

- (26) The sack of potatoes had been placed [above/below] the bag of flour, so it had to be moved first.
- (27) There is a gap in the wall. You can see the garden [through/behind] it.<sup>22</sup>

One could try to describe the inference done here in propositional terms. Consider *above*, which would be a relation here. If we combine the sack of potatoes ( $S_p$ ) and the bag of flour ( $B_f$ ) with the above-relation, what relations apply to this combined entity? Further, is the above relation purely spatial, e.g. if  $S_p$  is stored in a cabinet above  $B_f$ , or

---

<sup>21</sup>Much research has already gone into networks of frames and there are certainly more elegant ways of constructing this example.

<sup>22</sup>Both Davis (2019), numbers 19 and 31

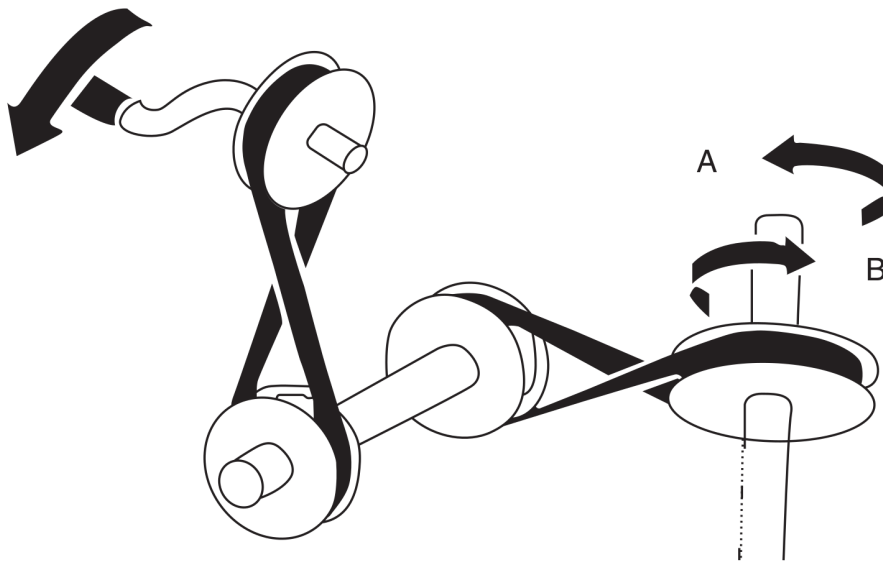


Figure 1: From Johnson-Laird (2008), p.30. Is the resulting movement A or B?

is the above relation directly relational in the sense that  $S_p$  exerts some force on  $B_f$  *from above*? For now let us assume that the mind decides on the second option, so  $S_p$  exerts force on  $B_f$  from above in this case, i.e.  $S_p$  is actually lying on  $B_f$ . This force must have a parameter to encode *how much* force is exerted. If dust was lying on  $B_f$ , the relation would be similar but the dust would *not* need to be moved first, because it does not exert enough force. Such parameters are difficult to handle with propositions, so the force would need a new proposition such as 'a force great enough to make it difficult to move the object'. This, however, gets complicated quickly.

One idea to circumvent the parameter proposition problem is that the mind uses internal *mental models* which operate not on symbolic representations but on spatial representations. A particularly convincing example of the mind operating on such spatial representations can be seen in figure 1. To answer the question whether turning the handle in 1 results in movement A or B, the mind seems to *simulate* the movement, which Johnson-Laird (2008) calls *mental animation*. Such an operation seems to be much easier done by a model than a network of propositions. To even describe the system in figure 1 using propositions seems extremely tedious. It is much simpler to have a system that can represent spatial relations, e.g. through coordinates, and then operate on those.

Johnson-Laird (2008) proposes that human reasoning *in general* works through such mental simulation, a claim that we need not discuss here. In this work we are only interested in how mental simulation can provide a sensible component for modeling language processing. Let us consider (26) again and assume that a model is available

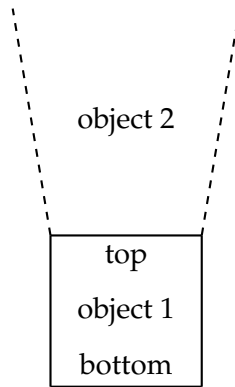


Figure 2: A possible representation of an above-encoding.

that can compute the interactions and forces in the scene, i.e. a model of how a sack of potatoes  $S_p$  and a bag of flour  $B_f$  can move if  $S_p$  is sitting on top of  $B_f$ . The mind would need to start from a  $S_p$ -on- $B_f$  scene and then simulate further developments. Since the  $S_p$ -on- $B_f$  scene is a *spatial representation*, we will need to explain how it is constructed from the input sentence, i.e. how the mind converts the relevant part of the input sentence to a spatial representation. One way to do this would be to use what we may call spatial encodings, i.e. encodings of recurring spatial patterns. Such encodings are a common theoretical entity in the literature and are, for example, proposed by Lakoff (1987) as *image schemas*. The above-encoding would be a spatial pattern as in figure 2.

Once the spatial representation is established, we need to judge whether  $S_p$  or  $B_f$  has to be moved first. Recall that  $S_p$  sits on  $B_f$ . In this case, moving  $S_p$  first and  $B_f$  second would work. Moving  $B_f$  first requires a much greater force and  $S_p$  could drop. If there is a mechanism that rates these two simulations, that mechanism could infer that the first is more plausible and thus resolving the *it* to the sack of potatoes in (26). We will revisit this whole resolution process later, for now the important message is: the model that does the simulation on the  $S_p$ -on- $B_f$  scene encodes some knowledge about movement, position, forces, weights and so on. Even if there was a strict propositional implementation, it seems much simpler to use a model that contains parameters. Together, spatial encodings and mental models allow for easy inference also on spatial relations.

The most intuitive examples of the power of mental simulation stem from spatial relations, the approach may also scale to other inferences, however. This would require *simulating* what the city councilmen do. It could be argued that inference over propositions and inference through mental models are simply two different descriptions of the same process, leading to the same result on the same input. We will see, however, that mental models lend themselves much better to modeling inference than propositional accounts do in the cases that this work focuses on.

It is important to note that *knowledge* works differently for propositions and models. Without going into much detail, we can say that the artificial system has world knowledge if its network of propositions is *true*, i.e. the propositions are true and the relations between them are correct. For mental models, we can say that the artificial system has world knowledge if the models allow for accurate *prediction* and *simulation*. In the end, this describes the same state of affairs, namely that the system's internal encoding of the world is faithful to how the world *is*.

## 4.2 Embodiment and Grounding

One thing to note about mental simulation is that the term is also used in a different, more specific way. Here, we introduced mental simulation as the operation of general models of the world that seem very intuitive for the spatial dynamics but could also be broadened to work on more abstract entities. A different approach aims at simulations in the motor-system specifically, see e.g. Glenberg et al. (2012) or Kaup et al. (2010). The *models* are basic motor-programs in this case and the simulation activates these without activating the associated physical movement.

This more specific approach falls under the wider category of *embodied cognition* and was also proposed to explain language understanding, as in Glenberg et al. (2012). Unfortunately, embodied cognition does not present a homogeneous framework and different approaches may vary quite profoundly, see Wilson (2002) for an overview. Most approaches will, in one way or another, aim at shifting the load of cognitive processes to the body, away from the abstract symbolic approaches of early artificial intelligence research.

In terms of language, this also holds a promise that we already tapped into above. Trying to come up with a propositional network that explained the logical implications of something as simple as 'above' seems very hard. Using a spatial relational encoding instead, one that the human mind supposedly already has anyway, seems much easier. Hampe (2005) discusses image schemas, which can be seen as spatial relational encodings here, and calls them 'embodied' but also *directly meaningful*. Hampe takes them to be 'preconceptual structures, which arise from, or are grounded in, human recurrent bodily movements through space, perceptual interactions, and ways of manipulating objects'.<sup>23</sup>

In the quote above, Hampe (2005) also describes image schemas to be 'grounded' in bodily interactions. The problem of grounding originates from symbolic approaches to cognition and language and the famous Chinese room argument brought forward by Searle (1980), see e.g. Cole (2020) for an in-depth discussion. The argument can be summarized as follows: imagine being locked in a room, having a near infinite amount of time and a huge book. From time to time, a piece of paper will be slid under the door containing combinations of lines and curves which are indeed symbols of the

---

<sup>23</sup>Hampe (2005), p. 1

Chinese language. The book contains detailed instructions on what to do with such symbols so that you only need to follow the instructions (you have unlimited pens and paper at your disposal too) but no explanation ever of what any of the lines or curves actually do. You follow the instructions and slide the end result under the door. On the other side of the door, there is a speaker of Chinese. To her it seems as if there is someone in the room communicating in Chinese with her (at a very slow pace, but still). Searle (1980) takes this to be analogous to a completely functional computational model of language processing. If the model was perfect, it would seem as if it actually communicated in Chinese. Searle however holds that it seems unreasonable to assume the person in the room (or the model for that matter) actually *spoke* or *understood* Chinese. The model does not understand anything at all, the symbols it operates on have no meaning to it whatsoever.

One reply to this argument is to say that the symbols within the model need to be linked to the real world 'in the right way', see e.g. Fodor (1987). If the link was established, the symbols would get their 'meaning' from the real world entities that they are connected to. This led Harnad (1990) to label the question about how symbols get meanings as the *symbol grounding problem*. Harnad thinks that the proposal to simply link symbols to real world referents 'radically underestimates the difficulty of picking out the objects, events and states of affairs in the world that symbols refer to, i.e., it trivializes the symbol grounding problem.'<sup>24</sup> For symbol grounding, as for embodiment, a variety of interpretations of the term has arisen. On the one end of the spectrum, the original Chinese room argument led to discussion about consciousness (see Cole (2020)) and the grand question of the relation between mind and body, aiming at core questions of Philosophy. On the other end of the spectrum, Steels (2008) claims to have solved the problem with a system that is able to establish stable encodings for colors that it is presented with. This is supposed to make clear how there are a variety of research projects that seem to target different goals.

In this work, the discussion of consciousness and the relation between mind and body is way out of scope. At the same time, discussing whether certain processes that are traditionally seen as computations within cognition are actually realized in smart bodily structures is also out of scope. In the above discussion however, both embodiment and grounding were also taken to refer to the connection of language to basic cognitive capacities and sensory input. To avoid ambiguous terminology, this work will refer to this connection as a *mapping*, so the question becomes how symbols are *mapped* to non-symbolic entities within the cognitive system. Mental model and mental simulation are used in the more general sense introduced above and do not necessarily involve the motor system.

---

<sup>24</sup>Harnad (1990), p. 340

### 4.3 Truth Conditions and Scene Reconstructions

Given the above discussion of inference, mental models and grounding, we can revisit the requirement that understanding an utterance means knowing its truth conditions. Recall that knowing truth conditions is equal to knowing which real world situations the utterance denotes. Even though Harnad (1990) pointed out that grounding is highly non-trivial, let us assume we have a grounded artificial system, either containing a network of propositions or mental models. We will focus on mental models in the following, so let us assume the artificial system has fully functioning models of its world. If we assume grounding, we can bypass truth conditions and focus on knowing which real world situations the utterance denotes. To understand an utterance, the artificial system then has to *reconstruct* the utterance in terms of its mental models which, by being grounded, corresponds exactly to the situations that the utterance denotes.

What makes the symbol grounding problem so difficult, as Harnad (1990) pointed out, is that even if we are able to link object category names to sets of objects in the real world, or color names to wavelengths, this link will most likely be lost once we have to combine symbols into complex compositional structures. The problem is thus building *reconstruction* but maintaining the mapping. In this work, we reserve the term *scene reconstruction* for this purpose. Note that the idea that comprehension is equivalent to reconstructing the described scene in models has been proposed by others, see e.g. Zwaan et al. (1998).

**Scene reconstruction** Given a grounded artificial system, the scene reconstruction is the internal reconstruction of what an utterance denotes in terms of the system's model of its world.

In terms of city councilmen and demonstrators, assume the artificial system is able to simulate the two possible interpretations of (1 a). Then only one of the simulations adheres to model predictions, i.e. the model only predicts that the permit is denied if the city councilmen fear violence, not (so much) if the demonstrators fear violence. This resolves the ambiguity, but the simulation also serves as the description of the real world states that (1 a) denotes by virtue of being grounded. In a very limited sense, SHRDLU (Winograd, 1972) *was* grounded, but this grounding was established by hard coding the link, making it difficult to extend.<sup>25</sup> As a next step, we will discuss how an artificial system can acquire mental models in order to be able to simulate and reconstruct utterances.

---

<sup>25</sup>As a side note, Searle (1980) explicitly denies that SHRDLU carried natural language meaning.



## 5 Modeling World Knowledge

Our model starts out with no prior knowledge, especially without the ability to comprehend language. Information encoded in language is thus not available to our artificial system. The goal of this work is to create a system that learns world knowledge from the world directly. To judge whether the model has acquired certain pieces of world knowledge, we can check whether the system is *surprised* by a certain given information or whether it can predict it. The standard measure for this is *prediction error*, which a successful system will minimize.

Minimizing prediction error by itself does not suffice in this work however. Our artificial system should not only learn world knowledge but it needs to be able to connect the learned models to natural language too. A huge generic neural network, such as those introduced in 2.3, may be excellent at reducing prediction error but difficult to link with natural language expressions afterwards. Instead, we will look into current theories in cognitive science.

### 5.1 Predictive Processing Accounts

One of the most promising approaches to a unified model of cognition revolves solely around hierarchical layers of prediction error minimization and has been named the *predictive mind* by Hohwy (2013) or *Predictive processing (PP)* by A. Clark (2016). The theory is based on a mathematical foundation, see e.g. Friston (2010), and older neural modeling, see e.g. Rao et al. (1999). It aims at proposing a somewhat general theory of cognition or ‘unified brain theory’ and has received a lot of attention over the last years.

In order to describe PP, let us assume that cognition always has a *current state*. This state reflects what the person currently thinks about (the world), does, wishes etc. In terms of the brain, the current state determines how the brain will react to incoming input. An intuitive idea of how the brain processes incoming sensory data is *bottom-up processing*, i.e. sensory data is fed into the system at the ‘bottom’ and is then processed in different stages, making its way up to the top to influence the current state of the system. Imagine the process of seeing a bear. Let us idealize and say the sensory input is an image-like representation of colors and light intensities. The visual system may now make out edges and areas of specific colors, an object recognition system may identify *fur, brown, big* and combine these into *bear*. Further up, a system may now take this and integrate it into the overall representation of the current environment, maybe increasing a danger level. This would be a bottom-up process.

The driving idea of PP on the other hand is *top-down processing*. Wiese et al. (2017) point out core features of PP, putting top-down processing as number one. Top-down processing starts at the top with a specific *prior expectation* and then predicts how lower levels of processing *should* behave if the prediction was correct. Consider the bear situation again, although now we also need to set a prior expectation. Let us say, the

person was looking at a forest and in that spot where the bear appears the person expected a patch of moss in the distance. So the prior would be *moss* up top, sending down the expectation of *green* and 'no real structure' to the object recognition system we assumed above. This system will take this prediction and send expectations of certain edges (none really) and colors (shades of green) to the visual system further down. This reversal is why this is called top-down processing. When the bear enters the scene, the visual system will now generate an *error*, i.e. neither are there shades of green in that spot nor are there no real edges. The object identification system will likewise generate an error now, i.e. that can not be moss. Now the top-level prior has to be changed.

The errors have to be very specific in order for the system to run. If the errors do not carry enough information, e.g. only give the feedback 'wrong', the system will have to test every possible prior it can think of, until finding the correct one ('bear') by chance. It seems plausible to assume that such a procedure would take too long, reducing the systems chance of survival in the face of bears significantly. Note that this also holds for the bottom-up system. Where the top-down system has to have smart error-handling, the bottom-up system has to have smart identification systems. The advantage of a top-down system is that the error provides feedback for learning, so the system can try to reduce the overall magnitude of errors in the short- and the long-term to build knowledge about the world.

Wiese et al. (2017) also mention that a PP system forms statistical predictions, i.e. a distribution over expected states. Instead of predicting 'moss' or 'bear', the system incorporates relative certainties, for example 'moss:98% , bear:0.1% (because it is a forest after all), other:1.9% '. Further points were already incorporated in the description above, i.e. a PP system is structured as a hierarchy of lower level systems and higher level systems, which predict each other and learn to minimize prediction error.

This leaves us with the abstract system in figure 3. Note that it is generally assumed that PP can be used to describe the process of selecting actions, see Wiese et al. (2017) but also Friston (2010) for a formalization.

It is important that the updating in figure 3 has two meanings or modes of operation, short-term and long-term updating (as noted above). A short-term update is a quick adjustment of current predictions, i.e. when the system predicts moss but gets an error and quickly adjusts to predict bear (and maybe a flight reaction). This short-term error correction basically updates the internal states of the layers as they are predicted from above, i.e. the current representation of the world. Short-term updating is important for the model to be able to function in a dynamic environment, but it does not facilitate learning. The 'bear-prediction' and all its necessary components has to exist already in order for the short-term updating to work correctly. The system thus needs *long-term updating* which is able to change the prediction function of a layer, i.e. *learn encodings*. This part is often difficult to model, but absolutely necessary. The

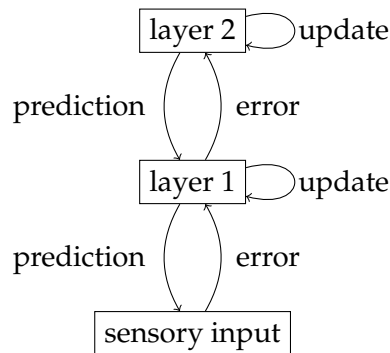


Figure 3: A first depiction of a PP system.

difference between short-term and long-term, between updating the internal state and updating the generative function, will appear very often from now on.

Given that the model can also produce action choices, a functioning PP system is supposed to be able to implement an autonomous agent, learning about its world as far as the hard limits of its layers allow. The model forms predictions about the future and thus facilitates mental simulation. We can call a model that can produce a full prediction of its environment a *generative model*. Unfortunately, the implementation of such a model is highly non-trivial. There is no simple algorithm yet that we can use for each layer and just let the system run. To make things easier, we can provide more structure so that the model does not need to learn *everything* from scratch.

## 5.2 Pre-Defined Structure and Event-Predictive Cognition

Concerning pre-defined structure, one approach would be to mirror what is known about the brain. I.e. replicate certain regions, equip those sub-models with processing capabilities usually ascribed to their related regions and link them up accordingly. This approach has been taken, see e.g. Eliasmith (2013). In this work, we instead try to build an artificial system inspired by what is known about psychological principles. This does not mean that knowledge about the brain is discarded.

In the discussion of world knowledge above, three areas were distinguished: firstly, the representation of objects and their features (object concepts). Secondly, the spatial representation of how a scene is set up (spatial relational encodings). And thirdly, the model that encoded how the scene will develop (mental models). In a similar spirit, Butz (2017) proposes three different kinds of encodings within a model such as ours.

1. *Spatial predictive encodings* for the spatial setup. Whether one object fits into another would be encoded through these.
2. *Top-down predictive encodings* for objects and features. These may predict specific spatial encodings too. What color(s) an apple should have is encoded through

these.

3. *Temporal predictive encodings* for the development of a scene over time. These encodings predict changes in the other two types of encodings over time. What happens if an apple falls from a tree would be encoded in these.

These three types should interact strongly. For example, in order to predict an appropriate reaction to a bear, the top-down encoding has to recognize the bear and the temporal predictive encoding has to analyze spatial predictive encodings to determine whether, how and when the bear could reach us. I.e. in reacting to the bear, the mind has to draw on all kinds of encodings.

In this work we want to focus specifically on temporal predictive encodings, i.e. how a scene develops and how it can be predicted. For this, we will draw on research on *event cognition*. The human mind tends to perceive certain periods as *events*. This has been well documented in psychological research, i.e. human participants will tend to not only divide a stream of sensory input into smaller units, they will also do this in a fairly objective manner, see e.g. Zacks et al. (2001). When participants were shown a video of every day tasks, the judgments of when to divide the stream would coincide quite nicely between participants, i.e. participants tended to divide the stream into *events* in a rather uniform manner. This even held when the participants were asked to change the granularity. Asking for a coarse-grained division would lead to a set of divisions which matched those of other participants. Equally, asking for a fine-grained division would do so too. This has been taken as evidence that the division of the temporal dimension into events has different levels of granularity, each of which is similar across different people. Zacks et al. (2001) thus define an event as a period in time that is perceived by the human mind to have a somewhat definitive start and end point.

These psychological findings have led to the development of event-cognitive models, e.g. in the form of event segmentation theory by Kurby et al. (2008). They propose that the mind applies different predictive models for different events, switching models when one event stops and another starts, i.e. at an *event boundary*. Because these models are specialized, they can also be used to detect these boundaries: if the model starts to produce large prediction errors, the event it originally matched must have ended and the model should be replaced. Such an unexpectedly large prediction error that calls for a model switch can be labeled as *surprise*. We will apply those ideas to build our artificial system.

### 5.3 Emerging Structure

The approaches outlined above aim at the emergence of abstractions, i.e. patterns or structures like those that seem to play a role in human cognition. Top-down predictions are supposed to build a hierarchy as in figure 3 with raw visual input at the bottom

and higher level object concepts towards the top. Each layer is thus supposed to turn a more abstract encoding into a prediction about a less abstract encoding further down. The same applies to temporal encoding of events. As event perception seems to have different layers of abstraction that are consciously accessible, event-encodings must also form hierarchies with more fine-grained segments further down and more coarse-grained segments further up.

Even a single layer will form abstractions. Recall how there are two different kinds of updating in figure 3. A short-term update corrects the current prediction as a reaction to the dynamic environment. Long-term updates on the other hand change how the layers behave in principle to minimize prediction error. This distinction carries over to the encodings described. There will always be currently active encodings, see also Butz (2017), adapted through short-term updates to represent the current state of the world. A top-down encoding may be active when seeing a bear, however, a certain movement-event encoding may be active because the bear is walking. On the other hand, there are at all times all the latent predictive encodings (simply predictive encodings in Butz (2017)) which model patterns in the sensory input but are not active. A bear should already have certain learned encodings associated with it because a human should not wait for the prediction error to predict that the bear may attack *in the near future*. This potential danger is a *pattern* that has to be learned. The way the bear walks or how fast it is are patterns that have to be learned. If such patterns were not encoded, prediction would fail because the connection between bear and danger could not have been learned. There would simply not be a pattern like 'danger' or 'bear', any animal that shows a flight reaction has to have some stimulus-response pattern set up. When such patterns are learned in encodings, an abstraction has been taken place from a recurring appearance of the pattern to its generalization. Each appearance will have had its own specific sensory input, including irrelevant information, and the generalization will abstract away from specific details. Note that the specific details may still be represented in the *active encodings* in each situation but will not be part of the learned pattern. The main hypothesis of theories of predictive processing is that the patterns that are learned and the structures that emerge ultimately allow the system to have the cognitive abilities of humans.

In our artificial system, we will focus on predicting events and how they follow upon one another. This will necessarily include their context, spatial configurations and objects within them. Since these models are predictive, the artificial system will be able to do simulations. This necessitates that an event contains all the information that enables the agent to predict how the event will proceed and which event will follow the current (Butz, 2016). This includes but is not limited to predictions about the spatial development of the scene and changes in attributes of the objects present. While focusing on a scene, an agent will build such predictive representations and continuously integrate incoming information into the representation, drawing on prior knowledge of the world's and the object's behavior. Note that Zwaan et al. (1998)

propose a similar model of language comprehension although their model remains theoretical, see also Zwaan (2003).

Further, the question of how inference is done in general is not immediately answered by PP accounts or event-cognitive approaches, even though they explain simulation. We will have to explore the problem of inference once the underlying mental models are established and available.

## 6 The Model and Main Hypotheses

At this point the most important parts of the overall artificial system of this work are in place and it is time to formulate a condensed abstract system that will be filled with life, worked out in detail, formalized and implemented over the next two chapters. In order to easily reference to this system, we call our system LEARNA, a loose acronym for *Learning Event Abstractions to Resolve Natural language Ambiguity*. As discussed above, LEARNA will learn about its mini world, build out encodings and have basic world knowledge. With this world knowledge, LEARNA should be able to do inference on and understand utterance, i.e. LEARNA will comprise basic language understanding. Since this represents a range of different functionalities, we can split LEARNA into three different parts following the description and discussion above.

1. **Event-predictive system (EPS)**: controls the agent in its environment, processes sensory input (but not language), learns predictive models and provides storage for learned encodings as well as the representation of situationally active encodings. This part of the system will be able to acquire world knowledge and use it in interactions. The EPS employs spatial, top-down and temporal encodings, the latter following the event-cognition theories. All encodings are predictive, therefore simulations into the future are possible.
2. **Language processing system (LPS)**: processes language interactions, parses grammar and operates a lexicon of known lexemes and their possible meanings. It is able to build different possible interpretations of incoming sentences if there is room for interpretation and will also be able to learn new words. This system thus incorporates syntax and lexical semantics, but only very limited compositional semantics.
3. **Inference simulation system (ISS)**: translates between possible interpretations and possible simulations or active encodings. In terms of language, the inference system is supposed to do what humans do when they *infer* the meaning of a sentence. For humans, this quite generally encompasses pragmatics and world knowledge. This model will focus on world knowledge, i.e. what the EPS can provide. It is important to note that the inference system also runs the *simulations* using the encodings provided by the EPS.

These parts can easily be differentiated because many of their internal operations are somewhat isolated from the other parts. They can communicate through different interfaces though. The interface between the LPS and the ISS stands out since it has its dedicated representational format that we may call an *Abstract scene representation (ASR)*. An interpretation produced by the language system may be formulated as an ASR, so the inference system operates on ASRs and can reduce active encodings from the EPS to an ASR. An ASR contains *selected* information about entities and their

dynamics over one or more *abstract* timesteps. They are called *abstract* because the timesteps are under-determined chunks of time and the information is reduced, i.e. only a few pieces of information are really included. The chunks of time correlate with events. Ideally, the selected information is the *relevant* information.

The ISS communicates with the EPS directly, i.e. it has direct access to all encodings and models. Note, however, that it has *access* but cannot *change* them, i.e. the ISS does not have the power to update any encodings. Finally, the LPS also has access to the EPS for linking lexemes with encodings. This communication channel is not used otherwise however. Figure 4 presents these interconnections.

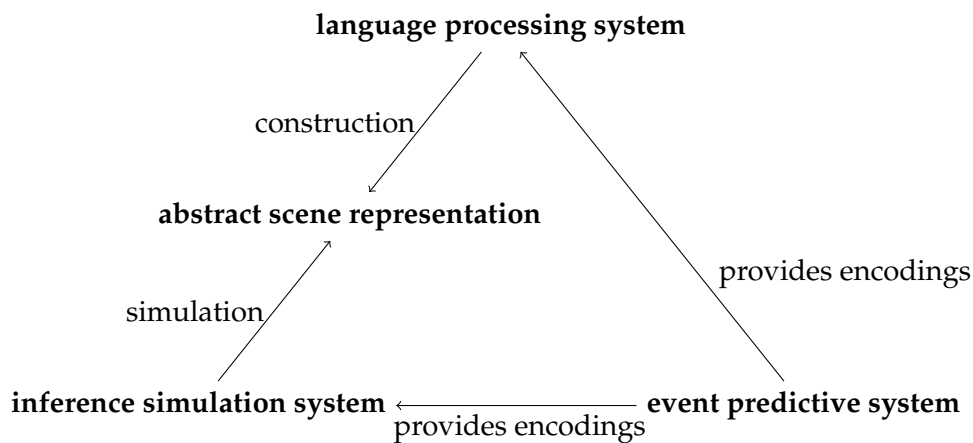


Figure 4: The three main parts of LEARNA and the role of ASRs.

## 6.1 Main Hypotheses

After having established a theoretical basis for the modeling and empirical parts in this work, we can formulate the two main hypotheses.

1. **Mapping language to cognition:** we hypothesize that within LEARNA, structures emerge in the EPS that we can directly link to lexemes, so the mental lexicon in the LPS contains mappings between lexemes and learned encodings in the EPS. The ISS further allows the mapping of complex utterances to complex representations, therefore considerably advancing the strength of the mapping. This allows the system to understand Winograd-Schema sentences that have been adapted to the mini world in which the system operates and, by extension, resolve different kinds of ambiguity.
2. **Ambiguity and information gaps:** we assume that ambiguity is often resolved by the inference mechanism through world knowledge. In cases where this is not possible, the inference mechanism misses specific information, i.e. there is



a destructive information gap. Information gaps are nonetheless useful if they can be expected to be filled by the inference mechanism because they lead to shorter utterances. We hypothesize that human speakers actively manage the production of information gaps in regards to their world knowledge to make use of this advantage.

## 7 The Event-Predictive System

In the description of the EPS, the goal was to create a generative model<sup>26</sup> that can predict its environment. This was to be done through a PP system of layers that predict lower layers and only receive the error as feedback. As additional pre-defined structure, spatial relational encoding, top-down predictive encodings and temporal predictive encodings were introduced. Spatial encodings predict spatial relations between and within (e.g. shape) objects, whereas top-down encodings build a hierarchy of abstractions over objects, states of the world and their features. Temporal encodings predict the dynamic development of the other two encodings and were further structured in terms of event-cognition. The following section deals with the question how the EPS functions, describing its parts on the computational level of description. Consider once more (4), repeated below, which will guide through the description as the leading example:

- (4) The box falls to the ground and it breaks

Let us now examine an event that could be described by (4), as seen in the sequence in figure 5. To use consistent terminology in the following, we say that such a dynamic sequence is a *scene*, which consists of discrete *timesteps*. Each of these timesteps we call a *situation*. A situation can be considered as a scene that was frozen at a point in time.



Figure 5: Event sequence described by *The box falls to the ground and it breaks*, the box disappears after hitting the ground.

### 7.1 Learning and Applying Encodings

First of all, it is important to note that encodings are neither necessarily symbolic nor necessarily what one would usually call a *code*, i.e. a list of numbers. As discussed above, a mental model encodes a pattern. An encoding might also just manifest itself through a stable internal state of a system. Capturing patterns, encodings arise from a specific *similarity* of the instances of those patterns. If boxes in the mini world of (4) always or often break upon hitting the ground, this can be encoded. If there are specific boxes that do not show that behavior, the similarity is broken and there is need

<sup>26</sup>Recall that a generative model is one that generates states by itself, as opposed to a model that can merely evaluate given states.

for two encodings. If there are objects that fall if not supported by something, this pattern can be encoded too. To learn an encoding, the similarity between the different instances has to be found and the instances have to be *clustered*. Once the encoding is established, it can be used to *classify* new instances as to whether they follow the same pattern, i.e. belong to the same cluster.

To give a nice example from the animal domain, frogs have (through evolution) developed the ability to catch flies in their field of vision with their tongue quite reliably (Lettvin et al., 1959). Interestingly, frogs will also try to catch a small black dot on a screen, even if it is round. The pattern must thus be that most edible things (for frogs) are small dark objects. The frog nervous system thus recognizes small dark objects (cluster) as something edible (encoding) and issues a catch movement, even if it is a dot on a screen (similarity). Maybe if black dots on screens were prevalent in frog environments, evolution would have found a more fine-grained encoding to not lash at screens too often.

The difficulty of computing the similarity of two encodings can vary quite a bit. On the symbolic end, it seems nearly impossible: the symbol cat is much more similar to the symbol car than to the symbol dog by the only available measure (letter by letter similarity). Cats should be more similar to dogs though. If a system used only symbolic encodings, it would probably have to learn each similarity between different symbols and remember it. On the other end of the spectrum, encodings live in a simple, high dimensional vector space of real numbers:  $\mathbb{R}^n$ . Each encoding is a vector  $v \in \mathbb{R}^n$ , so  $v = (x_1, x_2, \dots, x_n)$  where every  $x_i$  is a real number. To compute the similarity between some vectors  $v$  and  $w$ , there are a few simple mathematical formulas, the simplest probably being pairwise distance, i.e.  $\sum_{i=0}^n |x_i - y_i|$ .<sup>27</sup> Such encodings are called *distributed representations* because their content is distributed over the dimension of the vector space they live in. In between symbolic and distributed representations there is a near infinite mix of the two. One could, for example, make specific dimensions binary (yes/no), symbolic (nationality) or discrete (age in years), making similarity harder to compute but still possible. In the ideal case of distributed representations, data points corresponding to an encoding will cluster in neat isolated areas. Each of those areas can then be used as a categorization and boundaries between categories arise naturally.

A further advantage of vector representations is how easy it is to apply a function to vectors, as opposed to symbols. A probability distribution or any other function can be calculated over a vector space given a formula<sup>28</sup>, but has to be learned and memorized like a table over a set of symbols. Even better, if the function is smooth, changing the vector representation slightly will also only have a slight impact on the

---

<sup>27</sup>This formula would read: sum over the absolute values (i.e. drop a negative sign if there is one) of  $x_1 - y_1, x_2 - y_2$  up until  $x_n - y_n$ .

<sup>28</sup>Formula sounds like a complex mathematical tool that the mind would never actually implement. Most formulas in cognitive modeling only use basic arithmetic operations however, i.e. +, -, \*, /. All functions used in our system are build this way.

calculated output.

The main function of the EPS is to find good encodings, specifically top-down encodings, spatial relational encodings and temporal event-predictive encodings. ‘Good’ in this case means that the encoding captures a pattern, allows for classification correctly and ideally also allows for comparison *between* patterns.

### Top-Down Encodings

The EPS first builds top-down encodings of all new objects it encounters, in figure 5 this would mean encodings of *red box* and of *ground*. Whenever a red box appears after the encoding is built, the EPS can now activate that encoding. LEARNA builds encodings that include the size of the object and its most *typical* state in the world. If the system only ever saw the scene in figure 5 but it had a temporal encoding of *falling* it would memorize that red boxes are falling most of the time, for example. For top-down encodings, the EPS has to handle the fact that similarity between objects can be misleading.

### Spatial Relational Encodings

Once top-down encodings are established and all the objects in the scene are classified, the EPS builds spatial relational encodings. These describe patterns in spatial relations, for example whether an object is attached to another or whether they are just very close as well as the direction in which they touch. In figure 5, there would be encodings of the box being *above* the ground and the box touching the ground (for one timestep). Note that for spatial relations, similarity is easy to compute.

### Temporal (Event) Encodings

Once top-down encodings and spatial relational encodings are established, each *situation* can be classified in terms of spatial relations and entities. This allows building encodings of the temporal dynamics. The EPS should build encodings that capture uniform patterns. In figure 5 these would be the uniform downwards acceleration of the falling box (falling) and the non-movement of the ground (resting). The breaking of the box is different because it happens instantaneously, but the EPS should build an encoding of *breaking* nonetheless. Recall that the theory of event segmentation (Zacks et al., 2007) proposes a simple and elegant mechanism in the case of temporal predictions: an encoding is used for as long as the error it produces is within expected bounds. If we assume that we can estimate the expected error of an encoding from the average error of that encoding so far, we can use this as a boundary of an acceptable error, see e.g. Gumbach et al. (2017). This means an error should not be larger than the average historical error plus  $\theta$  standard deviations  $\sigma$  to allow room for variance.

$$averageError + \theta\sigma$$

Once the error exceeds these bounds, the pattern is taken to be unfit and the encoding is switched for another one that fits. In the case of the ground in figure 5, the encoding of being at rest can become very precise, since the prediction is simple (no change). If the ground started moving, this would be unexpected and the encoding would be switched. This is the mechanism that the EPS uses for *classification*. The encodings of uniform patterns like falling or resting will be called event dynamics models in the following, because they encode the *dynamics* of an event.

Once there are event dynamics models for falling, resting and breaking in figure 5, the EPS also builds encodings of when to switch between these encodings, see also Zacks et al. (2007) as well as Gumbsch et al. (2019), Gumbsch et al. (2017). Otherwise the artificial system would be surprised every time the box starts falling or when it breaks after hitting the ground. These encodings will be called transition models in the following, because they describe transitions between event dynamics models. Note that transition models rely on all types of encodings the EPS builds. To recognize that a box breaks when it hits the ground, the transition model needs to have encodings for the box and the ground (top-down), the spatial relation between the two and falling (before) and breaking (after).

## 7.2 A Computational Description of the Event Predictive System

Now we can formulate what the EPS does at a timestep  $t$ . First it receives sensory information from the world and identifies the objects (top-down encodings) it needs to do prediction on as well as the spatial relations between the objects. If there is a prediction from the last timestep, the *prediction error* can be computed. Note that these prediction errors correspond to predictions from  $t - 1$  to  $t$ , stemming from event dynamics models applied at timestep  $t - 1$ . If the prediction error causes surprise for one of these event dynamics models, the EPS switches the model for a better fitting one and learns a transition model. Once all switches are carried out as necessary, we can assume that the change from  $t - 1$  to  $t$  falls into the categories that the new set of event dynamics models encodes. These models can thus be *updated* with this change. The functions of the EPS can be summarized as in figure 6.

If the EPS builds useful and correct event dynamics models, its overall prediction error *within events* will be minimized, i.e. while the box is falling in figure 6, the box's behavior should be predictable. If, additionally, EPS builds useful and correct transition models, this should also hold for *between events* prediction error. In this case, figure 6 as a whole should be predictable. It is rarely the case that a scene is completely predictable however, physical mechanisms can produce chaotic behavior or agents may choose to behave in unexpected ways. In this case, the prediction error can never be zero and the EPS should ideally minimize overall prediction error as much

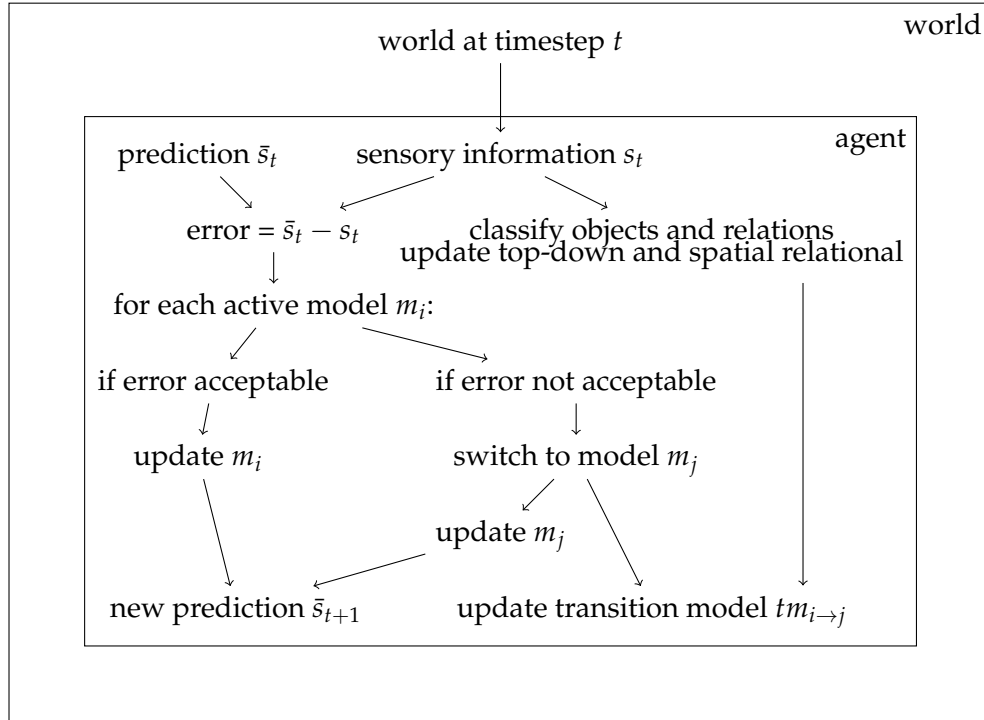


Figure 6: An abstract description of the event predictive system and the functions it has to fulfill.

as possible. Even unpredictable behavior follows a specific distribution though. For example, completely random and unpredictable data would correspond to a *uniform distribution*, as every possible outcome has the same probability. The EPS should thus try to minimize the *average error* in such probabilistic cases. This means that the EPS should minimize the following formula, where  $s_t$  is the sensory information at timestep  $t$ ,  $\bar{s}_t$  is the prediction of  $s_t$  generated by the EPS,  $d(x, y)$  is a difference function and  $T$  is the set of experienced timesteps.

$$\frac{1}{|T|} \sum_{t \in T} d(s_t, \bar{s}_t)$$

Note that this could be extended into the more general formalisms of free energy, see e.g. Friston (2010), but this is beyond the scope of this work.

### 7.3 The Structure of Events

The description of the EPS and how it learns to encode events is supposed to provide *structured* encodings, which we will later map to language. That structure is very minimal however, it only encodes which object the dynamic change is applied to and

thematic role	description
agent	is doing or performing the action
patient	the action is done on the patient or the patient's state changes
experiencer	witnesses the action or experiences an emotion
instrument	the instrument used to perform the action
location	the location at which the action takes place
goal	the goal of the movement or action
path	the path of the movement or action

Table 1: Common thematic roles, Löbner (2012), p. 138

what the environment of that object is like at the time of change. There is much more structure to events however. In language, events are usually described by verbs and their arguments, see e.g. the introduction of Bohnemeyer et al. (2011) or more generally Löbner (2012). The verb itself expresses the action that is done or that happens and it can be extended with arguments to define what is happening in more detail. These arguments can be grouped by what is called their *thematic role* and there are a number of different approaches as to how to do this grouping. There are main groups, however, that are identified by most approaches and are summarized in table 1. Note that these thematic roles refer to the meaning that they add to the event that is described. Identifying such roles does not necessarily say anything about syntactic realization in a specific language.

In theory, such structures can be used to add pre-defined structure to a representation of an event. Instead of using a natural language sentence, an event can be described in such a structured way to avoid the need to translate between a thematic role and its specific implementation in a language. This leads to a more abstract and uniform representation of an event, see also table 2. Note that none of these roles are necessary for all events, even though there always needs to be an object on which the dynamics can unfold. For example, 'I hear a sound' could be grouped as 'experiencer action patient', but 'I scream' should be 'agent action'. This will resurface later when the language system is implemented.

In this work, we avoid using such specific roles to pre-structure our system, i.e. the EPS does not contain any more structure than what was described above. Instead, we focus on what emerges from a general learning mechanism.

#### 7.4 Related Models of Event-Dynamics

With some background established, we can look at how related models approach the task of learning event encodings. Franklin et al. (2020) present a model that operates directly on a distributed representation of a scene. They discuss ways to generate these representations but their model is agnostic as to where they come from. One of the

Agent	Action	Patient	Instrument	Location	Recipient
Waiter	Give	Fish		Fancy_restaurant	John
John	Eat	fish		Fancy_restaurant	

Table 2: An example of structured representations of two consecutive scenes Elman et al., 2019 p. 259

specific representations they apply starts from structured event representations using thematic roles as discussed above. They pre-define these thematic role representations, transform them into a vector, and then pass the vector to their system as input. They then train a neural network model to predict the change from one scene to the next.

A similar approach is taken by Elman et al. (2019). They define a vocabulary of ‘activity components’ and use thematic roles to represent a single scene. In their case, a scene is thus a combination of six thematic roles, each of which can be filled with a component. These scenes are then connected to form meaningful sequences, an example can be seen in table 2. They then train a neural network model on sequences of common activities, such as going to a fancy restaurant. The network tries (and is trained) to predict two things. One is the internal structure of scenes, e.g. that with ‘agent(John)’ and ‘action(eat)’, fish is most probably the patient, but not the location of instrument. The other is the development of sequences of scenes, i.e. that the first example scene from table 2 is followed by the the second. They report that their model can fill in sensible components, for example ‘location(fancy\_restaurant)’ if the input was ‘agent(John)’, ‘action(cut)’ and ‘patient(steak)’. They also report that their model can, to some degree at least, develop a meaningful sequence from a starting scene.

Both Elman et al. (2019) and Franklin et al. (2020) focus on the prediction between strongly pre-structured representations. Gumbsch et al. (2019) present a more low-level approach with less pre-defined structure, see also Gumbsch et al. (2017). Their approach is very similar to the approach in this work in regards to event dynamics models and transition models since this work draws heavily on their modeling architecture. The model operates on the motor system and movement information of a simulated robot, trying to predict limb and body movement from the current state. This produces encodings of motor primitives, e.g. walking or turning. To segment motor primitives, the system triggers model switches once the error exceeds a threshold in the same way described above and learns a transition model. Their transition models further include their own event dynamics model that tries to predict sudden changes in the input that may not be captured by either the old or the new event dynamics model. They argue that this is specifically useful for changes in motor commands, e.g. when changing from ‘walking’ to ‘standing’, since neither the walking model nor the standing model include the strong change in velocity that the switch induces.

Finally, Schrodtt, Kneissler et al. (2017) and Schrodtt, Röhm et al. (2017) implemented an event-predictive model in their original implementation of BrainControl. Brain-



Control provides a base for this work, see below. Their implementation focuses on state-changes of objects that are triggered through collisions. All possible changes are pre-defined and receive unique *tags*. Whenever a collision occurs, the system will link changes in the environment that immediately follow the collision to that collision's context, i.e. a situation encoding. The system thus learns *transition models* rather than event dynamics model. The system then uses these transition models to predict collision effects when simulating into the future.

## 8 Implementation of the Event-Predictive System

In this work, we use a 2-dimensional mini world called *BrainControl* where objects can move in 2-dimensional space, subject to gravity.<sup>29</sup> *BrainControl* is simulated in discrete timesteps, i.e. moving objects are moved a specific distance from one timestep to the next and changes to their internal states are also applied from timestep to timestep. Objects are simulated as axis-parallel rectangles, i.e. each object has a  $(x,y)$  position as well as a width and a height. Width and height are stable, i.e. objects can not change in shape.

While the engine that simulates the world of *BrainControl* is large and has access to a range of different parameters and functions, the model should ideally be more general and only have access to what can plausibly be seen as sensory information. We could get perfect prediction accuracy by using the engine as a model of itself but such an artificial system would only work for *BrainControl*. What we really want is a system that only depends on patterns that also hold for the real world, e.g. the basic laws of physics. The engine of *BrainControl* reproduces these laws in a simplified fashion. It computes forces from three different sources: collisions, gravity and the object's own motor actions. The engine then translates these forces into movement and moves the objects accordingly. It resolves collisions, i.e. it triggers status changes upon collision and ensures that objects do not move into one another if they should not according to game logic. For example, two boxes should not move into one another, but a box may move into and through a bulb. Figure 7 presents an example scene from *BrainControl*.

### 8.1 Sensory Information Available to the System

At any timestep, the EPS has access to the following information *for each object*. Note that in this work the sensory information does not contain any noise.

**id:** a consistent and unique identifier of the object. This identifier allows to track identity over time.

**type:** the *type* of the object, e.g. *green virus* or *red box*.

**(w,h):** the width and height of the object. Recall that objects in *BrainControl* are simulated as rectangles with a fixed width and height.

**(x,y):** absolute position, i.e. relative to the top-left corner of the game world.

**(vx, vy):** velocity in both x-direction (horizontal) and y-direction (vertical).

---

<sup>29</sup>The original implementation of *BrainControl* was done in student projects and was derived from the 2-dimensional Mario World used by Schrodtt, Kneissler et al. (2017). *BrainControl* is publicly available under [github.com/CognitiveModeling/BrainControl](https://github.com/CognitiveModeling/BrainControl).

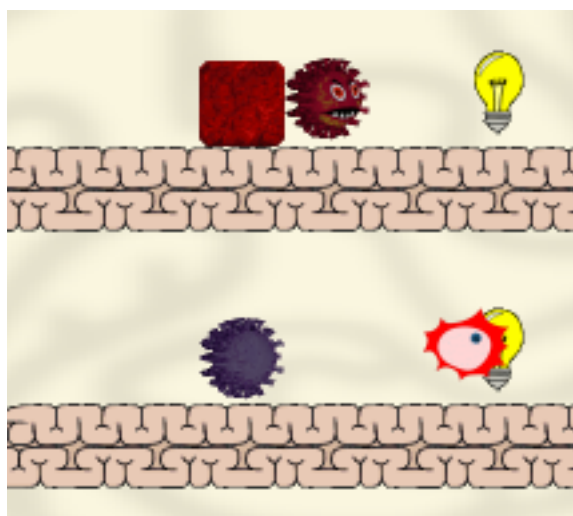


Figure 7: An example scene from BrainControl.

**distances:** current distances to other objects in the scene.

**touching:** other objects that are currently touching the object, including the direction. This direction can be top, bottom, left, right or overlap.

**(h,e):** health and energy of the object. These are internal states that influence an object's status, e.g. if health drops to zero the object is removed by the engine.

**motor commands:** the current motor commands of the object. These are zero if the object cannot move.

While this gives the EPS information that may be considered privileged (internal states) or even unknown (motor commands) in the real world, inferring these states is out of scope for this work.

The EPS is specifically provided id and type of an object, so it does not need to learn object categories and it does not need to track objects through time. This was also done to simplify the task, as object recognition and identity through time are out of scope for this work.

Since id, type, width height cannot change, they are not part of the dynamic object representation. Further, distances and touching objects are not used for event dynamics models, see also figure 6. The remaining pieces of information, i.e. position, velocity, internal states and motor commands, are numeric and can be represented by a vector. We will call these pieces of information *features*. Note that a feature is the type of information, e.g. health, and the health of an object can be represented by a *value*, e.g. 2.5. At any timestep, each object has such a vector containing the current values, we will call this the *feature vector*.

## 8.2 Models, Predictions and Errors

The task of an event dynamics model is to predict the state of an object in the next timestep given the information about the current and earlier timesteps. In this work, this includes change of position, i.e. velocity, and change of internal states, i.e. health and energy. We do not predict actions, i.e. motor commands. To simplify the task, we factorize the event dynamics model, i.e. we train simpler models for each feature separately. This means that there are models that predict  $vx$  and there are different models that predict  $health$ . The simple models receive the *feature vector* as input and have to predict the value of one feature for the next timestep. For this task, we employ linear models.<sup>30</sup> This means the models learn functions of the following kind:

$$\text{predicted value} = b + a_1 * \text{input}_1 + a_2 * \text{input}_2 + \dots + a_n * \text{input}_n$$

Note that the models do not predict the *change* of value, but the actual value. This is helpful in cases where a value changes suddenly, e.g. health is set to zero because the object was forcefully destroyed. Such a change can be easily predicted if the model simply predicts the specific value (0). While linear models are very simple, they suffice in our case. The models are trained by adjusting the *parameters*  $a_1, a_2, \dots, a_n$  as well as  $b$  to minimize error. At every timestep, the predicted value is compared to the actual value and the parameters are adjusted according to the error. In our implementation, we use the *recursive least squares algorithm* to learn the parameters, see e.g. Hayes (1996). The advantage of recursive least squares is its very fast approximation of the ideal parameters. This comes at the cost of a more complex basic algorithm, which is acceptable in this case since the number of parameters is very small. Summing this up, we arrive at a simple formulation of how the basic models are trained, shown in figure 8.

The basic models that are learned by the EPS in practice can be found in appendix D. As an example, the final model describing downwards acceleration through gravitation has the following formula:

$$M_{35} : vy_{t+1} = 0.28 + vy * 1$$

This can be read as ‘the value of vertical velocity  $vy$  in  $t + 1$  will be current  $vy * 1$  plus 0.28’. This describes a constant acceleration downwards. The model *id* is 35. The final model describing change in energy through movement has the following formula:

$$M_8 : e_{t+1} = 0.025 + Mvx * -0.03 + Mvy * 0.006 + Mact * -0.3 + e * 1,$$

$Mvx$  and  $Mvy$  are motor commands for x and y movement and  $Mact$  is the interaction motor command. These are the building blocks of event dynamics models. At

---

<sup>30</sup>The dynamics of the mini world follow linear patterns. Learning non-linear patterns is out of scope of this work.

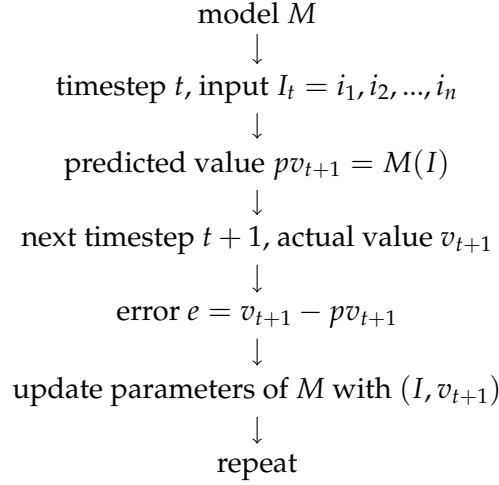


Figure 8: The procedure of prediction and learning for a model.

each timestep, the set of active linear models for an object defines the current event dynamics model.

### 8.3 Model Updates and Surprise

Following Gumbsch et al. (2019), we use a moving Gaussian probability density function to estimate the expected error.<sup>31</sup> Whenever an error is computed, we can check whether the error lies outside of a predefined number of standard deviations, i.e. we say an error is acceptable if the following holds, as discussed above:

$$error < \theta * SD$$

Here,  $SD$  is the standard deviation of the error of the current model and  $\theta$  is a predefined parameter.<sup>32</sup> We use  $\theta = 2$  in our implementation. Our mini world is free of noise and successful models will thus have near perfect performance, as well as quick convergence (see above). We can thus expect that a small cutoff should suffice. Note further that we ignore the mean of the Gaussian. In our case, the models either converge to the actual value or do not converge at all. There are no situations in BrainControl where an event requires non-zero means in the error function. Models in BrainControl will usually not converge to exactly zero because of rounding issues,

<sup>31</sup>A Gaussian distribution is, intuitively speaking, a distribution where the probability density is shaped like a bell curve around the mean. Its probability density function is  $f(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$ , where  $\mu$  is the mean and  $\sigma$  is the *standard deviation*, denoting how much the function spreads around the mean. Gaussian distributions are also known as *normal* distributions.

<sup>32</sup>The standard deviation can be defined as the square root of the variance, where the variance is  $\frac{1}{n-1} \sum_{i=1}^n (x_i - mean(x))^2$ .

so we set an artificial minimal value as a lower bound of the error function. Values smaller than this lower bound are considered equal to zero in most cases.

This way we estimate whether a model's error should be considered acceptable or surprising. In case an error is acceptable, the model gets updated as described above. If the error is considered to be too large, a model switch is initiated. For this, all known models try predicting the current value from the last timesteps feature vector, i.e. do exactly what the failed model also tried. The model that shows the lowest error is then chosen to *replace* the failed model and updated in its place. The parameter update with  $(I, v_{n+1})$  is done for the new model, not for the old one. This becomes especially important when trying to predict switches between models.

To update the error estimate, we use the error *after* updating with  $(I, v_{n+1})$ , i.e. the difference between  $v_{n+1}$  and what the updated model would predict given  $I$ . Ideally, the error should reflect the performance of the *current* model over all historic data seen so far, but to ensure this, all historic data would have to be stored. Let us assume that the current model  $M$  was active for  $k$  timesteps, being updated at each timestep. Now after  $k$  timesteps, the current model is indeed an updated model  $M^k$ . If  $(I_1, v_2), (I_2, v_3), \dots, (I_k, v_{k+1})$  was the stream of sensory inputs, we would ideally want to know the average error of  $M^k$  on all this data. That is

$$\text{Mean}(\text{Ideal Error}) = \frac{1}{k} \sum_{i=1}^k M^k(I_i) - v_{i+1}$$

where  $M^k(I_i)$  is the prediction of model  $M^k$  given sensory information  $I_i$ . Since this historic information is not stored, we approximate this as follows, see also Franklin et al., 2020 for a discussion of this issue.

$$\text{Mean}(\text{Approximated Error}) = \frac{1}{k} \sum_{i=1}^k M^i(I_i) - v_{i+1}$$

In cases where no known model sufficiently satisfies the sensory information, a new model is created and initialized with zeros. It is then updated with  $(I, v_{n+1})$ , which is the first real update for the model and its initial error is then the error after that update.

## 8.4 Transition Models

To predict switches of models, we employ transition models, as discussed above. In the BrainControl world, switches in behavior stem either from the activation of a motor command, the application of some external force or some internal state reaching a critical value, like health hitting zero. External forces in the BrainControl world are always collisions, there is no wind for example. Coincidental transitions are impossible. A transition model should be given information about collisions and the

current internal state of the object the transition applies to. A transition model should thus encode the following information.

**old model, new model:** the basic models that were switched, i.e. this transition model will encode the switching from *old model* to *new model*.

**main object type:** the type of the object the transition applies to, i.e. the type of the object for which *old model* was switched to *new model*.

**main object features:** an average feature vector for the main object.

**touching objects:** objects that are touching the main object at time of transition.

**touching objects features:** a list of features of the touching objects that are relevant to the transition.

This is how transition models are implemented for this work. Object type and model ids are set once and do not change afterwards, i.e. we learn one transition model per combination of object type and models. Feature vectors have numerical values, so we use Gaussians again to learn a probability distribution. The updating of the features for the main object is shown in 9. In that example, a transition model has learned to predict when a green virus will stop moving in order to rest, which depends on its energy level. The Gaussian for energy will thus converge on that level, whereas the Gaussian for health will broaden, as the health value is irrelevant for this transition. Note that the transition is not always updated on the same *object* but only of objects of the same *type*. This is safe to do since we assume that objects of the same type consistently behave the same way.

Recall from the discussion above and figure 6 that transition models use spatial relational encodings. This is implemented by learning patterns of touching objects, we dismiss objects that are more distant because they cannot affect the features of the main object. We pre-structure spatial relational encodings by defining possible touch-directions (top, bottom, left, right, overlap) and by only including the motor commands of those touching objects while ignoring other features. This pre-structuring is necessary because relations are not numerical, i.e. there is no immediate way of computing similarity, so we help the EPS with additional structure.

An example of transition that depends on the context would be a ball being pushed by a green virus or by a spiky red cell. The result will be mutual slower movement. If the resulting movement is the same, the same transition model will be triggered upon being pushed by another ball as well as by a box. We assume that if two objects trigger the same transition through collision, then their states will be centered around one value for those transitions. This means we can use a single Gaussian for each of the features of the object that pushes the ball from the defined direction. This is illustrated in figure 10.

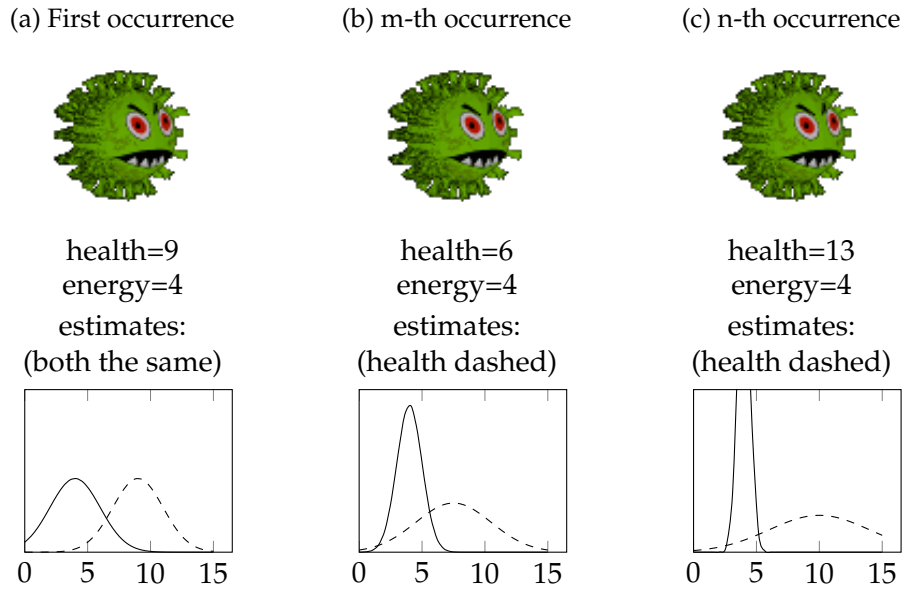


Figure 9: Learning the dependency of a transition on an internal state. The green virus stops moving when reaching energy level 4 to rest. In all cases, the transition is from the ‘moves-model’ to the ‘at-rest-model’. The Gaussian for health (dashed) has a large standard deviation after the n-th occurrence, the Gaussian for energy (solid) becomes very narrow.

Note that object type and collision direction are not numerical in the same sense. The transition above is triggered by a virus as well as a spiky cell, let us say, coming from the right, making the ball they collide with move left. Let us call this transition the *being-pushed transition*. In this implementation, we capture variance in the types of other objects by using sets of possible objects that trigger the transition. These sets are built for each of the different directions. In this case, the direction *right* would thus have a set that contains virus and cell. Whenever the transition occurs, the types of the currently touching objects are added to the sets matching the direction of touch. Further, there is a special *none-type* that signals that this direction was *empty*. Otherwise, we can not learn whether a direction is necessary or optional. In the example of the being-pushed transition, the transition is mostly taking place on the ground, so there usually is another collision registered, namely with the ground. But the ground is not a necessary component of being pushed, the ball could also be pushed while falling. Thus, when the transition is seen while falling and there is no ground beneath, the special *none-type* is added in direction *bottom*. All this can be seen in figure 10.

In this work, we learn and update transition models not only when there was an actual model *switch*, but also if a model stays. Such transition models thus encode the context of an event being active.

This concludes the description of the transition models we apply. Their perfor-



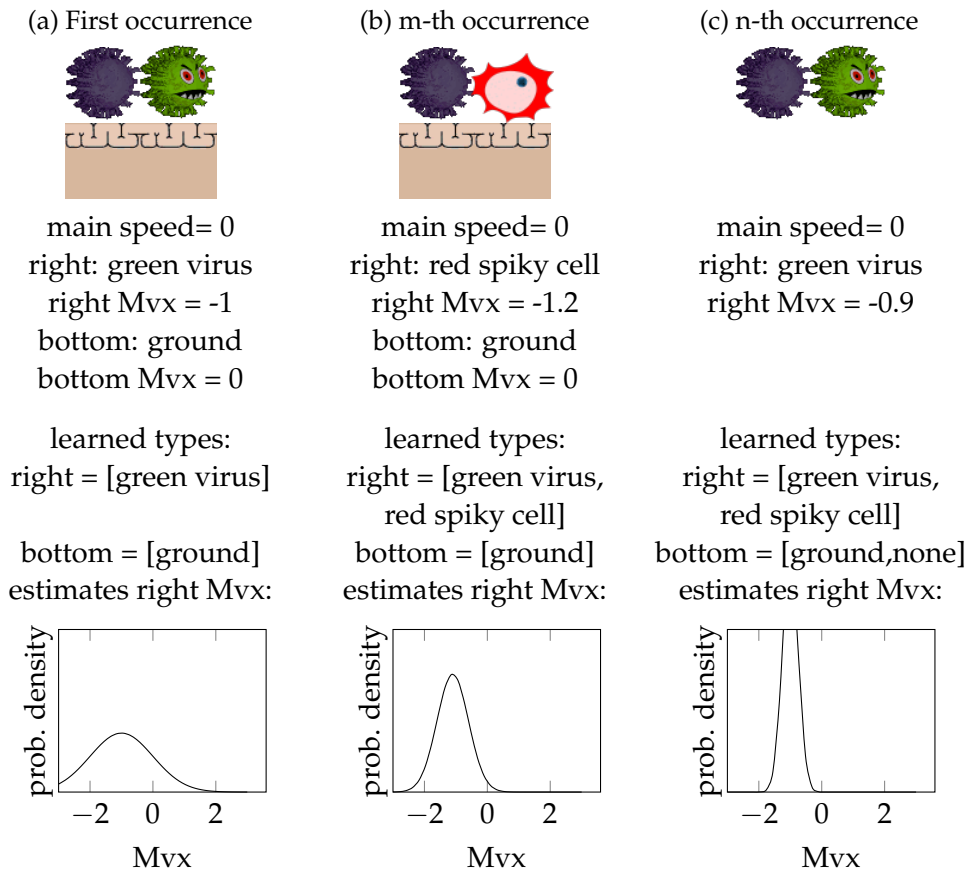


Figure 10: Learning the dependency of a transition on the surroundings. The transition refers to the ball on the left, which changes from being at rest to moving left. This shows (symbolically) the first occurrence on the left and a later m-th as well as an even later n-th occurrence. Only a few features and sets are shown. E.g. in the third occurrence the ball is falling for the first time so the y-speed feature would now also be updated to include a bigger range. The sets refer to the sets of objects learned, here only the sets for direction *right* and *bottom* are shown. For features, only the estimate for the *speed* of the object touching from the *right* is shown.

mance will be evaluated and discussed in section 9. In summary, they learn Gaussian estimates for the features of the main object as well as touching objects. Touching objects are grouped by the direction of touch. For each direction we keep a set of possible types that could collide from that direction to trigger the transition. If a collision from a certain direction is not necessary to trigger the transition, a special *none-type* is registered for that direction.

## 8.5 Relation Models

We implement relation models that capture an abstract relation between objects, i.e. how their distance is changing, whether they touch each other and if they do, from which direction. In the case of touching, these relation models are equivalent to the touching objects patterns that transition models learn.

Relation transition models are then learned to encode transitions between such relation models. The models do not play a role in the EPS beyond this, they are not updated further or used for prediction otherwise. They will, however, be used as encodings when processing language.

## 8.6 Summary of the Implementation

This concludes the algorithmic level description of the models applied by the EPS. Note that this implements the processes described in figure 6, i.e. how basic models are updated and switched. Note that the EPS does not bundle basic models that occur together into a combined model directly. A scene encoding, and thus an event dynamics model, is defined by the basic models that are active together. The behavior is illustrated in figure 11 for summary.

If we try to map the implemented parts in our system with the theoretical parts described in section 7, we end up with figure 12. In theory, all the different encodings are highly interdependent, predicting each other. This is something that our system cautiously approaches. We will see, however, that the patterns this system learns already lead to sensible structures that can be used to link to natural language utterances.

## 8.7 Implementational Details

There are a few peculiarities of the implementation that are worth pointing out. The first concerns collisions. Because of the discrete nature of movement in BrainControl, objects usually collide before completing their movement, as shown in figure 13. The change in actual position is therefore reduced for the timestep where the collision happens. This collision-dependent shortening would need a sophisticated and specialized model to compute. So instead of distance traveled, the sensory input contains actual speed, i.e. the movement as if there was no collision.<sup>33</sup> On an intuitive level, we assume that objects do not reduce their speed before impact but keep going until actual impact with the same speed. In predicting movement speed and forces, we are interested in this actual speed.

---

<sup>33</sup>Even for the engine, this is a more complex matter. Especially when two objects are moving towards each other.


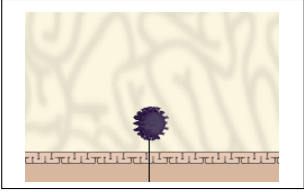
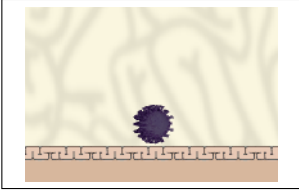
(a) Abstract Model	(b) Example at $t$	(c) Example at $t + 1$
		
<b>Sensory Info</b> position non-collision speed type and unique ID internal states map: dir $\rightarrow$ otherID (dir $\in [t, b, l, r, o]$ )	<b>Sensory Info</b> (136, 183.8) (0, 5.04) ball, 19 health: 9 map: dir $\rightarrow$ otherID empty for all	<b>Sensory Info</b> (136, 188) (0, 5.32) ball, 19 health: 9 bottom $\rightarrow$ <i>Obj</i> <i>Obj</i> is ground tile
<b>Predictive Model</b> predicted feature $f: I_t \rightarrow pv_{t+1}$ $f \sim b + a_1x_1 + \dots$ average error SD	$M_{35}$ vertical speed $f(I_t) = 5.32$ $f(I) = 0.28 + vy$ 0.0	$M_4$ vertical speed $f(I_{t+1}) = 0$ $f(I) = 0$ 0.0
<b>Transition Model</b> switch = $M_i \rightarrow M_j$ gaussian for each feat for each direction: set of possible types gaussian for each feat	$TM_{275}$ switch = $M_{35} \rightarrow M_{35}$ gaussian for each feat bottom none $\emptyset$	$TM_{285}$ switch = $M_{35} \rightarrow M_4$ gaussian for each feat bottom [ground, platform, ...] zero for all feats
<b>Relation Models</b> for each related <i>Obj</i> : $\Delta dist \in [-, 0, +]$ touch $\in [yes, no]$ direction $\in [t, b, l, r, o]$	<b>Relation Models</b> for <i>Obj</i> (ground) $\Delta dist = -$ touch = no direction = $\emptyset$	<b>Relation Models</b> for <i>Obj</i> (ground) $\Delta dist = 0$ touch = yes direction = bottom

Figure 11: A summary of the implemented structures in the EPS. *Feature* is abbreviated *feat*, all other entries should be clear from the text.

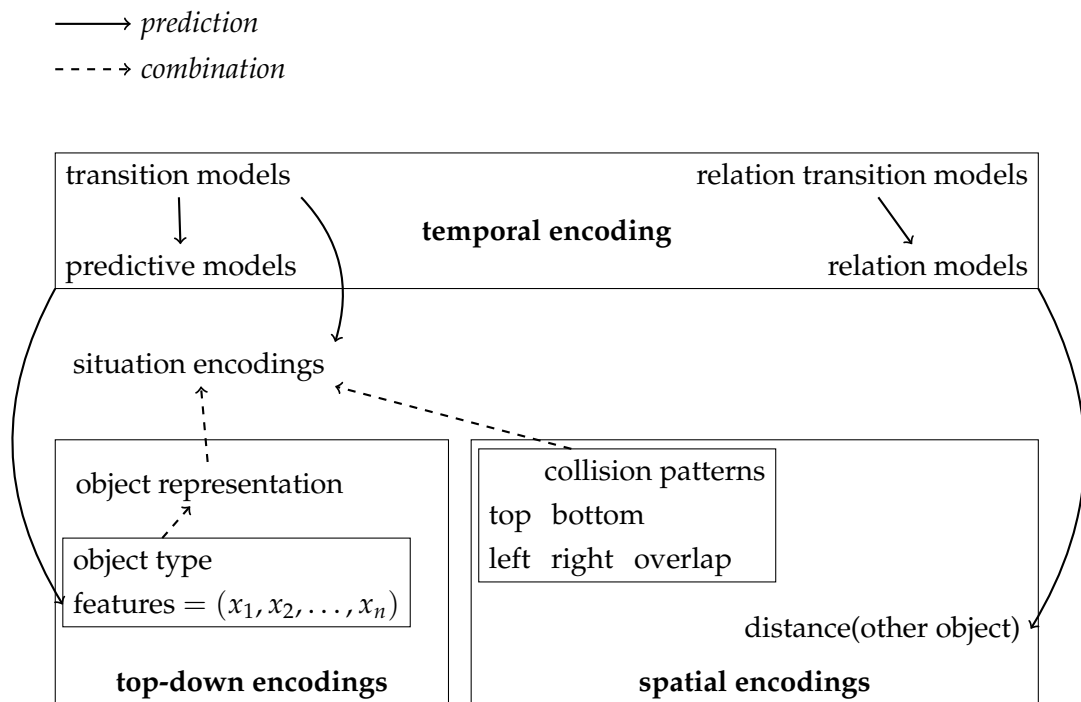


Figure 12: The parts that are implemented in this work and how they are linked to the structures proposed in theory.

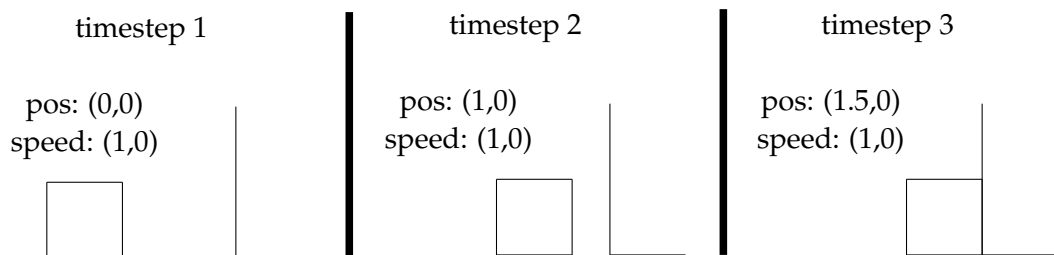


Figure 13: A box moving with a constant speed of 1 in the horizontal and 0 in the vertical. In timestep 3, this movement is stopped by a collision. While the supposed speed is still (1,0) according to the forces at the beginning of the movement, the actual movement is (0.5,0).

A second point concerns the implementation of transition model updates.<sup>34</sup> Consider the system at the end of some timestep  $t$ . The active models were just checked and updated and one of the models was switched, let us say  $M_1$  was replaced with  $M_2$ . Now,  $M_2$  was updated with the sensory input of the *last* timestep and the actual value of this current timestep, i.e. with  $(I_{t-1}, v_t)$ . The transition model  $TM_1 : M_1 \rightarrow M_2$  is thus also trained on the features and collisions at timestep  $t - 1$ ! We train transition models on the context of the *last* timestep because we also update the models for the context of the last timestep. To make this clear, imagine a box falling off a platform. In  $t - 1$ , the box just moved over the edge and is now touching no object but also not yet falling. This is specific to the BrainControl engine, which computes gravity first and then moves objects, recomputing gravity at the start of the next timestep. From  $t - 1$  to  $t$ , the engine realizes that the box is not on solid ground anymore and gravity starts to pull the box down. The engine thus increases the vertical speed of the object. The learning system thus realizes that from  $t - 1$  to  $t$ ,  $M_1$  (not falling) was indeed wrong but  $M_2$  (falling) is a better fit.  $M_1$  was the right model for going from  $t - 2$  to  $t - 1$  and  $M_2$  is the right model to go from  $t - 1$  to  $t$ . The transition  $TM_1 : M_1 \rightarrow M_2$  thus happened in  $t - 1$  and especially in the context of  $t - 1$ .

A final point concerns *destruction*. In BrainControl, objects can be destroyed under certain circumstances, but this will always include their health being zero before they are removed from the game by the engine. We implement a mechanism that keeps those objects in the domain of the EPS for 30 timesteps after they are removed from the game. The EPS then learns that their health is zero, but does not predict any other features anymore. This enables the EPS to learn a transition to the zero-health model, which allows the prediction of breaking.

---

<sup>34</sup>This might seem a small technicality but may be easily missed on re-implementation. In a system like BrainControl with very exact parameters, missing it will make transition models fail beyond rescue. It thus seems worth it to quickly explain this slightly confusing detail.

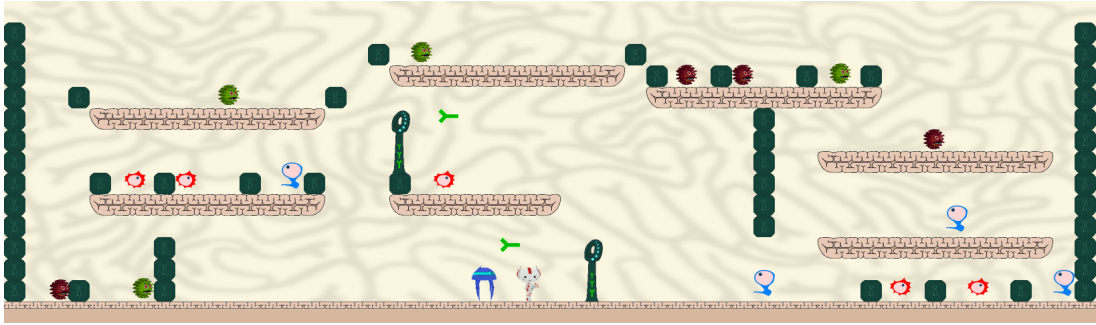


Figure 14: Level 1, used for training

## 9 Evaluation and Discussion of the Implementation

To evaluate the EPS as it is implemented here, we look at how the EPS minimizes overall prediction error and whether specific events can be encoded after training. After that, we will discuss limitations and possible improvements. For reproducible testing, we employ a training scheme using four different setups of the mini world. The first setup contains all moving entities but only allows sideways movement. It can be seen in figure 14 and will be called *level 1*. Note that all levels contain two robot entities in the center, these do not move and their features are not predicted. They represent the player and LEARNA. The levels contain five different objects that move by themselves, a blue tall cell, a red spiky cell, a green and a red virus (those have faces) and green flying arrows. All other objects do not move by themselves.

The moving objects are controlled by simple rules. They move left or right randomly, rest if their energy is low, attack hostile objects and flee if their health is low and there is a hostile object nearby. They consume bulbs and wrenches if there is no hostile object around. Cells and viruses are hostile towards the other group, but not within, i.e. a cell attacks a virus but not another cell. Training level 2 includes consumable bulbs as well as boxes and balls that can be pushed off platforms, this is shown in figure 15. Note that entities will avoid falling off platforms, i.e. turn around at the edge. Training level 3 includes consumable wrenches and all combinations of entities being hostile towards one another are situated on a shared platform, leading to aggression. This can be seen in figure 16.

Finally, training level 4 is built in a way to allow falling off platforms and moving objects are controlled in a way that they will drop off, i.e. they will not stop and turn as in the other levels. This is specific to level 4, which can be seen in figure 17.

Training was done on a pre-defined sequence of these training levels. 60 levels were run for a total of 41750 timesteps.<sup>35</sup> Appendix D contains a list of all basic models that were learned.

<sup>35</sup>The exact training sequence was 1, 4, 2, 3, 1, 2, 2, 4, 4, 2, 2, 4, 4, 4, 2, 4, 3, 3, 3, 3, 3, 3, 1, 2, 4, 3, followed by 8 repetitions of the sequence 1, 2, 3, 4

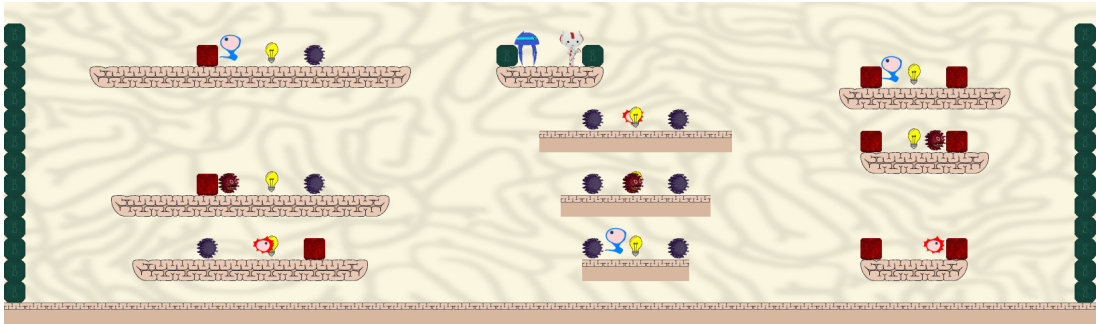


Figure 15: Level 2, used for training

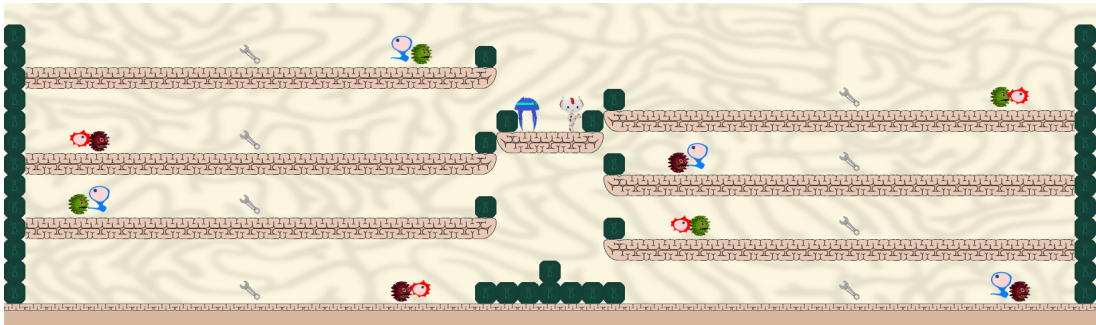


Figure 16: Level 3, used for training

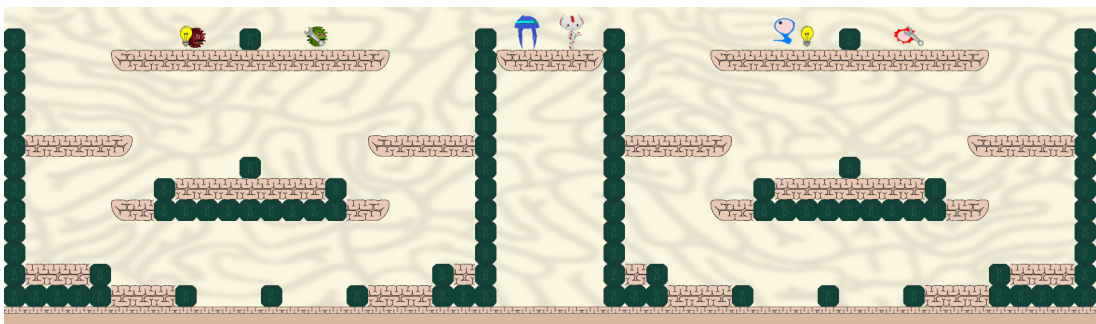


Figure 17: Level 4, used for training

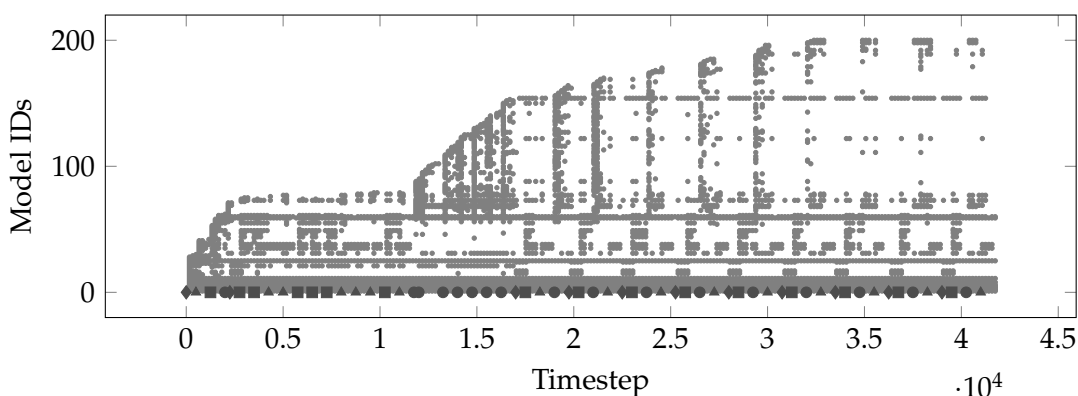


Figure 18: Model use over time. The symbols at the bottom indicate that a training level was started at that timestep: diamond is level 1, square is level 2, circle is level 3, triangle is level 4.

### 9.1 Overall Error Minimization

New models were learned until late in training, as can be seen in figure 18, showing which models were used over time. This can be mostly attributed to level 3, where entities attack each other. As can be seen in the full list of models in appendix D—models with IDs of 74 and larger try to model a change in health—which changes due to objects attacking each other. Other dynamics are learned much quicker. Note how at the onset of training level 3, a large range of models is always activated until the correct stable models are learned around timestep 33000 (see also figure 19 below), making the unsuccessful models obsolete.

This learning delay of health models can also be seen in figure 19 where the average standard deviation (SD) of active models is shown. At the onset of training level 3, average SD rises quickly and remains high for the complete level. This only ceases at around timestep 33000, indicating that the correct models were found. We can assume that from this point onward, the basic predictive models were fully learned, i.e. there is no noticeable within-event error anymore.

The between-events prediction on the other hand is more difficult and produces decreasing but noticeable error till the end of training. This can be seen in figure 20, which shows average errors across active models. This includes errors that stem from failing to predict a transition, i.e. this describes the overall error the EPS produces.

One common cause of such errors are cases where the Gaussian for a feature is slightly too restrictive. For example, the transition from being stable (vertical velocity being zero) to falling for red viruses is learned correctly, as seen in table 3, which shows the respective transition models. The spatial relational encoding for staying level requires footing for direction *bottom*, whereas the spatial relational encoding for the transition to falling requires that direction to be empty. Nonetheless, the errors that



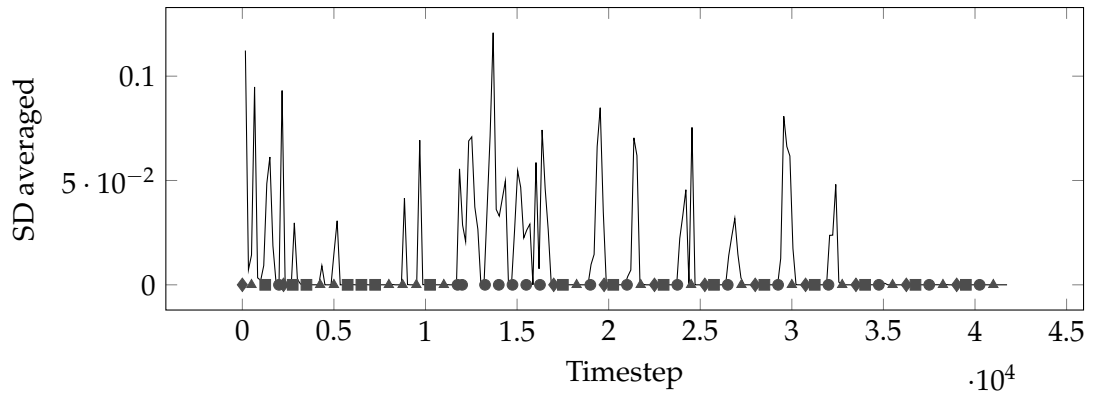


Figure 19: Average SD of models in use. This is a good indicator of when new models are created. The symbols at the bottom indicate that a training level was started at that timestep: diamond is level 1, square is level 2, circle is level 3, triangle is level 4.

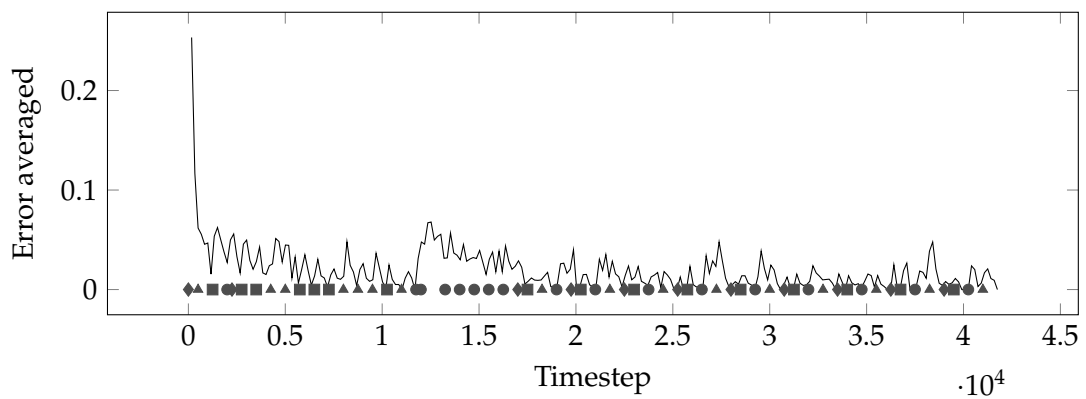


Figure 20: Average error of models in use. The symbols at the bottom indicate that a training level was started at that timestep: diamond is level 1, square is level 2, circle is level 3, triangle is level 4.

Table 3: The transition models for not moving in the vertical (69) and starting to fall (244). *From* and *to* describe the models that the transition applies to, *t*: is the number of times this transition model was applied successfully. The rest describes the situation encoding, i.e. the features of the main object as well as sets of types and features for objects of the different possible directions. If a direction only contains the none-type, it is not shown.

ID	Transition Model
69	from: 4, to: 4, t: 112078, main object: red virus, main features: vx: 0.001~1.83, Mxdir: -0.013~0.822, Mvx: 0.69~0.75, Mact: 0.048~0.121, h: 8.066~3.959, e: 6.843~2.282, LEFT:{types: none, iron block1, movable box, virus ball, cell long blue, cell spikes red, features:DX: -0~0, DTOTAL: 0~0, Mxdir: 0.179~0.403, Mvx: 0.011~0.133, Mact: 0.01~0.023}, RIGHT:{types: none, iron block1, movable box, virus ball, cell long blue, cell spikes red, features:DX: 0~0, DTOTAL: 0~0, Mxdir: -0.237~0.445, Mvx: 0.013~0.142, Mact: 0.013~0.026}, BOTTOM:{types: iron block1, top left wrench head, top center wrench head, top right wrench head, top dirt border ground}, OVERLAP:{types: none, wrench, bulb flower}
244	from: 4, to: 35, t: 92, main object: red virus, main features: vx: 0.022~2.342, Mxdir: 0.034~0.998, Mvx: 1.239~0.413, Mact: 0.049~0.029, h: 8.59~2.316, e: 8.219~1.954,

show up towards the end of the training often include these transitions, i.e. they were not applied correctly. This can be caused by restrictive Gaussians as the one for energy of transition model 244. With  $\theta = 2$ , the maximum acceptable value for the feature energy would be  $8.219 + 2 * 1.954 = 12.127$ . The actual maximum for energy is around 13 however. This means that there *will be* cases where this transition model fails to predict correctly, but these cases require peripheral values for energy or health. Since such errors (stemming from peripheral feature values) do not hinder simulation later, we ignore this issue in this work and accept occasional errors in transition models.

## 9.2 Learned Models

To see which models become stable during training, see the table in appendix D. Stable models will have been experienced much more often than the unsuccessful ones. During training these failed attempts are not invoked anymore after the correct model is learned. These failed attempts may be explained by overfitting as those models learn dependencies on features that are actually independent, but the model still fits the formula to their specific values during training. This usually happens if there is little variance in the specific feature value. Once the model is applied to another object,

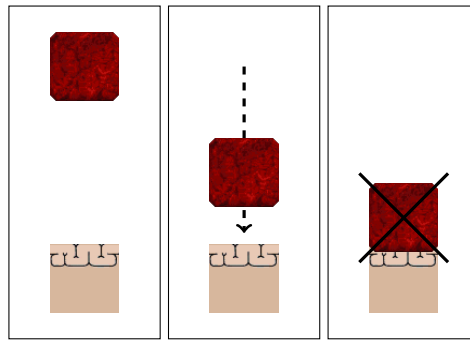


Figure 21: Event of a box falling and breaking

for example, and those feature values are now different, the model is wrong. In the following, we will examine specific stable models and their transition models.

### **A Box Falling and Breaking**

When a box falls, at the start of the event the model for vertical speed changes from being at rest to falling. During the event, the box falls and at the end, if the box hits the ground, it breaks, meaning its model for health switches to the zero model. The event is illustrated in figure 21. The models involved in this are listed in table 4.

### **A Cell Pushing a Ball**

When a cell pushes a ball, the cell's model for movement changes from normal movement to pushing at the start of the event. During the event, the ball falls and at the end, if the ball hits the ground, it breaks, meaning its model for health switches to the zero model. The event is illustrated in figure 22. The models involved in this are listed in table 5.

### **A Green Arrow Flying**

A flying arrow is difficult to visualize, but figure 23 symbolizes a green arrow flying. Flying is learned through its movement model, which is shown in table 6.

### **A Cell Attacking and Fleeing**

Figure 24 shows a blue tall cell attacking a virus and fleeing the scene afterwards. This event has three parts, i.e. first the attacking movement, then the actual fighting and finally the fleeing, each using different movement models. The cell further loses health while attacking, which is captured in a different model. The models involved in this are listed in table 7.

Table 4: Transition models for a box falling and breaking.  $TM_{264}$  encodes the context of being on stable ground, i.e. not moving in the vertical  $M_4$ .  $TM_{281}$  encodes the switch to falling  $M_{35}$ ,  $TM_{283}$  the context of falling. Note that the switch occurs when there is no vertical movement yet. Finally  $TM_{265}$  encodes the context of the no-change-in-health model  $M_5$ .  $TM_{1806}$  encodes the switch to the being-broken model  $M_{154}$ , i.e. health is zero. This switch thus encodes the context of breaking, requiring specific ground below *and* larger downwards speed ( $v_y > 7$ ). This thus encodes somewhat implicitly that the object has to be *falling* in order to break. The context of being broken,  $TM_{1807}$ , is empty, since there is no context for broken objects, they are removed shortly after breaking.

ID	Transition Model
264	from: 4, to: 4, t: 20298, main object: movable box, main features: vx: -0.076~0.59, h: 8.865~2.221, LEFT:{types: none, cell long blue, red virus, cell spikes red, features:Mxdir: 0.88~0.354, Mvx: 1.09~0.522, Mact: 0.045~0.031}, RIGHT:{types: none, cell long blue, red virus, cell spikes red, features:DX: 0~0, DTOTAL: 0~0, Mxdir: -0.925~0.29, Mvx: 1.197~0.506, Mact: 0.045~0.03}, BOTTOM:{types: top left wrench head, top center wrench head, top right wrench head}
281	from: 4, to: 35, t: 141, main object: movable box, main features: vx: -0.145~1.191, h: 8.583~2.255, , LEFT:{types: none, cell long blue, red virus, cell spikes red, features:Mxdir: 0.924~0.364, Mvx: 1.204~0.4, Mact: 0.052~0.029}, RIGHT:{types: none, cell long blue, red virus, cell spikes red, features:Mxdir: -1.001~0.056, Mvx: 1.297~0.415, Mact: 0.055~0.028}
283	from: 35, to: 35, t: 3442, main object: movable box, main features: vx: -0.108~0.687, vy: 3.923~2.472, h: 8.567~2.256, LEFT:{types: none, cell long blue, red virus, cell spikes red, features:Mxdir: 0.438~0.898, Mvx: 1.196~0.373, Mact: 0.051~0.029}, RIGHT:{types: none, cell long blue, red virus, cell spikes red, features:DX: 0~0, DTOTAL: 0~0, Mxdir: -0.384~0.924, Mvx: 1.253~0.372, Mact: 0.049~0.029}
265	from: 5, to: 5, t: 17900, main object: movable box, main features: vx: -0.019~0.247, vy: 0.728~1.885, h: 8.883~2.242, LEFT:{types: none, cell spikes red, features:Mxdir: -0.018~0.134, Mvx: 0.017~0.126, Mact: 0.001~0.007}, RIGHT:{types: none, cell long blue, red virus, features:Mxdir: 0.017~0.13, Mvx: 0.019~0.146, Mact: 0.001~0.009}, BOTTOM:{types: none, top left wrench head, top center wrench head, top right wrench head}
1806	from: 5, to: 154, t: 79, main object: movable box, main features: vx: -0.004~0.385, vy: 7.096~2.232, h: 8.67~2.321, BOT- TOM:{types: top dirt border ground}
1807	from: 154, to: 154, t: 4819, main object: movable box

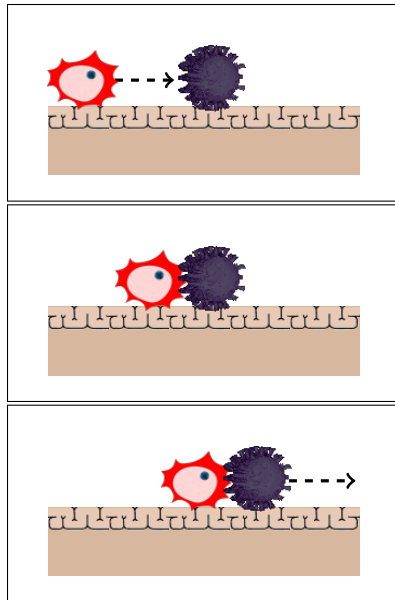


Figure 22: Event of a cell pushing a ball

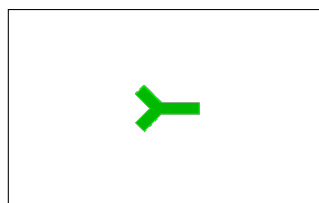


Figure 23: Symbolic illustration of a flying arrow

Table 5: Transition models for a cell pushing a ball.  $TM_{125}$  encodes the context of normal movement (model  $M_{25}$ ),  $TM_{516}$  the switch to the *pushing* model  $M_{55}$  and  $TM_{365}$  the context of pushing.  $TM_{395}$  on the other hand encodes the switch from being at rest to *being pushed* ( $M_{61}$ ) for the ball, note the necessary animate entities on the left having positive movement force (Mvx 1.3).  $TM_{388}$  encodes the context of being pushed.

ID	Transition Model
122	from: 25, to: 25, t: 23626, main object: cell spikes red, main features: vx: 2.25~0.259, Mxdir: 1~0.057, Mvx: 1.001~0.115, Mact: 0.05~0.029, h: 11.267~3.111, e: 7.381~2.003, LEFT:{types: none, green virus, features:Mxdir: 0.867~0.499, Mvx: 2.072~0.518, Mact: 0.04~0.024}, BOTTOM:{types: iron block1, none, top left wrench head, top center wrench head, top right wrench head, top dirt border ground}, OVERLAP:{types: none, wrench, bulb flower}
516	from: 25, to: 55, t: 41, main object: cell spikes red, main features: vx: 2.257~0.248, Mxdir: 1.008~0.058, Mvx: 0.997~0.116, Mact: 0.048~0.031, h: 9.413~2.15, e: 8.728~2.726, RIGHT:{types: movable box, virus ball}, BOTTOM:{types: top center wrench head, top right wrench head, top dirt border ground}
365	from: 55, to: 55, t: 1435, main object: cell spikes red, main features: vx: 1.5~0, Mxdir: 1~0.059, Mvx: 1.278~0.403, Mact: 0.05~0.029, h: 9.251~2.246, e: 8.594~2.728, RIGHT:{types: movable box, virus ball}, BOTTOM:{types: top center wrench head, top right wrench head, top dirt border ground}
395	from: 2, to: 61, t: 99, main object: virus ball, main features: h: 9.114~2.21, LEFT:{types: cell long blue, red virus, cell spikes red, features:Mxdir: 0.996~0.054, Mvx: 1.285~0.393, Mact: 0.05~0.029}, BOTTOM:{types: top center wrench head, top right wrench head, top dirt border ground}
388	from: 61, to: 61, t: 2487, main object: virus ball, main features: vx: 1.2~0, h: 8.998~2.059, LEFT:{types: cell long blue, red virus, cell spikes red, features:Mxdir: 0.998~0.058, Mvx: 1.331~0.416, Mact: 0.049~0.029}, BOTTOM:{types: top center wrench head, top right wrench head, top dirt border ground}

Table 6: Transition model for a green arrow flying.  $TM_{134}$  encodes the situations in which the flying model  $M_{24}$  is active. Note that this is one of the only situations where upwards movement is active ( $M_{vy}$  1).

ID	Transition Model
134	from: 24, to: 24, t: 11260, main object: bullet willy, main features: vx: 4.496~0.52, Mxdir: 1~0.057, Mvx: 0.999~0.116, Mvy: 1~0, h: 8.916~2.252, e: 9.159~2.28,

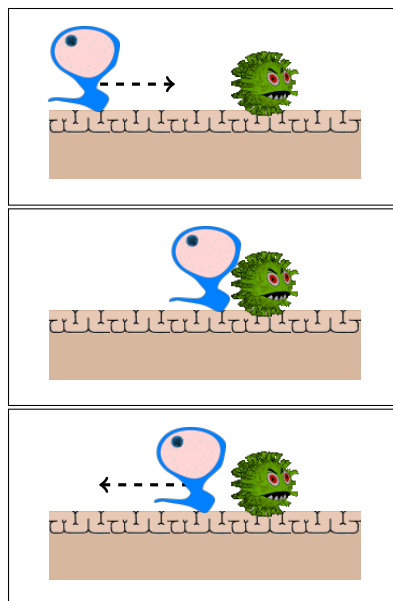


Figure 24: Event of a cell attacking a virus and fleeing

Table 7: Transition models for a cell attacking and fleeing.  $TM_{377}$  describes the context of the attacking move ( $M_{60}$ ), moving with high force ( $Mvx$  1.8).  $TM_{405}$  describes the change to the not-moving model  $M_2$ , which in this case is activated because the two objects push one another into a stalemate.  $TM_{653}$  then describes the switch to the fleeing model  $M_{69}$  at low health (h 1).  $TM_{2808}$  describes the context of the loosing-health model  $M_{198}$

ID	Transition Model
377	from: 60, to: 60, t: 4705, main object: cell long blue, main features: vx: 2.426~0.236, Mxdir: 1~0.058, Mvx: 1.797~0.172, Mact: 0.05~0.029, h: 8.655~2.535, e: 6.936~2.417, RIGHT:{types: none, green virus, features:Mxdir: 1.008~0.06, Mvx: 0.982~0.117, Mact: 0.048~0.029}, BOTTOM:{types: iron block1, top left wrench head, top center wrench head, top right wrench head, top dirt border ground}, OVERLAP:{types: none, wrench, bulb flower}
405	from: 60, to: 2, t: 104, main object: cell long blue, main features: vx: 2.439~0.219, Mxdir: 1.005~0.057, Mvx: 0.018~0.002, Mact: 0.055~0.029, h: 8.536~2.488, e: 7.313~2.342, RIGHT:{types: iron block1, red virus, green virus, features:Mxdir: -0.358~0.483, Mvx: 0.006~0.009, Mact: 0.016~0.029}, BOTTOM:{types: iron block1, top center wrench head, top right wrench head, top dirt border ground}
653	from: 2, to: 69, t: 37, main object: cell long blue, main features: Mxdir: -0.996~0.058, Mvx: 2.897~0.525, Mact: 0.047~0.031, h: 1.024~0.193, e: 8.753~2.83, LEFT:{types: none, iron block1, green virus, features:Mxdir: -0.486~0.486, Mvx: 0.495~0.495, Mact: 0.022~0.022}, RIGHT:{types: none, red virus, green virus, features:Mxdir: -0.943~0.351, Mvx: 1.726~0.362, Mact: 0.048~0.031}, BOTTOM:{types: top center wrench head, top dirt border ground}
2808	from: 198, to: 198, t: 1132, main object: cell long blue, main features: vx: 0.042~0.572, Mxdir: -0.01~1.002, Mvx: 0.045~0.263, Mact: 0.049~0.028, h: 4.47~3.019, e: 8.61~2.128, LEFT:{types: none, iron block1, green virus, red virus, features:DX: -0~0, DTOTAL: 0~0, Mxdir: 0.912~0.333, Mvx: 0.018~0.033, Mact: 0.048~0.03}, RIGHT:{types: none, iron block1, green virus, red virus, features:Mxdir: -0.833~0.445, Mvx: 0.035~0.146, Mact: 0.046~0.031}, BOTTOM:{types: iron block1, top center wrench head, top right wrench head, top dirt border ground}



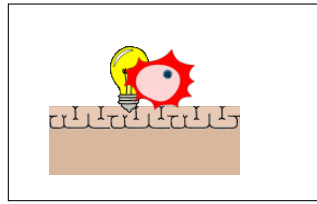


Figure 25: Event of a cell consuming a bulb

### A Cell Consuming a Bulb

Finally, figure 25 shows a cell consuming a bulb, which is difficult to illustrate. Consuming is learned through the onset of the *interaction* motor command of the cell, which triggers an energy gain for the cell. This change in energy models then identifies the event. The relevant transition models are listed in table 8.

### 9.3 Overcoming Limitations

While the EPS successfully learns to predict a range of events as discussed above, there are limitations to the implementation. One major limitation becomes apparent when an object interaction can happen from different directions. For example, if a box was destroyed by a collision from the left *or* the right, the situation encoding of the transition model will fail because it experiences the transition with an empty left some times and an empty right other times. It will thus learn that both collisions from the left *and* from the right are meaningless for that destruction transition. This limitation stems from the context representation chosen here, which will also be discussed below.

Because these context representations only encode other objects that are in direct contact, there cannot be action at a distance. The system cannot learn that a switch opens a door if it is not in direct contact with that door, i.e. physical action at a distance. The system can also not learn that an animate<sup>36</sup> object such as a virus will move *towards* another object for example.

Probably the most obvious limitation is the linearity of the predictive models and an intuitive extension would be to use more complex models that can fit more complex patterns. These could be polynomial functions of higher order, for example quadratic or cubic models, but also neural networks models. Since the focus of this work is the emerging structure rather than low-level model learning however, linear models seem justified.

Lastly, sudden changes constitute a limitation, e.g. if an object was bouncing off of another object. This *reversal* in speed would only produce data at one timestep. In this case the probability of under-fitting the model and ending up with a somewhat random model would be very high. Gumbsch et al., 2019 for example train such short

---

<sup>36</sup>Animate in the sense that the object actually exerts movement forces in the game.

Table 8: Transition models for the cell consuming a bulb. Note that the order of TM IDs is not following the event sequence here.  $TM_{535}$  encodes the context for switching from the standard energy model  $M_8$  to the model encoding the intake of energy  $M_{73}$ . The consumption of the bulb is thus encoded only implicitly by requiring a bulb (or a wrench) to overlap.  $TM_{529}$  encodes the context of actually consuming, note that the EPS wrongly learned that this can also take place in the absence of overlapping objects (none-type present).  $TM_{532}$  finally encodes the switch back to the standard model. As a nice detail, note that one red spiky cell must have been surprised by a red virus while consuming at one point.  $TM_{532}$  contains the type red virus as an object that can touch from the right.

ID	Transition Model
529	from: 73, to: 73, t: 1574, main object: cell spikes red, main features: vx: 0.005~0.133, Mxdir: 0~0.088, Mvx: 0.008~0.088, Mact: 0.845~0.369, h: 10.353~2.509, e: 9.043~2.155, BOTTOM:{types: top center wrench head, top dirt border ground}, OVERLAP:{types: wrench, none, bulb flower}
532	from: 73, to: 8, t: 171, main object: cell spikes red, main features: vx: -0.025~0.604, Mxdir: -0.17~0.825, Mvx: 0.7~0.466, Mact: 0.331~0.442, h: 10.944~2.952, e: 11.957~2.128, , LEFT:{types: none, iron block1}, RIGHT:{types: none, red virus, features:Mvx: 0.018~0, Mact: 0.043~0}, BOTTOM:{types: top center wrench head, top dirt border ground}, OVER- LAP:{types: wrench, none, bulb flower}
535	from: 8, to: 73, t: 147, main object: cell spikes red, main features: vx: 0.482~2.171, Mact: 1.002~0.11, h: 9.996~2.479, e: 10.205~3.194, BOTTOM:{types: top center wrench head, top dirt border ground}, OVERLAP:{types: wrench, bulb flower}

models as parts of the transitions they are accompanied with. We leave this to future work.

#### 9.4 Extending Spatial Relational Encodings

The limitations above could be overcome through context representations, so let us analyze their implementation in more detail. In this work, a transition model will learn Gaussian distributions for the current features of the main object (whose models switch) and types and features for objects that are directly touching the main object. An ideal situation encoding would need to learn which elements actually impact the switch and how. These dependencies are simplified in our model, but can become very complex in the real world. If a ball bounces off the ground, it makes a difference whether the ground is made of concrete, grass, sand or water, but the difference is more in the specific strength of the bounce and less in the principle of what happens. In our implementation, a lot of elements are hidden by design. Further, the exact relation is grouped into left, right, top, bottom and overlap, dismissing more fine grained patterns. Object types are used in a symbolic way, dismissing possible similarities between objects or abstract object groupings. A more general model would need to overcome these specific pre-defined reductions and operate on more flexible representations. In the end, the situation encoding is a learned encoding of a *pattern*, it is clustering different situations into a generalized one. The following will illustrate this perspective.

Consider once more the example of the green virus that moves until its energy reaches four, independent of its health, as shown in figure 9. If we consider the vector of energy and health as a point in a plane, all the situations where energy is higher than four would be connected to the moving-event and all those where energy is smaller would be connected to the standing-still-event. We can call this form of representation a situation space. See also figure 26a, where the situation space is split into the ‘moving-situations’ and the ‘standing-still-situations’. Since the transition from movement to stopping happens when energy is four, independent of health, the set of all transitions will be a line, effectively separating the two event-areas. This is a simple example, but we can expand. It might be the case that the virus also stops moving if health reaches one. This will introduce another boundary, shown in figure 26b. The implementation in this work uses Gaussian distributions as boundaries instead of singular lines, which essentially leads to fuzzy boundaries as seen in figure 26c.

It is important to realize that our implementation is limited to exactly such context patterns: in the specific situation space that the system uses, events have to have fuzzy but axis-parallel boundaries. Otherwise the system will not be able to distinguish them. This is the case if there are no dependencies between different features, i.e. if there is no co-variance between the Gaussian distributions. If the green virus stopped moving when the sum of health and energy reached five for example, the boundary

would not be axis-parallel anymore but somewhat diagonal, as seen in figure 26d. Note that this limitation does not only apply to this work but also to the models in Gumbsch et al., 2019 and Franklin et al., 2020, which also dismiss co-variance because computing co-variances comes at a huge computational cost. To effectively capture such co-variances, an implementation of situation encoding has to be specifically designed to do so. Neural network models can capture co-variances, but this is out of scope here.

One possible answer might be to use more complex models again to ‘learn’ the boundary. In 26d for example, a model learning linear combinations would already suffice but would be at an advantage compared to sets of Gaussian distributions in this case. Ideally, the situation encodings are just complex enough to capture event boundaries but not more complex than that. This, however, depends on what the situation space looks like. This is something that may be difficult to foresee. In our system, the world of BrainControl is adjusted to yield a situation space in which events are neatly bounded.

Franklin et al., 2020 on the other hand use a neural network architecture to build situation representations (and thus a situation space) when they test their system on video data. The structure of the resulting situation space is unknown and it is not immediately clear whether events form nicely bounded areas at all. They acknowledge this and hypothesize that this could lead to problems when trying to use their system on more complex data. To visualize this point, see figure 27 which shows three possible situation spaces for the stop/move event-space. Ideally, the situation space should be well formed to allow easy learning of boundaries. Note that the structure of the situation space corresponds directly to the encoding of a situation, i.e. once the encoding is fixed, the situation space is fixed. In our system, the situation space is fixed through the implementation of how the transition models learn situation encodings.

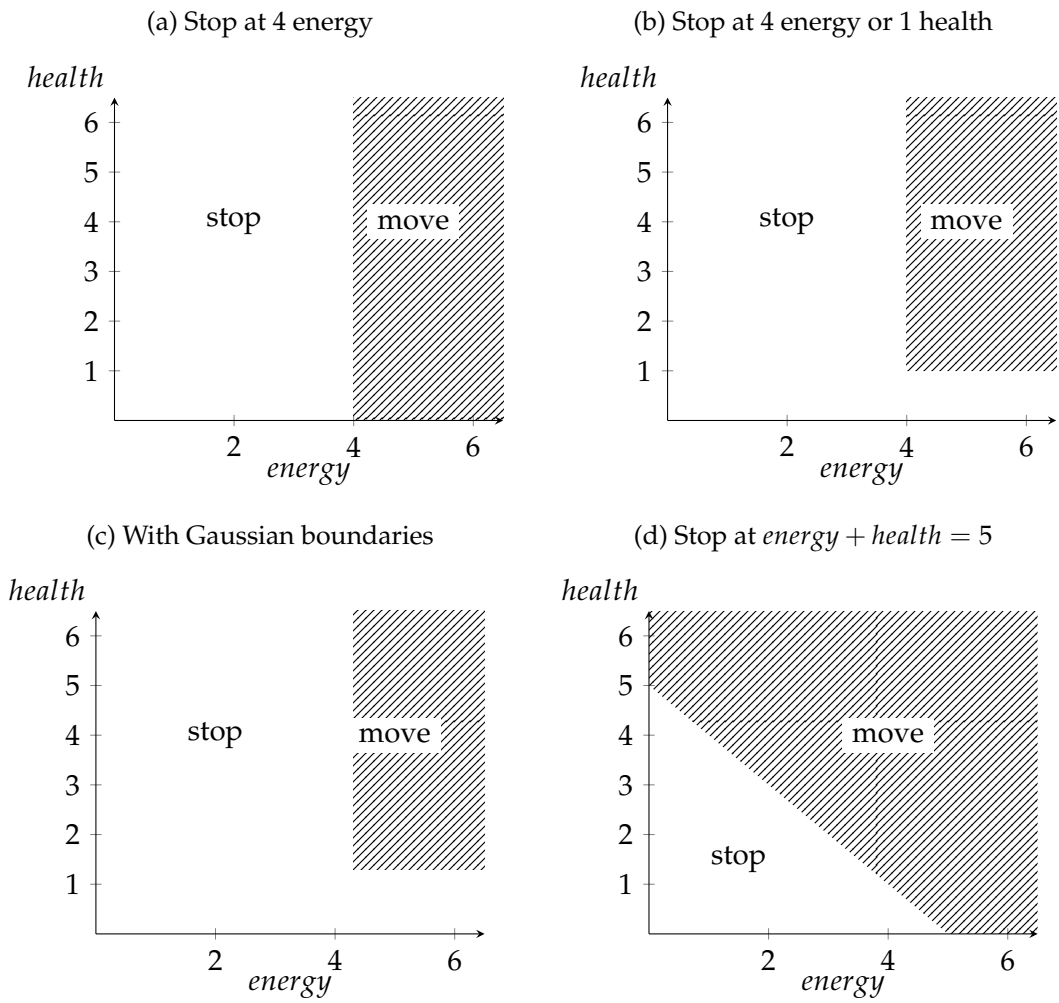


Figure 26: Boundaries between events in situation space. Our system will be able to learn a, b and c, but not d.

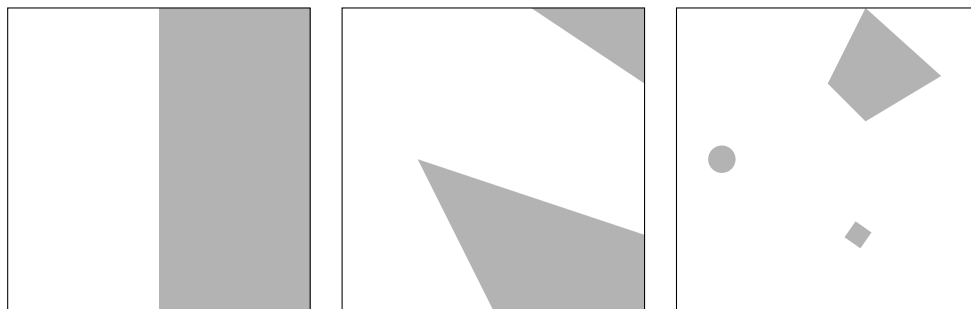


Figure 27: Three possible situation spaces. Grey area correlates with move, white area correlates with stop.

## 10 The Language Processing System

To implement the language system, we follow the theoretical discussion in section 3: the system should receive an utterance as input and try to build a representation of the event that the utterance describes. It should identify lexemes in the utterance using a mental lexicon and determine their roles using grammatical rules. Finally, this is then combined into a scene reconstruction by the ISS, as described in section 6. We restrict LEARNA to simple utterances in this work, i.e. plain utterances that describe scenes. Given the discussion on the EPS, we can now say that an utterance has to describe a scene within the BrainControl world, denoting scenes that the EPS can understand. We have seen in the preceding sections that the EPS provides transition models, factorized event dynamics models as well as relation models as structures that the LPS can build upon. Entity or top-down encodings are also available, but the EPS focuses on the other encodings. These encodings are thus available to the LPS to connect to lexemes.

It should be noted that the original BrainControl also implements language processing, see e.g. Schrodte, Kneissler et al., 2017. Their system learns effects of collisions, i.e. changes in states of entities that result from collisions. These collision effects are mapped onto a pre-defined system of tags. These tags are then used in a context-free grammar that exhaustively describes a mapping between possible sentences and sequences of tag-defined object changes. This means that the LPS and the ISS, as described in this work, are replaced by the set of tags and the context-free grammar. This work presents a very different approach, so it is difficult to compare their system to ours.

### 10.1 Constructing Abstract Scene Representations

In this section we will describe the LPS on the computational level of description and specifically outline the ASR that should be constructed from an utterance. The ASR should capture the information contained in the utterance and make it available to the ISS. Let us start with a simple utterance that describes *one* object undergoing a change, i.e. an utterance of the form ‘subject-verb’ in English. Sticking to the BrainControl World, let us start with the simple sentence 28. The structures that the EPS employs when perceiving a falling box were shown in figure 21. The dynamic development of falling was encoded by a model that predicted the change in vertical velocity. The situation as a whole was encoded by a transition model which captured the context of the model. Note that the transition model is operating on the box as its main object. This is a basic necessity; the model predicting change has to operate on one entity. We assume that this entity corresponds to the subject-position of the sentence.

(28) The box falls.

Further, *box* can simply be linked to the box-type encoded by the EPS. What *falls* is mapped to is a question of great debate. The EPS calls on several event dynamics

models, transition models and relation models when perceiving the falling box in figure 21. In the case of falling, the only *necessary* part seems to be the model predicting the downward movement. In all cases of falling, this model will be present. Note that this is an assessment from the outside, knowing what falling means. While other cases may be more complex, in the case of falling it would suffice to map *falls* to the model of downward movement.

We thus propose that the intersection of *the box falls* and the encoding as built by the EPS is the transition model which encodes *falling* and the encoded type *box* as the object the model operates on. *The box falls* would require applying the transition model on an object of the type *box*. This forms the basis of the ASR and by extension of the LPS. Object categories will be mapped to *types* as encoded by the EPS and verbs will be mapped to transition models. How this mapping is learned is beyond the scope of this work, we simply assume that the mapping is established in the following.

(29) The box falls to the ground.

Progressing to a slightly more complex utterance 29, the system has to also integrate *to* into the scene. LEARNA approaches prepositions like *to* by assuming spatial meaning, e.g. moving towards and arriving at something in the case of *to*. We thus map selected prepositions onto the *relation transition models* that are learned by the EPS. Recall that the EPS encodes patterns of relations in *relation models* and their transitions in relation transition models. This fits nicely with *to* in 29. There is a relation model encoding that two objects have decreasing distance and another one that encodes two objects being in direct contact. The relation transition model that encodes the transition from the first to the second then corresponds to the spatial meaning of *to*. Prepositions are thus mapped to the relation transition models that emerged in EPS through learning. How the mapping is learned is again beyond the scope of this work.

(30) The box falls to the ground and the box breaks.

To add more complexity, note that nouns are so far mapped to object types, but they refer to actual objects. Consider the utterance 30. The question arises whether there are two boxes in the scene or just one. This requires hidden identifiers. The LPS thus has to establish object-IDs in addition to types. In the case of 30, the system may actually construct two interpretations, one with one box that falls and then breaks and another one with two boxes. Such an identifier is also necessary if *pronouns* are used.

Using these methods, the LPS uses a mental lexicon that maps nouns to object types, verbs to transition models and prepositions to relation transition models. Additionally, the LPS contains a grammar function that analyzes the building blocks and determines which lexeme or lexemes fulfill which role. In this work, we only use simple roles, i.e. either *subject* as an object that a transition model should operate on or *object* as part of a relation.

Finally, we pre-define certain conjunctions that describe a *time-dependence*, e.g. *before*. The LPS uses these pre-defined conjunctions to analyze utterances like 30, which actually describes a *sequence* of changes.

In summary, the ASR that the LPS ends up with has three components. It defines objects that have to be present, it can define transition models and relation transition models on those models and it can order the models to be active in a temporal sequence. This way, the ASR defines necessary requirements on the scene reconstruction. We assume that this captures the *meaning* of the simple utterances that LEARNA focuses on. Building the scene reconstruction from the ASR is the task of the ISS, which thus does not have to operate on language. At the same time, the LPS does not have to operate on models.

Given that language is ambiguous, it can happen that the LPS finds several ASRs that are legal but competing interpretations of an utterance. Arriving at several interpretations can result from several factors, recurring objects were already mentioned. In other cases, mappings may be ambiguous, e.g. if several models are mapped to *falling* or several possible object types are mapped to *box*. In such a case the LPS produces all possible interpretations and leaves it to the ISS to decide which are plausible.

## 10.2 The Lexicon Function

The EPS uses integer identifiers for models and types. The lexicon thus contains mappings between words and sets of integers, let us call this mapping function  $lex(name) \rightarrow \{id_x, \dots\}$ . *Virus* will be mapped to the integer types of green and red virus, i.e.  $lex('virus') = \{id_{gv}, id_{rv}\}$ . Our implementation does not include dedicated adjectives, so the lexicon will include entries such as  $lex('green virus') = \{id_{gv}\}$  and  $lex('red virus') = \{id_{rv}\}$ .<sup>37</sup> A list of known entity expressions can be found in appendix B.

The lexicon provides simple look-up, i.e. for some string  $s$ , we can retrieve the models or entities as  $lex(s)$ . To circumvent the problem of overlapping identifiers, the lexicon function is actually split into distinct mapping functions, each managing either noun phrases, verbs, prepositions or conjunctions. This means the LPS uses dedicated functions  $lex_{noun}$ ,  $lex_{verb}$ ,  $lex_{preposition}$  and  $lex_{conjunction}$ . When processing a string, the LPS thus has to decide beforehand which lexicon function to use. Specifically for LEARNA, a string should never be ambiguous as to which category it belongs to, see also below.

Finally, in addition to sets of integers describing models, the mental lexicon also contains information on the temporal meaning of a preposition. Recall that  $lex_{preposition}$  maps strings to relation transition model IDs. A relation transition model predicts a

---

<sup>37</sup>Possible names for entities are defined within the code and can not be added while the system is running. Verbs and prepositions *can* be added while running the program on the other hand. This uses a special matching mechanism, though the user still has to decide which word fits a specific transition model.



change in relation models and thus poses the question of *when* this change is supposed to happen. Consider the utterance in 31.

(31) The box falls from the platform to the ground.

The transition model for falling refers to a downward movement, which spans several timesteps. *From* and *to* describe a change in relation *relative* to the timesteps that falling refers to. Specifically, the box should be on the platform *first*, then fall and then hit the ground. The relation model transition that is linked to *from* happens when the downward movement starts. The relation model transition linked to *to* happens when the downward movement ends. One relation transition happens strictly *before* the other. While the inference mechanism could figure this out, there are two reasons why this time-dependence should be encoded beforehand. The first reason is *simplicity*: memorizing that the *to* transition happens after the ‘action’ is easy. As we will see, the inference mechanism is computationally very costly. So, having to reorder timesteps at the time of inference would come at a huge cost. The second reason is that there are cases where the time-dependence is strictly encoded in the sentence and no inference mechanism can recover that information. The sentence ‘I went to Germany before I learned German’ is just as plausible as ‘I learned German before I went to Germany’. ‘Before’ encodes a time-difference between the two actions which the inference mechanism can not recover. The LPS thus encodes whether a relation transition model should activate at the start or at the end of the transition model it depends on.

Conjunctions work differently. Their lexicon function  $lex_{conjunction}$  does not return an integer ID but instead tells the grammar function directly what kind of a sequence the conjunction encodes, i.e. whether it means that the preceding clause is to happen before, during or after the succeeding clause. Since conjunctions are pre-defined, these commands are hard coded and only meaningful in the context of the grammar module.

### 10.3 Processing a Sentence

While the EPS is learned from scratch, the rules to analyze syntax are pre-defined; learning grammar is beyond the scope of this work. To describe how a sentence is processed in terms of lexical meaning and grammar in LEARNA, we will follow the utterance in (32), which exhaustively showcases the implemented functionality. The exact representation of the ASR will be described in steps and summarized later. It is important to note that all rules or steps described in the following are necessary: if any of the steps fail, the grammar function will return an error and abort processing. LEARNA always operates on input sentences in an isolated fashion, i.e. references to prior inputs are impossible.

(32) ‘The box sits on the platform and it falls from the platform to the ground after the green virus falls.’

In the following we assume that (32) was provided to the LPS as input, which first removes all articles.<sup>38</sup> As a first step, all conjunctions are identified through string-matching and the sentence is split at each conjunction into what we call *clauses* here. This results in a representation shown in (33). The system can only parse a maximum of four conjunctions. More conjunctions would lead to what is called *combinatorial explosion*, see the discussion at the end of this section. To reduce ambiguity, ‘and’ is taken to mean ‘and then’ and is functionally equivalent to ‘before’. After identifying the clauses, the LPS parses the conjunctions and generates possible timelines, as seen in 34. While the ambiguity between the two timelines may not seem intuitive here, consider the more natural example (35). The timelines are abstract in the sense that a timestep in the ASR only defines the *order* of what is happening, not the duration.

(33)  $C_1$  =box sits on platform,  $C_2$  =it falls from platform to ground,  $C_3$  =green virus falls

(34)  $C_1$  and  $C_2$  after  $C_3 \Rightarrow \{(C_3 \rightarrow C_1 \rightarrow C_2), (C_1 \rightarrow C_3 \rightarrow C_2)\}$

(35) Tom went to town and did some shopping after he got his money

The LPS then analyzes each of the clauses in isolation. The structure of a clause is strictly defined; the LPS only parses clauses of the structure described here. It first tries to find the first word or the first two words of a clause in the noun phrase lexicon, e.g. ‘green virus’ in  $C_3$ . Pronouns like ‘it’ are also identified but return a special *none-type*. The LPS assumes pronouns can refer to preceding entities, but not succeeding ones. This first lexeme is taken to be the *subject*. The LPS then processes the next word as the *verb* of the sentence. After the verb, the system tries to match another noun phrase as a direct object. This is optional, i.e. the LPS will continue if there is no direct object. Finally, prepositions with succeeding noun phrases (prepositional phrases) are matched, e.g. ‘on platform’. The LPS can match more than one prepositional phrase. All this information is saved for the clause. An example of this is shown in 36. Relations are made up of a set of possible types and a set of possible models, e.g.  $R_1 : types = \{id_{platform1}, id_{platform2}, \dots\}, models = \{id_a, id_b, \dots\}$ . Prepositional phrases are always taken to belong to the clause they are in, so they can not take scope beyond a conjunction.

- (36) a.  $C_1$  : remaining clause = box sits on platform, s: , v: ,o: ,r:  
 b.  $C_1$  : remaining clause = sits on platform, s: $\{id_{box}\}$  , v: , o: , r:  
 c.  $C_1$  : remaining clause = on platform, s: $\{id_{box}\}$  , v: $\{id_x, id_y, \dots\}$  , o: , r:  
 d.  $C_1$  : remaining clause = , s: $\{id_{box}\}$  , v: $\{id_x, id_y, \dots\}$  , o: , r:( $R_1$ )

Direct objects present a curious case, as there is no unique relation implied by their role. The LPS thus interprets a direct object to mean potentially any *relation model*,

<sup>38</sup>Our system allows for articles to make the input more natural but ignores them during processing.

ID	type	references
1	<i>redbox</i>	the box
2	<i>platform</i>	the platform, it
3	<i>platform</i>	the platform
4	<i>ground</i>	the ground
5	<i>green – virus</i>	the green virus

timestep/subject	TM	RTM	references
$t_1/5$	$TM_x$		falls
$t_2/1$	$TM_y$	$RTM_a \rightarrow 2$	sits on the platform
$t_3/2$	$TM_z$	$RTM_b \rightarrow 3, RTM_c \rightarrow 4$	falls from the platform to the ground

Table 9: A possible abstract scene representation derived from the sentence: *The box sits on the platform and it falls from the platform to the ground after the green virus falls*. References are not part of the abstract representation but only included for understanding. Note that this represents a *wrong* interpretation of the sentence: the pronoun should refer to the box and the two mentions of platform should refer to the same object.

not relation transition model. It will generate ASRs that each contain the direct object related to the subject with one relation model.

Whenever an entity reference is encountered, the system additionally saves an *entity-mention* which encodes a sentence-wide unique identifier, as seen in 37.

(37) ‘ $C_1 = \text{box}_1$  sits on  $\text{platform}_2$ ,  $C_2 = \text{it}_3$  falls from  $\text{platform}_4$  to  $\text{ground}_5$ ,  $C_3 = \text{green virus}_6$  falls’

At this point, the LPS has built a set of possible timelines, knows which entities the sentence involves as a whole and which types or models the words can be mapped to. It has analyzed the grammatical structure and knows which objects the models operate on. The system now generates possible combinations of these sets and timelines in the form of ASRs. One possible ASR that could be derived from 32 through the process described here is shown in table 9. The set of ASRs is then produced as the output of the grammar module.

Note that the number of combinations rises quickly; this is known as combinatorial explosion. Consider (32) once more. Let us assume that each noun and verb can be mapped to 5 types or models. Let us also assume that each preposition is unambiguous and maps to exactly one relation transition model. In (38), each word is annotated with the number of types or models it could map to. The pronoun ‘it’ can refer to the box or the platform in this case. This leaves us with  $5^8 * 2 * 1^3 = 390625$  possible combinations

for this sentence. Additionally, there are 2 different possible timelines and there could be one or two platforms, so this leads to 1562500 possible combinations. In our system, this number is much smaller because each word often refers to one or two possible objects, and models are often unique for a given object type. Recall also that the LPS aborts if there are more than four conjunctions in a sentence to reduce combinatorial explosion.

(38) 'The box(5) sits(5) on(1) the platform(5) and it(2) falls(5) from(1) the platform(5) to(1) the ground(5) after the green virus(5) falls(5).'

To reduce the number of ASRs, clauses could be processed *and* simulated one after another, which could remove meaningless combinations at an earlier stage. This is out of scope in this work, however.

## 11 The Inference and Simulation System

In the ISS, the structures learned by the EPS and the ASRs generated by the LPS are to be merged into a consistent scene reconstruction, if possible. Once the LPS has constructed possible interpretations of an utterance, the ISS has to select those that are plausible given the system's world knowledge, ideally choosing the most plausible one. The main problem in this case is that an ASR contains only restrictions but the EPS does not allow evaluation of restrictions. The EPS can only provide prediction forward from a starting point, the ISS has to derive the scene reconstruction. The first question that arises is which level of detail is appropriate.

In respect to human communication it is difficult to judge how much detail is added during comprehension. Consider the utterance in (39a). There is a missing link between the first and the second that has to be filled. A human listener might even conclude that the utterance was actually that in (39b), judging the mishearing more plausible than what would be needed to fill in the information gap in the original utterance. Not every detail has to be filled in however, because some things seem irrelevant: whether Marie took the car, how long that took, what game she played exactly, what she was wearing at the time or what her mood was. On the other hand, some information may be necessary, maybe Marie would usually walk but at the time of the utterance the path is blocked and she has to take the car on a 30 km detour. In such a case, an attentive human listener may ask (39c). To ask this question, the listener would have to try to fill the information gap but then notice that the blocked path is at odds with Marie going to town.

- (39) a. Marie went to town and started gaming  
b. Marie went to town and it started raining  
c. How did Marie get there?

LEARNNA on the other hand is not equipped to judge the relevance of a detail. The ISS thus simply tries to build a complete simulated scene that satisfies the requirements and qualifies as a scene reconstruction, specifying all features of all the objects involved. While this may be inefficient, an incomplete simulation may lead to *false positives*, i.e. judging scenes plausible that are not, because the system does not know which information *can* be left out without any consequences.

Building a scene reconstruction is much more difficult than one might expect. As opposed to the LPS, the ISS has no chance of enumerating and evaluating all possible simulations. Assume the system wants to simulate a moving ball and rounds speed to the second digit. Assuming the maximum speed encountered in BrainControl is around 10, this results in the interval  $[-10.00, 10.00]_{/0}$  as possible values for speed, exactly 2000 distinct values. In our implementation, running one simulation already takes more than 100 milliseconds, so only simulating for different speed values would take at least 3:20 minutes before combinatorial explosion from combining different

features or different interpretations is taken into consideration. The inference system thus cannot construct all possible simulations. The following section will provide technical background on this.

### 11.1 Reconstructing a Scene Is Costly

Recall the situation space introduced in section 9.4, where the state of a scene at one timestep (a situation) could be seen as a vector  $v$  in a situation space. The scene reconstruction that the ISS searches for can then be seen as a sequence of situations, i.e. a sequence of vectors  $s = (v_1, v_2, \dots, v_n)$ . From this perspective, the LPS produces requirements that specific vectors have to adhere to, e.g. that there is an object of a specific type encoded in that vector. A requirement can then be abstracted to an equation saying that there has to be a vector  $v_i$  somewhere in  $s$  where some dimensions have specific values. Let us take the situation space in figure 26a as an example. A specific vector  $v_1 = (5.5, 6)$  in that situation space tells us that the object has 5.5 energy and 6 health. Now, let us assume that the object loses energy when moving, but loses health all the time. A sequence of situations can be seen as an ordered list of situation vectors  $s = (v_1, v_2, \dots, v_n)$  where each vector corresponds to one timestep, so it can be illustrated as a *path* through situation space, let us call this the *scene-path*. Starting from  $v_1$ , this would mean that the object travels through this situation space over time. See figure 28 for an illustration of the following.

Now, we can illustrate what an ASR looks like. The utterance ‘object moves and then stops’ would necessitate that an appropriate scene-path starts in the ‘move’-area and then reaches the ‘stop’-area. In terms of the situation space, ‘being in the move-area’ is equal to *energy*  $> 4$ . Requirements include a time-axis too: ‘objects moves and then stops’ probably does not mean that the object moves one timestep and then stops immediately. Instead, the ‘moving’ has to be stable at least for a short time. This effectively introduces an extended list of requirements that have to be satisfied in that order. In this setup, the ISS has to find a suitable scene-path.

Now the predictive models come back into play. Any two consecutive situations in a scene have to satisfy the predictive models too. In a completely deterministic case, there is only one pre-determined scene-path for any starting situation. Once the starting situation is set, deterministic models will prescribe a path forward. If the world is not strictly deterministic, predictive models will predict the likelihood of one situation following another, restricting what scene-paths can look like once the starting situation is set. In the EPS, these predictions are made up of normal distributions. All this is shown in figure 28.

Returning to actual understanding and different possible interpretations, the inference mechanism has to judge whether an interpretation, i.e. a set of requirements, can be satisfied by a scene-path that does not conflict with the predictive models. With this abstract description of the task, finding a scene-path through the situation space can be

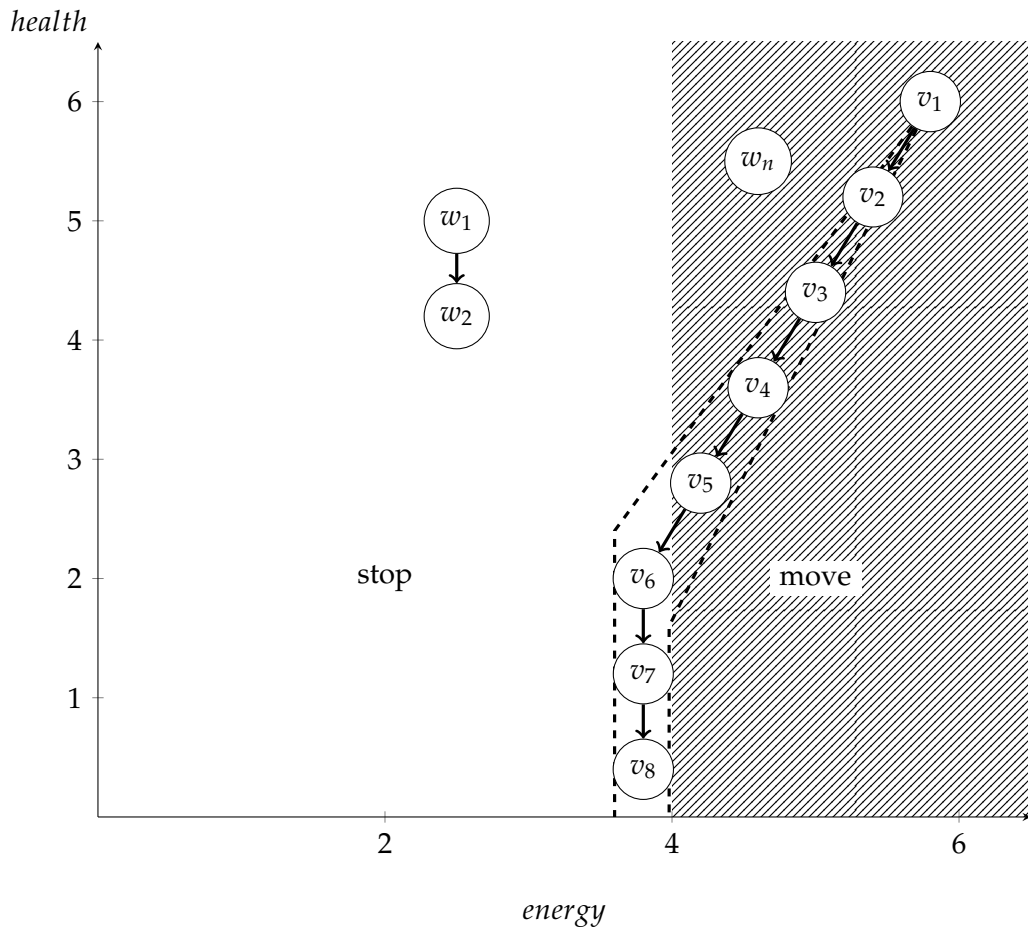


Figure 28: Visualization of a very simple situation space. Each vector describes a situation for the object,  $v=(5,4)$  for example says that the object has 5 energy and 4 health. The sequence describes the changes of the situation through time: the object loses health consistently and loses energy while moving, but stops moving once energy reaches 4. This sequence already satisfies the requirements derived from the sentence: *objects moves and then stops*. The dashed lines illustrate what an average 10% uncertainty may look like, i.e. how different the predictions may be. Note that a vector, once it is on the left side in the stop-area, can not move anymore under the assumptions of this simple situation space and there is no path to the move area from the stop area (w-nodes).

seen as an optimization problem. For example, each situation that is possible within the situation space could be regarded as a *node* so that predictions define connections between the nodes, creating something called a *graph*, see e.g. Korte et al. (2018). The structure in figure 28 is a graph. Then the problem could be formulated as a connectivity problem (Korte et al., 2018): are the nodes required by the ASR connected through predictions? The general algorithm for this problem visits every connection in the worst case. This is unacceptable because it would mean that the ISS would have to look at *every possible prediction*. Take the sentence ‘object stands still and then moves’ for example. It does not have a corresponding scene-path in 28: once an object stopped, there is no way back because the predictions can only decrease energy and health. The situations denoted  $w_1$  and  $w_n$  illustrate these requirements. This is obvious to a human observer that knows how this specific world works: there is no starting point within the stop-area that will end up in the move-area. But how does the algorithm make sure there is no connection? Without exploiting further structure, any simple algorithm will have to try every possible combination and see where the prediction will lead to. Note that this is already problematic in the simple example in 28. The computation cost grows exponentially with every additional dimension of the vector space and with every additional requirement set by the input sentence.

Another common algorithmic approach would view this problem as what is called a *constraint satisfaction problem*, see Freuder et al. (2006), the problem of assigning values to variables so that a set of requirements is satisfied. One popular technique is called *backtracking*. Backtracking tries to build assignments incrementally, e.g. by first finding a partial assignment that satisfies all the requirements, and then extending this partial assignment until a complete assignment is found. The advantage is that once a partial assignment does not satisfy one of the requirements, it can be abandoned and all other assignments that build upon that partial assignment will also not work. Backtracking thus views partial assignments as being ordered in a tree-like structure: if a partial assignment does not satisfy the requirements, its *branch* can be cut off or *pruned*. It is important to note however that backtracking can still lead to the enumeration of many partial solutions. A different approach called *local search* tries to start from a complete assignment that violates some requirements and improve this assignment to satisfy more and more requirements. Note that these approaches may still end up enumerating all possibilities in the worst case.

In terms of situation space, it seems natural to say that the variables are the timesteps of a scene-path. A scene-path is thus an assignment, but is mostly pre-determined by the predictive models. A partial assignment is thus not very useful: if we started with  $w_1$  and  $w_n$  in figure 28, there is no bridge  $w_2, \dots, w_{n-1}$  that makes this into a scene-path. If we start with a complete scene-path on the other hand, there is a chance of adapting certain properties of the path to make it fit more requirements. Given the sequential nature of the requirements in our case, we can start with a default scene-path and adapt it so that the requirements on the first timestep are satisfied.



If this is not possible, then there is no scene-path at all. Once they are satisfied, we can adapt the scene-path again so that the requirements on the second timestep are satisfied. If this is not possible, then again, there is no scene-path at all and we can abort. This way, we can build the scene-path iteratively but still prune some branches. Because enumeration is intractable, the *adaptation* will be the most critical part of our implementation. If the adaptation was random, the system would behave in the same way as the algorithm trying to find connections mentioned above, enumerating all possible scene-paths in the worst case.

In summary, it seems crucial to derive plausible heuristics to adapt scene-paths given certain requirements as the common algorithms are likely intractable. It should be added that our setup does not formally qualify for these algorithms, because the situation space is not bounded in the strong sense. The situation spaces discussed above describe two features of a *single object*. In our case, we may need to infer extra objects. ‘Marie went to town’ may imply a car that was not mentioned. Our system could theoretically decide to keep adding objects; there is no initial requirement not to do so. The system could potentially keep adding objects or simulate further and further into the future, leading to extensive running times. Adding objects has the effect of adding another situation-space however, extending the number of variables in the constraint satisfaction problem and the number of nodes in the graph. Neither of the algorithms is prepared for this extension. The constraint satisfaction problem is not even able to handle different scene-path lengths, which would also be an extension of the number of variables in the formalization chosen above. Nonetheless, they provide a valuable formal approximation to the task at hand. The problem of enlarging the search space will have to be dealt with in a more pragmatic way.

## 11.2 The Inference Mechanism

Given the discussion above, our ISS has to solve several problems: how to build a scene-path, how to adapt a scene-path and when to give up search. Giving up is *easy*: the system gives up when no heuristic is able to improve the current scene-path, when the scene-path is longer than a pre-defined threshold, or when there is a general error due to the limitations of the system itself. As the devil is in the details, this will be explicitly mentioned where necessary. If the ISS could solve these problems, the general mechanism of inference would be simple: build a default scene-path and adapt until it satisfies the requirements, abort the adaptation if there is no more chance of improving.

At this point it should be noted that Wharton et al. (1991) also describe a model to do the kind of inference the ISS does. Their ‘construction-integration model’ also approaches the problem of how to inject or use world knowledge during comprehension. Relying on propositional knowledge (see also section 4), their model receives a sentence and *constructs* a large set of interconnected propositions. In a second step, the

model *integrates* this large set, i.e. refines the initial set to be a precise and consistent interpretation of the sentence. While their description is less formal, their strategy may be translated into situation spaces and scene-paths as follows. First, a large number of situations are activated and their connections established. Their model explicitly relies on the assumption that within this large set, a viable scene-path fully exists. In the integration step, all the situations that do not belong to the scene-path are removed. I.e. the scene-path remains. This approach is very different from the one in this work. Given the structures provided by the EPS, activating a large set of situations would need to rely on very smart heuristics to not be swept away by combinatorial explosion. Instead, as few situations as possible should be activated or rather simulated.

Given a set of ASRs, the ISS tries to find a plausible scene for each interpretation individually. Given a single ASR, the ISS will first test whether there is an obvious contradiction, e.g. a mismatch between the object type encoded in a transition model and the type of the main object of the ASR. In such cases the ASR is discarded immediately. The inference mechanism then starts by assembling an initial default setup, i.e. a generic initial situation to start simulation and adaptation from. All objects are added into a bounded two dimensional space, the *mental map*. The idea of a mental map follows the *mental sketchpad*, a part of working memory for spatial representations introduced by Baddeley et al. (1974). Starting from this setup, the ISS then simulates forward using the encodings provided by the EPS. If the simulation diverges from the requirements, a special *matcher* function tries to generate a meaningful error signal. That error signal is then used to choose an appropriate adaptation heuristic. The heuristic will change the initial setup and the ISS repeats the process of simulation, matching, adapting.

### 11.3 Implementation of Simulation

At the very start, all necessary objects are inserted at pre-defined default positions, i.e. non-touching spots aligned on a grid. Furthermore, a default *ground* is added on the bottom of the mental map. The main reason to include a ground is the absolute nature of spatial representation: the mental map functions as a coordinate system with a unique origin in the top left corner and positions of objects are absolute in reference to that origin. Since the mental map is bounded, objects will by default fall out of its bounds after only few timesteps. This 'out-of-bounds' phenomenon is avoided by simply placing a ground at the lower bound of the mental map. Note that the ground tiles are added extra without regard for the objects that exist in the ASR, i.e. if the sentence mentioned the ground, the ASR will introduce another ground entity in addition to the default ground. This default setup is visualized in figure 29 for a correct ASR of the sentence: 'the box sits on the platform and it falls from the platform to the ground'.

With this basic setup as a starting situation, the system can run a *simulation* using

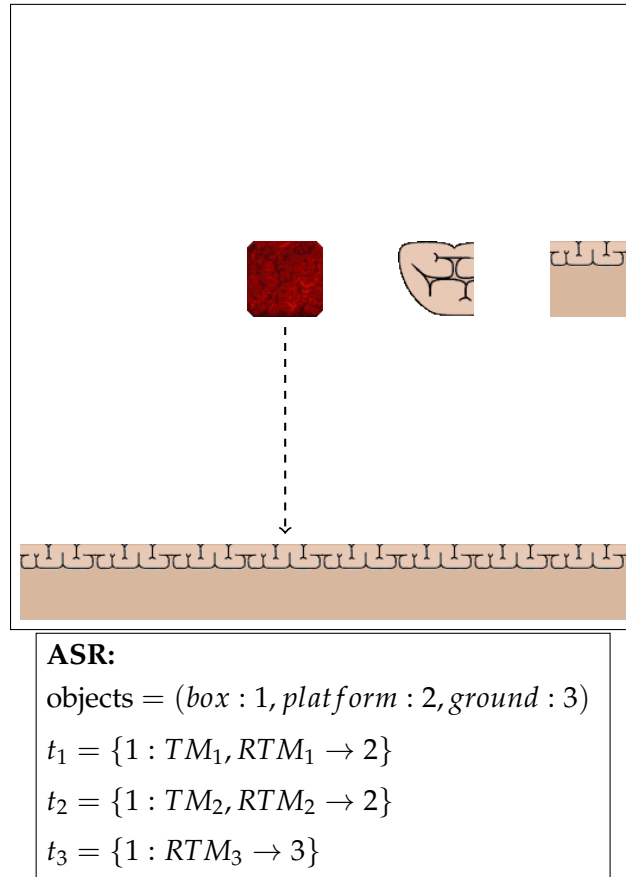


Figure 29: Default setup for a correct interpretation of: *the box sits on the platform and it falls from the platform to the ground*. The ASR contains three objects in this correct case. Box (1), platform (2) and ground (3), where box is the main object of all models in this specific case. There are three timesteps containing the required models. Note that there is an additional ground tile because the system adds a default ground *additionally*. If the sentence already contains ground objects, those will get identifiers in the LPS, whereas the default ground tiles have different identifiers. The system does not try to merge these because this is not central to our implementation and would have required considerable programming effort. The dashed line represents the development of the simulation from the starting situation.

the encodings from the EPS. While the predictive models provided by the EPS may contain uncertainties, i.e. ranges of possible predictions, the ISS always uses the mean. In normal operation, the models would simply predict the next timestep from the current situation. To simulate further, the system takes the predicted values  $pv_{t+1}$  as the actual next situation. The ISS can then apply transition models to produce simulated model switches. If the situation described by  $pv_{t+1}$  matches the learned context of a transition model  $TM$ , this transition will be triggered and the appropriate models will be switched. If there are several transition models that match the situation, the one that has been experienced the most times will be chosen. Finally, the potentially new set of models will be used to predict another timestep  $pv_{t+2}$ . This could be repeated indefinitely, so the simulation will be cut off after 150 timesteps. Further, if an object leaves the mental map, i.e. it shoots off the sides, the simulation is also aborted.

One problem with this approach is that it is not clear which models should be active at the very start. If the system starts simulation using incorrect initial models, the simulation will be flawed from the start. We solve this problem by initializing a simulation with *default models*. The EPS learns which models are applied to each object type the most times. These are usually zero models, i.e. no movement, no change. These learned defaults are used to initialize models for all simulated objects.

In the setup in figure 29, the box would be initialized with models that predict zero movement. Because the box is not supported by fixed objects, however, a transition model that predicts that the box should be falling is immediately applied. Platform and ground objects on the other hand never move in the actual BrainControl world and are thus predicted to always be at rest. Once the box has been falling enough it will collide with the ground. Recall that predicted movement was using a special ‘velocity before collision’ to avoid strange and complex non-linear changes upon collision, see also figure 13. In simulation, the predicted movement will thus not stop at the boundary of an object but move two objects right into one another, i.e. the box would be simulated *into* the ground. This is even more problematic because transitions use the collision direction, which in the simulation case would be *overlap* instead of *bottom*. Overlap is impossible for objects that actually collide in the real game world and thus cannot have been experienced. Therefore, the transition that would otherwise predict that the box collides with the ground will not be triggered. This problem is very complex and arises specifically because of the discrete nature of time in the game world. The ISS thus uses the pre-defined mathematical functions that the engine uses to resolve the collision, adjusting the *overlap* to *bottom* in this case. If, however, the objects are meant to overlap, the ISS will register that changing from *overlap* to *bottom* did not change anything (in regards to transition models) and thus reverse the adjustment. This way, objects that naturally collide in the game world will do so in simulation, given that the appropriate transition model is learned.

It seems obvious that the scene-path that results from simulation in figure 29 does not satisfy the ASR. If the simulation only returns *yes/no* as feedback, there will be no

cue for the adaptation heuristics as to what went wrong and what could be changed. Therefore the system needs a function that tries to match the ASR to the current scene-path and then to return a meaningful error message.

## 11.4 Comparing a Simulated Scene to the Requirements

The two main problems when comparing a simulation with an ASR are how to align the differing timescales and how to create a meaningful error message.

Before the matching, the ASR has to be adapted, however. If the ASR contains a transition model that describes the ongoing application of a model, i.e. the transition of one model to itself, this transition model should usually be active for more than just one timestep. A falling object should not only fall one timestep. The LPS is unaware of this issue and simply assigns a model ID to a timestep. Consider the transition models  $TM_1$  and  $TM_2$ , where  $TM_1$  predicts the reflexive transition  $M_1 \rightarrow M_1$  and  $TM_2$  predicts the transition  $M_1 \rightarrow M_2$ . Consider also an ASR that contains one object  $O_1$ , with  $TM_1$  in the first timestep and  $TM_2$  in the second. This could, for example, result from the sentence ‘the box falls and it breaks’ (ignoring the ground in this case). In simulation,  $M_1$  should be active for a few timesteps before the transition to  $M_2$  happens. Since the ASR timeline is  $t_1 : TM_1, t_2 : TM_2$  a scene-path where  $TM_1$  would only be active one timestep might match. To avoid this, the ASR is adapted to use  $M_1$  instead of  $TM_1$ , resulting in an adapted ASR  $t_1 : M_1, t_2 : TM_2$ . The system then only accepts scene-paths where predictive models are active longer than a pre-defined number of timesteps, which is set to 10 in this work.

Unfortunately, this introduces the problem of alignment between timescales. one timestep in the ASR can be realized in 10 or more timesteps in the simulated scene. This further introduces problems for the simultaneity of transitions. The LPS would be able to parse the sentence: *the ball falls to the ground while the box falls to the ground.*<sup>39</sup> *Falling* should be active for at least 10 timesteps. *To the ground* implies a switch of relation models and will only be active at exactly one timestep. Because the ASR uses discrete timesteps, however, this will also mean that both objects will have to hit the ground at the exact same timestep. In the end, this issue is beyond the scope of our system.

With the adapted ASR, the system will now try to match the timesteps of the ASR to the current scene-path one by one. Using the ASR introduced above as an example again, the system will try to match the requirement that  $M_1$  is  $O_1$ ’s active model for 10 *consecutive* timesteps. If  $M_1$  is actually active for  $O_1$  for 10 timesteps,  $t_1$  will be marked as satisfied and the system will now try to match the requirement that  $M_2$  is  $O_1$ ’s active model and the transition is described by  $TM_2$ . If  $M_1$  continues to be the active model,

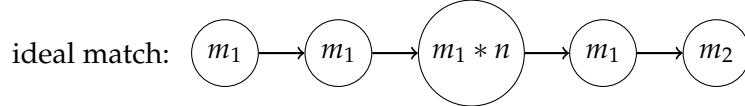
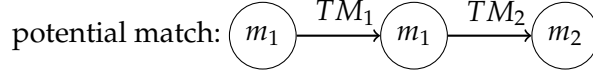
---

<sup>39</sup>It is a general pattern that our model has difficulties with simultaneity because of the strict requirements. In this case, simultaneity only means that there are some timesteps during which both objects are falling. Our system can only do all timesteps or none.

Actual timeline:

fall	hit
------	-----

**ASR:**  $t_1 = TM_1(m_1 \rightarrow m_1), t_2 = TM_2(m_1 \rightarrow m_2)$



**adapted ASR:**  $t_1 = m_1, t_2 = \{TM_2(m_1 \rightarrow m_2), m_2\}$

Figure 30: Adapting the ASR: because the transition models may describe the transition from one specific model to the exact same one, their respective predictive models are added to the ASR. Predictive models have to match at least 10 timesteps in our system. Transition models that predict a model switch remain in the ASR. Note that in this case, there is no additional timestep inserted because the two transition models can follow one another. Indeed, the sentence *the box hits the ground* could end up with the same adopted ASR, even though it would initially have looked different.

the system will simply wait. When  $M_1$  finally transitions into  $M_2$ , the system will mark  $TM_2$  as satisfied for now because it only happens once. If  $M_2$  continues for 10 timestep, the system will mark  $t_2$  satisfied as a whole. If  $M_2$  is active for less than 10 timesteps, the system will reset  $t_2$  completely, requiring  $TM_2$  to happen again. The adaptation process is illustrated in figure 30. Once all timesteps are marked as satisfied, the system will mark the current scene-path as *satisfying*, triggering the system to accept the ASR as a valid interpretation with the current scene-path as its realization.

If the system stops simulation *before* all timesteps are satisfied, there will now be information about ‘how far the scene-path got’, i.e. up to which timestep the ASR was satisfied. This in turn allows to return a specific error. The last timestep of the simulation that was in line with the ASR is also registered. A special error-finding function now inspects the difference between this timestep  $t_{match}$ , the following non-matching timestep  $t_{break}$  and the ASR-timestep that was supposed to be satisfied by  $t_{break}$  but was not. This error-finding function will look for a missing transition first. If a transition is missing, the function will try to find out what parts of the transition model’s situation encoding are missing. The function will then look for missing predictive models, and finally for missing relation models. If at least one of these searches is successful, the function will return the error that specifically encodes what

is missing. Sometimes, however, the function will not find an error even though the ASR is not yet satisfied. In such a case, the system will simply give up on the current ASR. Note that the error does not represent the things that are missing overall but only at a very local level, i.e. at exactly the point where the current scene-path started to depart from the requirements of the ASR. It represents a guess at what could be done to improve the scene-path. This error is then handed over to the adaptation heuristics.

## 11.5 Adaptation Heuristics

Adaptation of the scene-path follows a simple schema: try to adapt the setup given the current error, simulate the scene again and repeat. If there are errors or an adaptation heuristic is impossible to fulfill, the ISS will give up on the current ASR.

In principle, the error provided by the matching function could refer to all parts of the scene-path. Because the original ASR only contains transition models and relation transition models, however, these are the only errors that need to be solved by our system. An error from relation transition models will translate to an error in relation models which in turn contain the following information: whether the two objects touch and, if so, the direction of touch or the change in distance if they do not touch. For transition models, recall that whether a transition is triggered is determined by the situation through three aspects: the features of the main object, the type (and thus existence) of touching objects and the features of those touching objects. All available adaptation heuristics are described in the following, including errors for which they would be activated.

**Move-Away** The move-away heuristic is applied when objects touch but should not. This heuristic would be applied for the utterance *the ball falls to the ground* if the distance between ground and ball is not large enough to allow a long enough falling period. In such a case, this distance would be increased.

**Zero-Motor** The zero-motor heuristic is applied to *stop* movement, e.g. when an object already satisfied the requirement of touching another object but moves away.

**Test-TMs** The test-TMs heuristic is applied whenever movement is required of an object. This may be required explicitly by a situation encoding, or it may be triggered if an object is supposed to move away from an object but does not.

**Add-Move-Into-Path** The add-move-into-path heuristic is applied when a collision is required or if an object is required in general. This heuristic would be applied in the situation in figure 29, where the platform would be moved into the path of falling of the box.

**Add-Motor** The add-motor heuristic is applied in cases where a transition model relies on specific motor commands to be applied. The simulation function has

the ability to adjust the motor commands of objects, so this heuristic can require such an adjustment. For example, if the input utterance is *the virus moves to the box*, the transition model from the default at-rest model to the moving-model encodes that such a transition happens when the virus applies movement force. The add-motor heuristic tries to fulfill this requirement.

## 11.6 Summary

In summary, the ISS operates on each ASR provided by the LPS in isolation. It first establishes a neutral default scene, placing all requested objects as well as a default ground in the scene. Then the ASR is adapted so that transition models are switched for their respective predictive models. The mechanism then repeats a testing-adaptation scheme: it simulates the scene starting from the initial setup. A dedicated function tries to match the simulation with the ASR and produces a structured error if the simulation deviates from the ASR. This error refers to the first timestep of the ASR that is violated, ensuring that the errors are fixed in a chronological fashion. A set of adaptation mechanism are then available that try to fix the error.

In case an adaptation mechanism can fix an error, the initial setup will be altered. The inference mechanism then simulates the scene again from this new setup and repeats the matching and testing. When an error cannot be fixed, the inference is aborted and the ASR discarded. This produces a tree-like search structure for each ASR. Once a satisfying scene-path is found, it is returned as a valid scene reconstruction of the input sentence. In cases where the tree becomes too deep, the inference is also stopped. This search process can be seen in the discussion of different examples in section 12.



## 12 Evaluation and Discussion of the Implementation

Given the description of the LPS and the ISS above, these can now be combined with the EPS to form the complete LEARNA architecture. In the following, we will discuss sample utterances that LEARNA in its current form can or cannot understand. The problem with evaluating the system is that it is highly specialized in its current form in regards to vocabulary, so any standard data set is not applicable for testing. Instead, we have to analyze the behavior on a case by case basis.

A second problem for evaluation lies in the way the ISS works, i.e. by heuristically trying a range of manipulations. In more complex cases, this leads to a very large number of *failed* attempts of fixing the scene-path. For example, the ISS may try to apply adaptation heuristics 120 times for the utterance *the virus falls from the platform*. Analyzing what went wrong requires examining the exact steps of the program, finding out where the heuristics should succeed theoretically, tracking down this exact step in the large tree of failed attempts, and then see *why* the heuristics fail at that one timestep. The analysis in the following will have to remain more general.

In regards to the visualization of the scene reconstruction, note that the ISS initializes a simulation with the objects placed in the *middle* of the screen. If the simulation works, this placement will not be changed, i.e. objects will usually begin a simulation in the center, falling to the ground, unless the simulation does not specifically request something else.

### Pushing

Consider the utterance in (40). LEARNA is able to build a scene reconstruction successfully, which is illustrated in figure 31, showing a sequence of situations that represents the scene reconstruction. Note that in this case, the simulation works with the initial conditions set. This is most likely the case because the ISS manipulates the motor commands of the red virus right away, which is the only manipulation that is needed in this scene. Table 10 shows that there were no further adaptation heuristics necessary. This table shows the ASR how it is manipulated over time (ASR) as well as the situation at the beginning of the simulation (setup) and its changes due to adaptation heuristics. This kind of table is also used in more examples below in the same format.

(40) the red virus moves and it pushes the ball

(41) the red virus pushes the ball

On the other hand, LEARNA does not find a matching scene reconstruction for (41). As mentioned above, it is difficult to say what went wrong exactly, but a common error is that an ASR would require an additional step to be inserted. This happens if there is no direct transition from the default model, which in this case would be the not-moving model, to the required model, in this case the pushing model. This

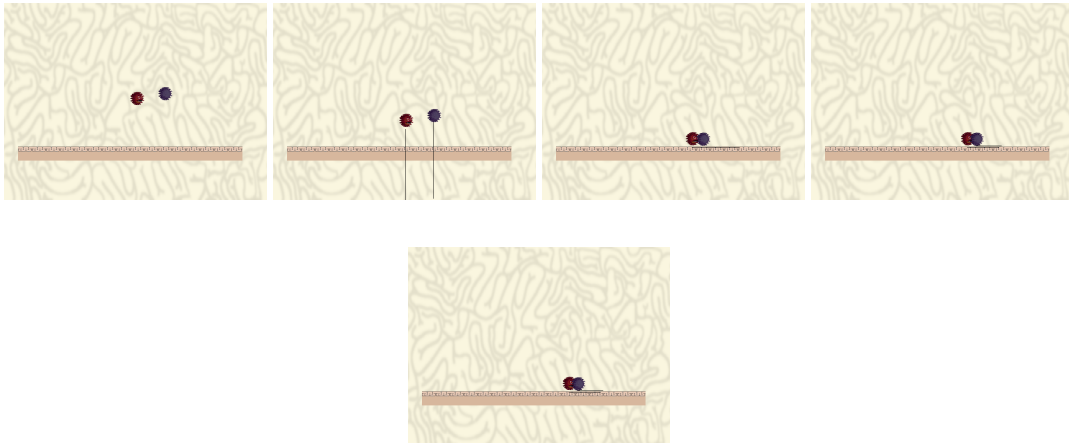
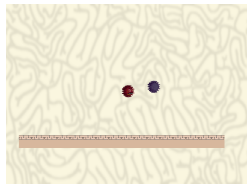


Figure 31: Simulation for the successful interpretation of the message: *the red virus moves and it pushes the ball*

is because entities do not push objects immediately from being at rest, but first start moving. Inserting such timesteps is not supported by the ISS, however.

Table 10: Search tree for the message *the red virus moves and it pushes the ball*

Error and Heuristic	ASR	Setup
initial, adaptation: none, initial call	$\text{objects}=\{\text{red virus:0, virus ball:2}\},$ $t_0 =\{ 0:\{PM_{25}\} \},$ $t_1 =\{ 0:\{PM_{55}, RM_6(2)\} \}$	

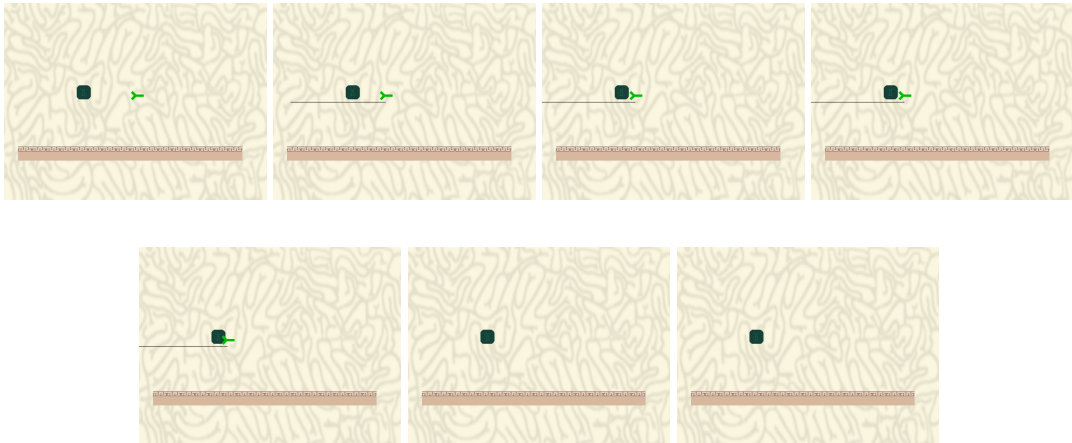


Figure 32: Simulation for the successful interpretation of the message: *arrow flies to the box and it breaks*

### Breaking/Flying/To

Consider now the utterance in (42). As illustrated in figure 32, LEARNA is able to find an appropriate scene reconstruction. In this case, the inference process contains more steps, as seen in table 12. What is especially noteworthy is that this example is ambiguous, as others that follow. This will be discussed in a separate section. In the same way, the utterance in (43) is simulated successfully. Note that the ISS inserts an additional ground tile here, as discussed above. This can be seen in figure 33. The ground tile is moved into the path of the falling box during inference, as can be seen in table (13).

- (42) arrow flies to the box and it breaks
- (43) the box falls to the ground and it breaks

Note that in (42) and (43), the *to* is simulated correctly. This is also the case for (44) and (45), both of which are simulated correctly.

- (44) the green virus falls to the ground
- (45) the virus moves to the box

Table 11: Search tree for the message *the virus moves to the box*

Error and Heuristic	ASR	Setup
tic		

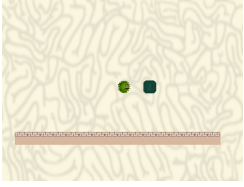
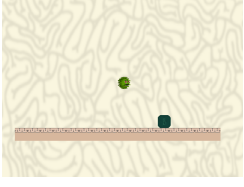
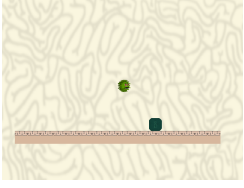

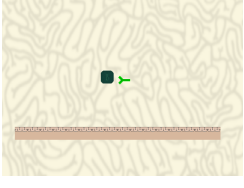
initial, adaptation: none, initial call	objects={green virus:0, iron block1:1}, $t_0 = \{ 0:\{PM_{25}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_6(1)\} \}$	
need decreasing distance between object 0 and object 1 but the distance actually increases, adaptation: add- move-into-path	objects={green virus:0, iron block1:1}, $t_0 = \{ 0:\{PM_{25}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_6(1)\} \}$	
need touch be- tween object 0 and object 1 while touch was not established before, adaptation: add- move-into-path	objects={green virus:0, iron block1:1}, $t_0 = \{ 0:\{PM_{25}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_6(1)\} \}$	

Table 12: Search tree for the message *arrow flies to the box and it breaks*

Error and Heuristic	ASR	Setup
initial, adaptation: none, initial call	objects={bullet willy:0, iron block1:1}, $t_0 = \{ 0:\{PM_{17}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_7(1)\} \}$ , $t_2 = \{ 0:\{PM_{154}\} \}$	
need decreasing distance between object 0 and object 1 but the distance actually increases, adaptation: add- move-into-path	objects={bullet willy:0, iron block1:1}, $t_0 = \{ 0:\{PM_{17}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_7(1)\} \}$ , $t_2 = \{ 0:\{PM_{154}\} \}$	

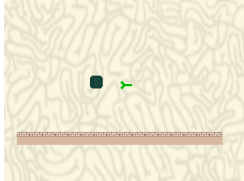
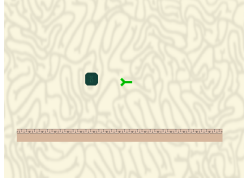
need decreasing distance between object 0 and object 1 but they actually touch, adaptation: move-away	objects={bullet willy:0, iron block1:1}, $t_0 = \{ 0:\{PM_{17}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_7(1)\} \}$ , $t_2 = \{ 0:\{PM_{154}\} \}$	
need decreasing distance between object 0 and object 1 but they actually touch, adaptation: move-away	objects={bullet willy:0, iron block1:1}, $t_0 = \{ 0:\{PM_{17}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_7(1)\} \}$ , $t_2 = \{ 0:\{PM_{154}\} \}$	

Table 13: Search tree for the message *the box falls to the ground and it breaks*

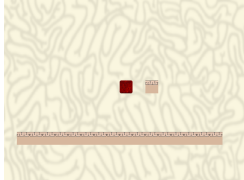
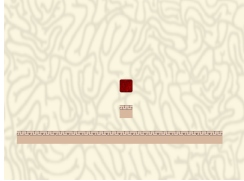
Error and Heuristic	ASR	Setup
initial, adaptation: none, initial call	objects={movable box:0, top dirt border ground:1}, $t_0 = \{ 0:\{PM_{35}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_4(1)\} \}$ , $t_2 = \{ 0:\{PM_{154}\} \}$	
need decreasing distance between object 0 and object 1 but the distance actually increases, adaptation: add-move-into-path	objects={movable box:0, top dirt border ground:1}, $t_0 = \{ 0:\{PM_{35}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_4(1)\} \}$ , $t_2 = \{ 0:\{PM_{154}\} \}$	

Table 14: Search tree for the message *the green virus falls to the ground*

Error and Heuristic

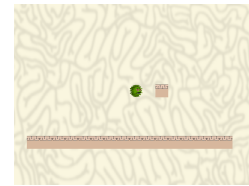
ASR

Setup

---

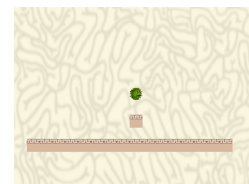
initial,  
adaptation: none,  
initial call

objects={green virus:0, top dirt border  
ground:1},  
 $t_0 = \{ 0: \{ PM_{35}, RM_2(1) \} \}$ ,  
 $t_1 = \{ 0: \{ RM_4(1) \} \}$



need decreasing  
distance between  
object 0 and object  
1 but the distance  
actually increases,  
adaptation: add-  
move-into-path

objects={green virus:0, top dirt border  
ground:1},  
 $t_0 = \{ 0: \{ PM_{35}, RM_2(1) \} \}$ ,  
 $t_1 = \{ 0: \{ RM_4(1) \} \}$



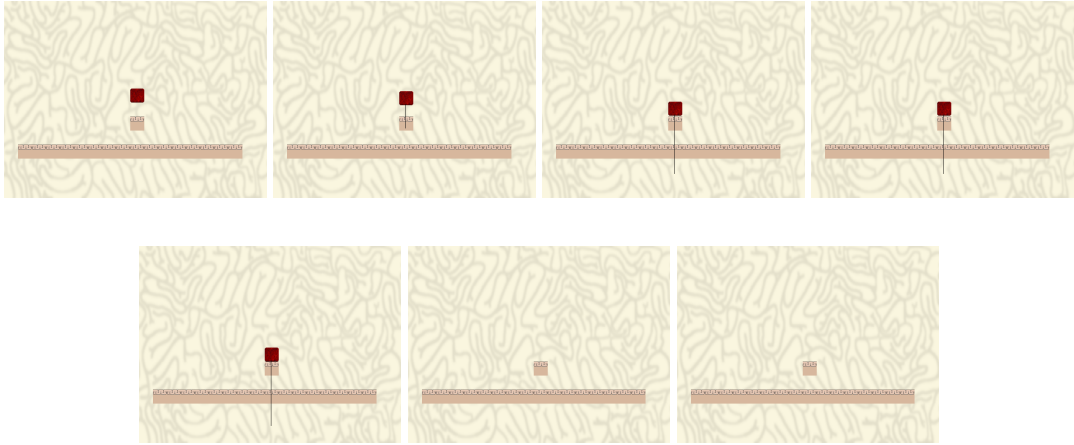


Figure 33: Simulation for the successful interpretation of the message: *the box falls to the ground and it breaks*

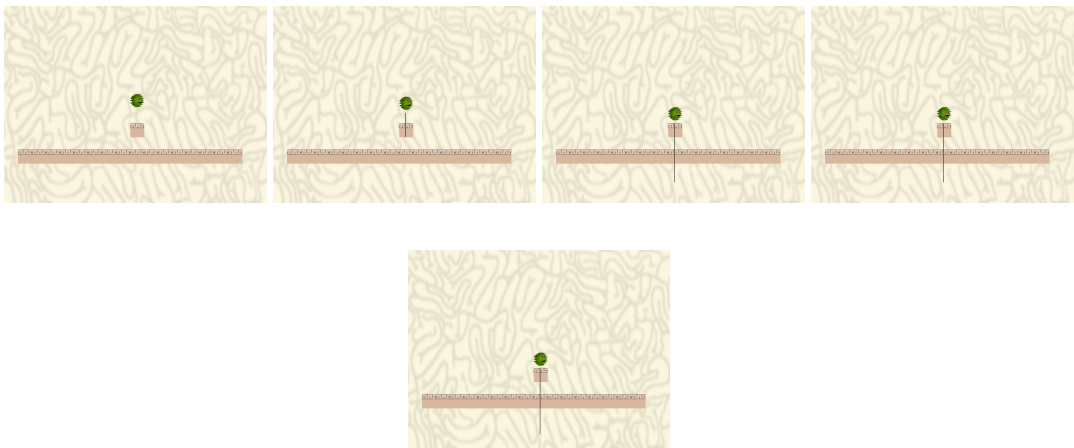


Figure 34: Simulation for the successful interpretation of the message: *the green virus falls to the ground*

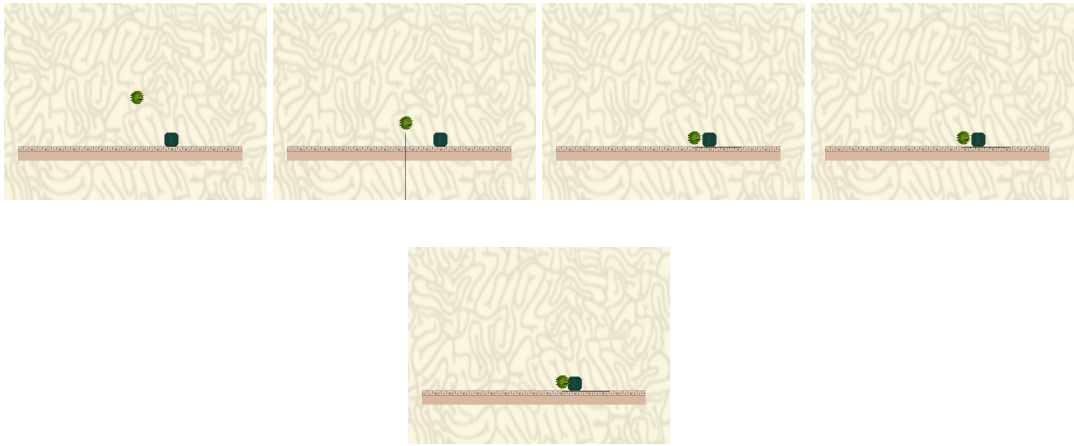


Figure 35: Simulation for the successful interpretation of the message: *the virus moves to the box*



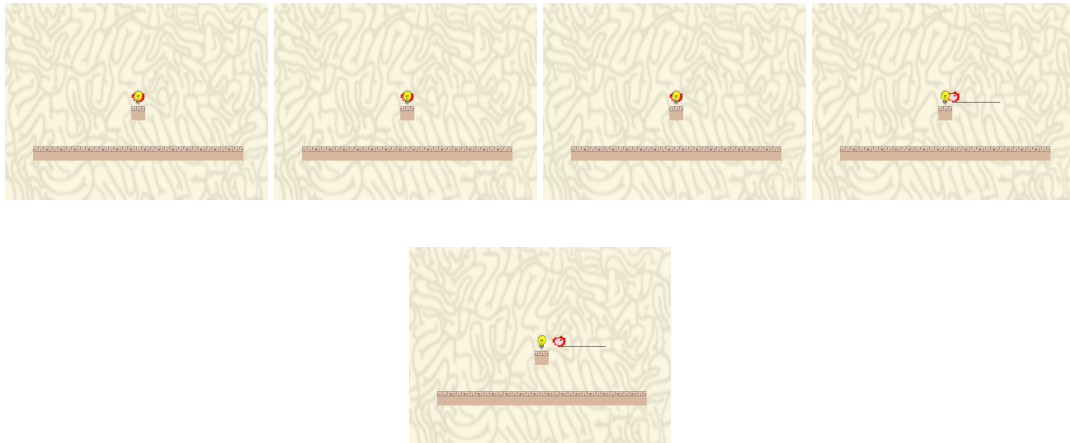


Figure 36: Simulation for the successful interpretation of the message: *the cell consumes the bulb and it moves*

### Consuming

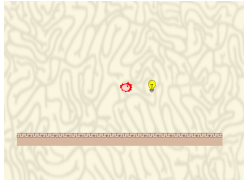
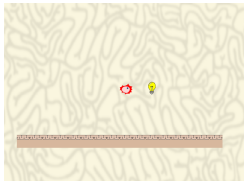
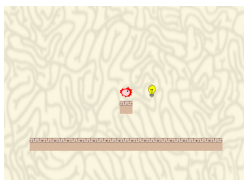

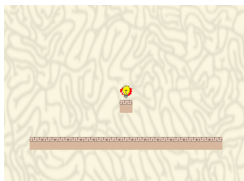
Figure 36 shows the successful simulation of (46), which required extensive manipulation through adaptation heuristics as seen in table 15. Note that this also showcases that direct objects *can* be processed correctly if the LPS and ISS work together. Interestingly, the utterance (47) cannot be simulated by the ISS. While the simulation successfully simulates the bulb losing health, eventually reaching a value below zero, the transition model for *breaking* is never triggered. This failure has a very specific reason: in BrainControl, the cell would adjust its motor commands before the cell is removed to not consume empty space. This adjustment is registered by the situation encoding of the breaking transition model, however. In this case, the motor command adaptation heuristic notices the necessity to adjust the motor commands, but it has no way of telling the simulation function *at which exact timestep* the motor commands should be changed. I.e. either they do not change, or they are changed way before the bulb's health reaches zero, effectively ending the consumption process prematurely.<sup>40</sup>

- (46) the cell consumes the bulb and it moves
- (47) the cell consumes the bulb and it breaks

Table 15: Search tree for the message *the cell consumes the bulb and it moves*

---

<sup>40</sup>This problem was analyzed by the author by means of following the code step by step with a debugger. It is thus difficult to provide evidence for it here other than the description.

Error and Heuristic	ASR	Setup
initial, adaptation: none, initial call	objects={cell spikes red:0, bulb flower:1}, $t_0 = \{ 0:\{PM_{73}, RM_7(1)\} \}$ , $t_1 = \{ 0:\{PM_{25}\} \}$	
need specific motor commands for object0, adaptation: set-motor-commands	objects={cell spikes red:0, bulb flower:1}, $t_0 = \{ 0:\{PM_{73}, f_4 = 0.0, f_5 = 0.0, RM_7(1)\} \}$ , $t_1 = \{ 0:\{PM_{25}\} \}$	
need new object 0 in direction BOTTOM, adaptation: add-move-into-path	objects={cell spikes red:0, bulb flower:1, top dirt border ground:-2}, $t_0 = \{ 0:\{PM_{73}, f_4 = 0.0, f_5 = 0.0, RM_7(1)\}, -2:\{f_4 = 0.0, f_5 = 0.0, f_6 = 0.0, f_7 = 0.0\} \}$ , $t_1 = \{ 0:\{PM_{25}\} \}$	
need new object 0 in direction OVERLAP, adaptation: add-move-into-path	objects={cell spikes red:0, bulb flower:1, top dirt border ground:-2}, $t_0 = \{ 0:\{PM_{73}, f_4 = 0.0, f_5 = 0.0, RM_7(1)\}, 1:\{f_4 = 0.0, f_5 = 0.0, f_6 = 0.0, f_7 = 0.0\}, -2:\{f_4 = 0.0, f_5 = 0.0, f_6 = 0.0, f_7 = 0.0\} \}$ , $t_1 = \{ 0:\{PM_{25}\} \}$	
need specific motor commands for object0, adaptation: set-motor-commands	objects={cell spikes red:0, bulb flower:1, top dirt border ground:-2}, $t_0 = \{ 0:\{PM_{73}, f_4 = 0.0, f_5 = 0.0, RM_7(1)\}, 1:\{f_4 = 0.0, f_5 = 0.0, f_6 = 0.0, f_7 = 0.0\}, -2:\{f_4 = 0.0, f_5 = 0.0, f_6 = 0.0, f_7 = 0.0\} \}$ , $t_1 = \{ 0:\{PM_{25}, f_4 = 0.99934596, f_5 = 1.0006675, f_7 = 0.0484942\} \}$	

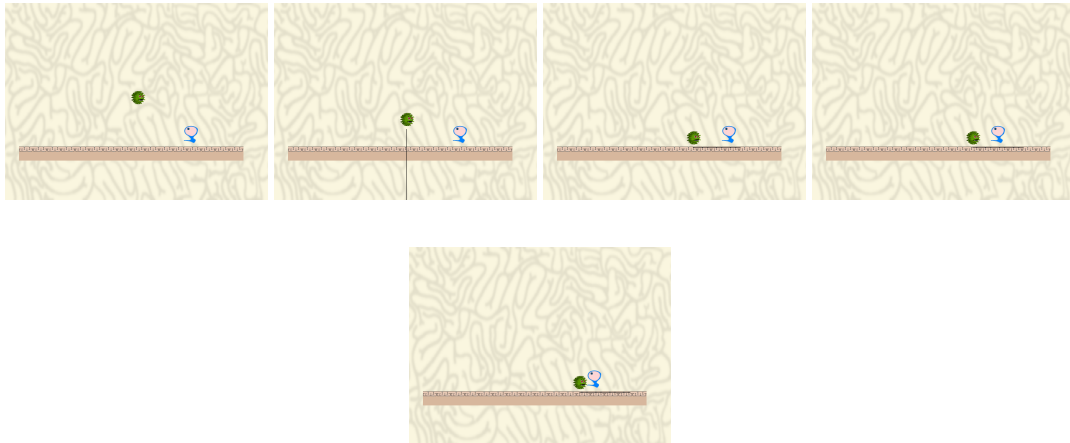


Figure 37: Simulation for the successful interpretation of the message: *the green virus moves and it attacks the blue cell*

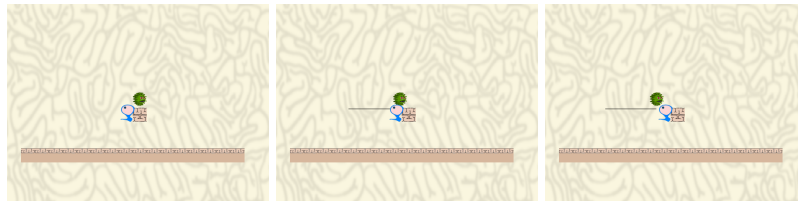


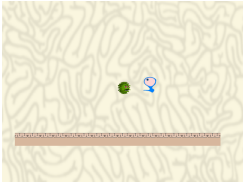
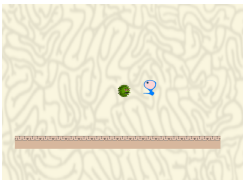

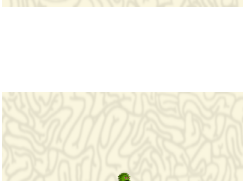
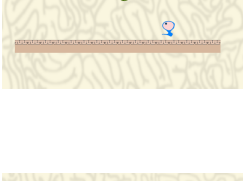
Figure 38: Simulation for the successful interpretation of the message: *the virus attacks the cell*

### Attacking / Fleeing

The utterance in (48) can be simulated successfully by LEARNA, as shown in figure 37, see table 16 for the adaptation process. What is curious about this example is that (49) is also simulated successfully. However, the simulation is non-standard in terms of what happens in BrainControl, as depicted in figure 38. This simulation exploits the fact that the transition model to trigger falling for a blue cell has never experienced the cell falling while something was on its right. This way, the cell seems glued to the platform and while the virus performs an attack, the scene does not play out the way it would in the world of BrainControl. Note that (50) can not be simulated, suffering the same problem as (47) above.

- (48) the green virus moves and it attacks the blue cell
- (49) the virus attacks the cell
- (50) the green virus attacks the blue cell and it flees

Table 16: Search tree for the message *the green virus moves and it attacks the blue cell*

Error and Heuristic	ASR	Setup
initial, adaptation: none, initial call	objects={green virus:0, cell long blue:2}, $t_0 = \{ 0:\{PM_{25}\} \}$ , $t_1 = \{ 0:\{PM_{60}, RM_2(2)\} \}$	
need specific motor commands for object0, adaptation: set-motor-commands	objects={green virus:0, cell long blue:2}, $t_0 = \{ 0:\{PM_{25}\} \}$ , $t_1 = \{ 0:\{PM_{60}, f_5 = 1.8012906, RM_2(2)\} \}$	
need decreasing distance between object 0 and object 2 but the distance actually increases, adaptation: add-move-into-path	objects={green virus:0, cell long blue:2}, $t_0 = \{ 0:\{PM_{25}\} \}$ , $t_1 = \{ 0:\{PM_{60}, f_5 = 1.8012906, RM_2(2)\} \}$	
need decreasing distance between object 0 and object 2 but they actually touch, adaptation: move-away	objects={green virus:0, cell long blue:2}, $t_0 = \{ 0:\{PM_{25}\} \}$ , $t_1 = \{ 0:\{PM_{60}, f_5 = 1.8012906, RM_2(2)\} \}$	
need decreasing distance between object 0 and object 2 but they actually touch, adaptation: move-away	objects={green virus:0, cell long blue:2}, $t_0 = \{ 0:\{PM_{25}\} \}$ , $t_1 = \{ 0:\{PM_{60}, f_5 = 1.8012906, RM_2(2)\} \}$	

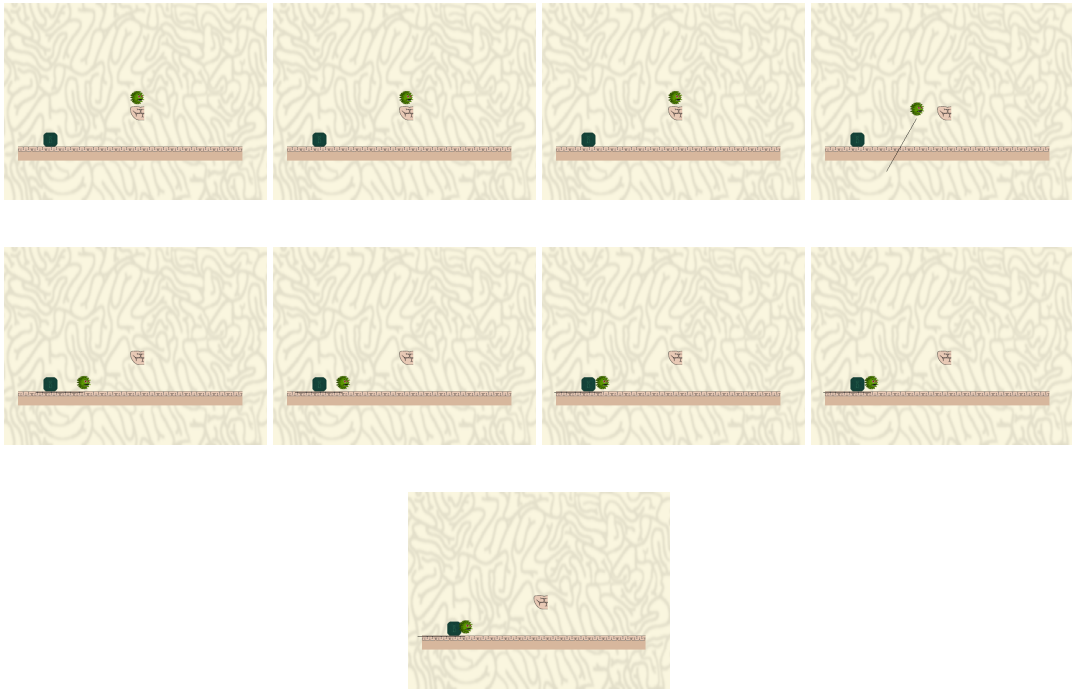


Figure 39: Simulation for the successful interpretation of the message: *the green virus moves from the platform and it moves to the green box*

**From**

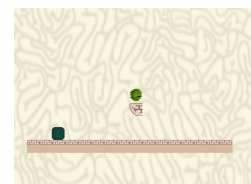
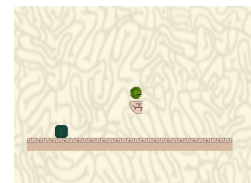
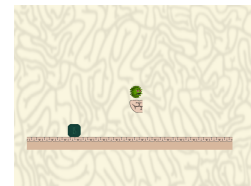
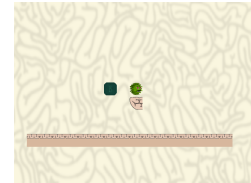
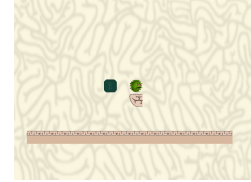
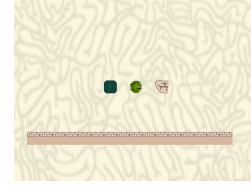
The utterance in (51) showcases a more complex inference process which is successful, see figure 39 and table 17. *From* can also lead to problems, however, as in (52), where the ISS is unable to insert transitions, in a similar way as in example (41) above.

- (51) the green virus moves from the platform and it moves to the green box
- (52) the green virus falls from the platform

Table 17: Search tree for the message *the green virus moves from the platform and it moves to the green box*

Error and Heuris- tic	ASR	Setup
--------------------------	-----	-------

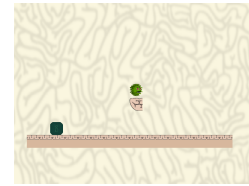
initial,	objects={green virus:0, top left wrench
adaptation: none,	head:1, iron block1:3},
intial call	$t_0 = \{ 0:\{RM_4(1)\} \},$
	$t_1 = \{ 0:\{PM_{11}, RM_3(1)\} \},$
	$t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \},$
	$t_3 = \{ 0:\{RM_7(3)\} \}$
need touch be-	objects={green virus:0, top left wrench
tween object 0	head:1, iron block1:3},
and object 1 while	$t_0 = \{ 0:\{RM_4(1)\} \},$
touch was not	$t_1 = \{ 0:\{PM_{11}, RM_3(1)\} \},$
established before,	$t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \},$
adaptation: add-	$t_3 = \{ 0:\{RM_7(3)\} \}$
move-into-path	
need specific mo-	objects={green virus:0, top left wrench
tor commands for	head:1, iron block1:3},
object0,	$t_0 = \{ 0:\{RM_4(1)\} \},$
adaptation: set-	$t_1 = \{ 0:\{PM_{11}, f_4 = -0.9998265, f_5 =$
motor-commands	$0.99474275, RM_3(1)\} \},$
	$t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \},$
	$t_3 = \{ 0:\{RM_7(3)\} \}$
need decreasing	objects={green virus:0, top left wrench
distance between	head:1, iron block1:3},
object 0 and object	$t_0 = \{ 0:\{RM_4(1)\} \},$
3 but the distance	$t_1 = \{ 0:\{PM_{11}, f_4 = -0.9998265, f_5 =$
actually increases,	$0.99474275, RM_3(1)\} \},$
adaptation: add-	$t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \},$
move-into-path	$t_3 = \{ 0:\{RM_7(3)\} \}$
need decreasing	objects={green virus:0, top left wrench
distance between	head:1, iron block1:3},
object 0 and object	$t_0 = \{ 0:\{RM_4(1)\} \},$
3 but they actually	$t_1 = \{ 0:\{PM_{11}, f_4 = -0.9998265, f_5 =$
touch,	$0.99474275, RM_3(1)\} \},$
adaptation: move-	$t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \},$
away	$t_3 = \{ 0:\{RM_7(3)\} \}$
need decreasing	objects={green virus:0, top left wrench
distance between	head:1, iron block1:3},
object 0 and object	$t_0 = \{ 0:\{RM_4(1)\} \},$
3 but they actually	$t_1 = \{ 0:\{PM_{11}, f_4 = -0.9998265, f_5 =$
touch,	$0.99474275, RM_3(1)\} \},$
adaptation: move-	$t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \},$
away	$t_3 = \{ 0:\{RM_7(3)\} \}$



need decreasing distance between object 0 and object 3 but they actually touch,  
 adaptation: move-away

objects={green virus:0, top left wrench head:1, iron block1:3},  
 $t_0 = \{ 0: \{ RM_4(1) \} \},$   
 $t_1 = \{ 0: \{ PM_{11}, f_4 = -0.9998265, f_5 = 0.99474275, RM_3(1) \} \},$   
 $t_2 = \{ 0: \{ PM_{11}, RM_2(3) \} \},$   
 $t_3 = \{ 0: \{ RM_7(3) \} \}$

---



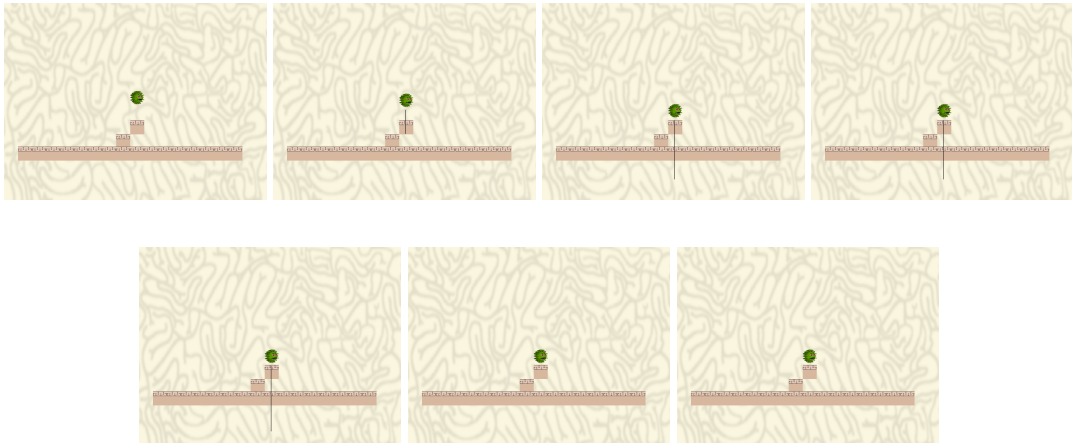


Figure 40: Simulation for the successful interpretation of the message: *the green virus falls to the ground and it rests on the ground*

### Falling and On

To also showcase that *on* is interpreted correctly, consider examples (53) and (54), both of which are simulated successfully, see figure 40 and table 18 for (53) and figure 41 and table 19 for (54). Note that for (53), the ASR contains two distinct ground objects, but the ISS is content with actually using only one of them. This is because in determining whether a scene-path matches an ASR, relations are only matched based on the relation type and the type of the object, not its ID. Thus, even if the ASR needs the ISS to establish the two relations with two different ground objects, the matcher function will accept a scene-path where the relations exist with one ground object, since it is the correct type.

Finally, (54) presents a somewhat curious case, but the simulation shown in figure 41 is acceptable. This example will also be discussed again later.

- (53) the green virus falls to the ground and it rests on the ground
- (54) the green virus rests on the platform and it moves to the box after it falls

Table 18: Search tree for the message *the green virus falls to the ground and it rests on the ground*

Error and Heuristic	ASR	Setup



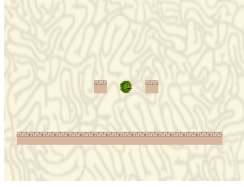

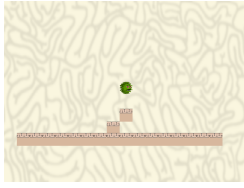

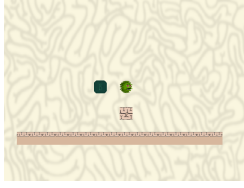
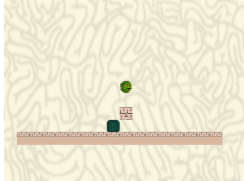
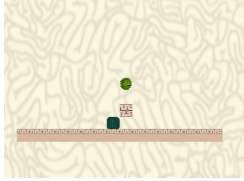

initial, adaptation: none, initial call	objects={green virus:0, top dirt border ground:1, top dirt border ground:3}, $t_0 = \{ 0:\{PM_{35}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_4(1)\} \}$ , $t_2 = \{ 1:\{PM_2, RM_4(3)\} \}$	
need decreasing distance between object 0 and object 1 but the distance actually increases, adaptation: add- move-into-path	objects={green virus:0, top dirt border ground:1, top dirt border ground:3}, $t_0 = \{ 0:\{PM_{35}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_4(1)\} \}$ , $t_2 = \{ 1:\{PM_2, RM_4(3)\} \}$	
need touch be- tween object 1 and object 3 while touch was not established before, adaptation: add- move-into-path	objects={green virus:0, top dirt border ground:1, top dirt border ground:3}, $t_0 = \{ 0:\{PM_{35}, RM_2(1)\} \}$ , $t_1 = \{ 0:\{RM_4(1)\} \}$ , $t_2 = \{ 1:\{PM_2, RM_4(3)\} \}$	

Table 19: Search tree for the message *the green virus rests on the platform and it moves to the box after it falls*

Error and Heuristic	ASR	Setup
initial, adaptation: none, initial call	objects={green virus:0, top center wrench head:1, iron block:3}, $t_0 = \{ 0:\{PM_{35}\} \}$ , $t_1 = \{ 0:\{PM_2, RM_4(1)\} \}$ , $t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \}$ , $t_3 = \{ 0:\{RM_5(3)\} \}$	
need touch be- tween object 0 and object 1 while touch was not established before, adaptation: add- move-into-path	objects={green virus:0, top center wrench head:1, iron block:3}, $t_0 = \{ 0:\{PM_{35}\} \}$ , $t_1 = \{ 0:\{PM_2, RM_4(1)\} \}$ , $t_2 = \{ 0:\{PM_{11}, RM_2(3)\} \}$ , $t_3 = \{ 0:\{RM_5(3)\} \}$	

<p>need decreasing distance between object 0 and object 3 but the distance actually increases, adaptation: add-move-into-path</p>	<p>objects={green virus:0, top wrench head:1, iron block1:3},  <math>t_0 = \{ 0: \{ PM_{35} \} \},</math>  <math>t_1 = \{ 0: \{ PM_2, RM_4(1) \} \},</math>  <math>t_2 = \{ 0: \{ PM_{11}, RM_2(3) \} \},</math>  <math>t_3 = \{ 0: \{ RM_5(3) \} \}</math></p>	
<p>need specific motor commands for object0, adaptation: set-motor-commands</p>	<p>objects={green virus:0, top wrench head:1, iron block1:3},  <math>t_0 = \{ 0: \{ PM_{35} \} \},</math>  <math>t_1 = \{ 0: \{ PM_2, RM_4(1) \} \},</math>  <math>t_2 = \{ 0: \{ PM_{11}, f_4 = -0.9998265, RM_2(3) \} \},</math>  <math>t_3 = \{ 0: \{ RM_5(3) \} \}</math></p>	
<p>need decreasing distance between object 0 and object 3 but the distance actually increases, adaptation: add-move-into-path</p>	<p>objects={green virus:0, top wrench head:1, iron block1:3},  <math>t_0 = \{ 0: \{ PM_{35} \} \},</math>  <math>t_1 = \{ 0: \{ PM_2, RM_4(1) \} \},</math>  <math>t_2 = \{ 0: \{ PM_{11}, f_4 = -0.9998265, RM_2(3) \} \},</math>  <math>t_3 = \{ 0: \{ RM_5(3) \} \}</math></p>	

---

Judging by the evidence presented here, we can say that the language system and the inference mechanism are able to do what they are supposed to do. There are certain limitations that would require further attention, as discussed in the cases that LEARNA did not resolve. On the positive side, note that LEARNA shows extreme flexibility: the rules for interpreting an utterance are compositional and not bound to predefined phrases. Prepositions are used correctly for different spatial patterns, even a direct object was understood correctly. Since LEARNA allows for free combination, the absolute number of utterance rises exponentially with every new lexeme: the above only showcased interesting examples, LEARNA is of course able to understand all possible *simple* sentences, e.g. *the virus moves*. This compositional flexibility should be noted.

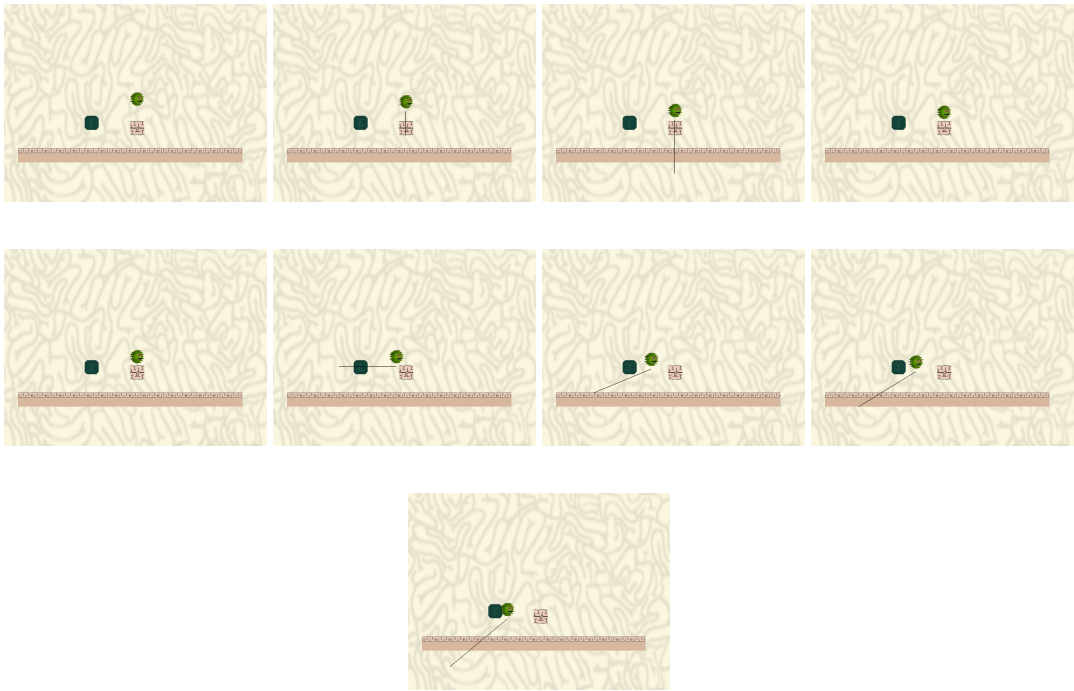


Figure 41: Simulation for the successful interpretation of the message: *the green virus rests on the platform and it moves to the box after it falls*

## 12.1 Limitations

One major limitation is that one word has to always be mapped to exactly one model, but the range of available models does not allow for combinations of models. For example, it is not possible to link ‘stand still’ to the combination of the ‘no-horizontal-movement’ model and the ‘no-vertical-movement’ model. Note, however, that such a model combination would arise in the EPS, and not in the mental lexicon. If the EPS provided a model for this combination, both the LPS and the ISS could immediately use it. A related limitation concerns the fact that the lexicon cannot link to specific features: there are no models in the EPS that encode *fast* or *slow*, and the top-down predictive encodings are skipped in favor of simple object types, so there are also no structures that encode *green* or *round*. The mental lexicon thus does not contain adjectives or adverbs. Such a mapping would also necessitate major changes in the ISS.

As discussed above, a major limitation of the ISS is the inability to infer abstract gaps: LEARNA can understand ‘robot pushes box and box is destroyed’ when the robot hits the box and the box is immediately destroyed. If the robot pushes the box off a platform and the box is destroyed as a consequence, however, the ISS will fail. The input sentence has to present a strict chronological succession without missing episodes. This leads to a more general limitation because the inference mechanism does not make use of the hierarchy of events provided by the EPS. Given an ASR, the inference mechanism does not check whether there is a scene-path in the space of transition models, something that may help with inferring such missing episodes.

In the current implementation, we assume that the succession in the ASR is exhaustive, so there is no need for filling gaps in the space of transition models. The ASR already represents a scene-path in the space of transition models: the transitions in the ASR directly follow *one another*. It seems plausible that human cognition makes use of such hierarchies between more detailed and more abstract encodings. Especially if there are more levels to the hierarchy than just the two. Generalizing the inference system to start with higher levels and then descend the hierarchy would be an important issue for future work.

Another issue arises because the ISS has to build a simulation for an input sentence in isolation. Neither other sentences nor the current environment can be referenced in a sentence. The LPS introduces new and unique object identifiers every time a sentence is processed and thus introduces new and unique objects. A more flexible assignment of reference would need to be implemented to include objects that are active in working memory, either through the surroundings or discourse. This is left to future work.

## 12.2 Resolving Ambiguity

LEARNNA, as opposed to theories of language processing, applies very little language-specific knowledge. The rules encoded by the LPS are very simple and processing biases are not used at all. Recall the discussion of ambiguity in section 3: human listeners plausibly use a range of different cues to reduce the need for inference. LEARNNA, on the other hand, employs world knowledge only since this is the focus of this work.

Interestingly, LEARNNA is able to handle different types of ambiguity beyond the pronoun ambiguity derived from WSSs, which it is able to resolve. Recall the utterances below which LEARNNA was able to understand correctly.

- (55) arrow flies to the box and it breaks
- (56) the box falls to the ground and it breaks
- (57) the cell consumes the bulb and it moves

In all cases, the pronoun is strictly speaking ambiguous. In (55), there are no examples of breaking boxes that an arrow collides with, but an arrow will break once it collides with a green box. In (56), the same is true of ground and red boxes. (57) is a further example where only the cell can move, not the bulb. Using the classification discussed in section 3, these are examples of ambiguity caused by the language system but resolved by world knowledge. Given the mini world BrainControl and how it behaves, there is only one possible interpretation of (56) and (57). (55) *could* also be interpreted differently. The arrow could fly to a *red box*, which subsequently drops to the ground and breaks. While this interpretation is much more complicated to simulate, it is generally possible. LEARNNA prefers the more straightforward interpretation, however.

Note that (55) and (56) also contain a *lexical ambiguity* in addition to the pronoun ambiguity. Box means movable red box on the one hand, an object in the world that can be pushed and that can break. Box also means *green block* on the other hand, a static object that is part of the game world and cannot move or change its state. One can argue that these two objects, due to being very different at their core, present a lexical ambiguity. An analogy in the real world could be *rock*, which could be used for a small rock that one can pick up, throw around, and move. Rock could also be used for a massive cliff, on the other hand. The meanings are related, but it could be argued that they are sufficiently different. This can be contrasted with the *referential ambiguity* in (57), where *cell* could refer to a red or a blue cell. Red and blue cells are two variants of the same type, behaving nearly identical. Only in specific cases is the difference even relevant; it could be argued that in (57), this is only underspecification but not ambiguity.

Even if the two types of boxes are deemed too similar to qualify as a lexical ambiguity, the ability of LEARNNA to resolve a real lexical ambiguity is shown. Finally,

reconsider example (58). The timeline of (58) is ambiguous because of the *scopal ambiguity* introduced by *and* and *after*. If *after* takes scope over *and*, the virus falls first, rests, and finally moves. If *and* takes scope over *after*, on the other hand, the virus first rests, then falls, and finally moves. Unfortunately, world knowledge does not strictly resolve this ambiguity, both interpretations are still viable. LEARNA decides on the first interpretation above, i.e. the virus is simulated as falling first, then resting, and finally moving. Note, though, that in principle, if one of the interpretations was prescribed by world knowledge, LEARNA would build the scene reconstruction accordingly.

(58) the green virus rests on the platform and it moves to the box after it falls

Even though the model implemented in this work is only a very coarse approximation of language processing, there is a range of ambiguities it can resolve. Note that resolving ambiguity is actually a byproduct of the system, i.e. there are no specific mechanisms in place to specifically resolve ambiguity. An ambiguity in the language system is simply translated into competing interpretations (ASRs); there is little more to our implementation in this regard. The main work is done while inferring scene reconstruction using world knowledge, which is the core contribution of LEARNA.

### 12.3 Language and Event-Predictive Structures

The evidence above suggests a few conclusions:

1. LEARNA *can* understand WS-like utterances as long as they adhere to the mini world of BrainControl, if understanding is taken to mean *being able to build a scene reconstruction*.
2. LEARNA is able to process utterances in a very flexible manner, allowing for compositionality. Even though the syntax of possible utterances is restricted, improving the capabilities of the LPS should be simple and is *independent* of both the ISS and the EPS.
3. LEARNA requires very little pre-defined structure; extending the EPS with more powerful models would immediately transfer to LEARNA as a whole without further adjustment.

It seems justified to say that LEARNA satisfies the requirements that we established in this work, specifically our first main hypothesis.

While LEARNA operates on a simple mini world, it can serve as an example of how event-predictive structures lend themselves to an implementation of language processing in principle. Information gaps are filled as necessary if possible. At the same time, the structures provided by the EPS provide a grounding for language processing, as is demanded by Harnad (1990), see also section 4.2: instead of building a language

processing system top-down, starting with symbols and then adding meaning, the system should be built bottom-up, starting with meaningful entities and then adding symbols. This does not imply that the symbol grounding problem is *solved*, because Harnad (1990) was mainly concerned with complexity when scaling a system to the real world. It can be taken as evidence, however, that event-predictive structures *facilitate* language grounding.

One of the most interesting lessons of LEARNA is that the process of *inference* is really difficult and time consuming. Ideally, the LPS should determine as much of the meaning as possible. On the other side, the EPS should ideally provide structures that make inference easy, even in complex cases. Being located on the interface between cognition and language, however, the inference processes seem to get too little attention when modeling language processing.

## 13 Experiment: Producing Information Gaps

<sup>41</sup>As an excursion from modeling comprehension, we will look at actual *human language production* in this section. We build on the discussion of the utility of ambiguity and information gaps in section 3.5. Recall that an information gap can be useful if it is easy to fill by the listener and also shortens the utterance, meaning it is less costly for the speaker. When producing an utterance, the speaker has to judge whether an information gap is easy to fill by the listener, otherwise they will risk inference failing and being misunderstood. As should have become clear over the course of this work, world knowledge is a very broad concept including a lot of very basic information that can be assumed to be shared across all listeners. In the following, we will describe an experiment where participants acquire world knowledge and then produce descriptions of scenes that either conform to what they learned or violate the formerly learned patterns. We predict that speakers are less likely to produce information gaps in cases where learned world knowledge is violated.

Information gaps and world knowledge have already been explored in this work. For world knowledge, we refer to the discussion in section 3.5. We assume event-cognitive models as they are introduced in section 5 as the form of representation. In order to have sufficient theoretical background, we will thus only need to discuss language production and how the three topics are linked.

### 13.1 Language Production and Predictability

Inspired by WSs, we will focus on reference production. Reconsider one of the guiding examples in this work, (4), repeated below. (4) is strictly ambiguous, but can be tweaked to be unambiguous, as in examples (59a) and (59b).

- (4) The box falls to the ground and it breaks.
- (59) a. The box falls to the ground and breaks
- b. The box falls to the ground and the box breaks

From the perspective of ambiguity and efficiency, speakers should *always* favor (59a), because there is no ambiguity and it is also the shortest utterance of the three. If we assume, however, that dropping the subject in the second part as in (59a), a form of ellipsis, necessitates inference, i.e. produces an information gap, (59a) should only be favored in cases where the inference is simple. As discussed, this depends on world knowledge, i.e. how likely it is that the box breaks. The experiment we describe here provides evidence that the latter is the case, but there is also existing empirical evidence that supports that prediction.

From the investigation of ambiguity in production, there is evidence that speakers are not as good at avoiding ambiguous statements as might be expected. Ferreira et al.

---

<sup>41</sup>This chapter has been reworked and published in Stegemann-Philipps, Butz et al., 2021



(2005) tested production on a simple naming task where participants had to point out a marked object in the presence of competitor objects. Theoretically, it is ideal to use a referring expression that is as short as possible while not being ambiguous, i.e. not including any competitor objects. Ferreira et al. (2005) indeed found that participants were likely to add extra detail if a small *bat* (the flying mammal) was to be pointed out but another, larger bat was a competitor item. In such a case, participants might produce *small bat* instead of simply *bat*. If there was only one mammal bat, however, and one of the competitor objects was a baseball bat, the participants were less likely to include extra detail, i.e. produce *bat*, even though their utterance is ambiguous in this case and they risk misunderstanding.

This has been taken as evidence that there are many factors that influence language production, but speakers are not very sensitive to potential ambiguity, i.e. the effect that potential ambiguity has on production is rather small, see also Wasow (2015). Haywood et al. (2005) asked participants to describe spatial setups, producing referring expressions like 'Put the penguin in the cup on the star'. If there are two penguins in the scene, this utterance is ambiguous. To make it unambiguous, one can say 'Put the penguin that's in the cup on the star'.<sup>42</sup> Haywood et al. (2005) found evidence that speakers are more likely to include an extra 'that's' when describing scenes where this reduced ambiguity, i.e. speakers seem to be sensitive to the potential ambiguity. Interestingly, the difference was rather small between scenes that allowed for one interpretation only and scenes where both readings were plausible. A much larger effect was *priming*, i.e. the occurrence of a similar structure before the utterance is produced. In this case, priming may happen because the conversation partner already used 'that's'. This actually made the speaker use this cue much more than the potential for ambiguity.

If this evidence is correct, we can expect that speakers may reduce the use of the (ambiguous) pronoun and choose unambiguous utterances from the list in (59). This, however, should only be a small effect. There also is empirical evidence that speakers will adjust utterances on how *predictable* something is. Jaeger (2010) tested this for utterances like '[m]y boss confirmed (that) we were absolutely crazy'<sup>43</sup> and found evidence that speakers include the complementizer *that* depending on predictability. If *we* is highly predictable, speakers will tend to not produce the *that*. If *we* is less predictable, speakers will tend to add a *that*. Jaeger (2010) takes this as evidence that speakers actively smooth the *information density* of an utterance, i.e. the rate of surprising information, calling this the *uniform information density hypothesis*. Because it is difficult to judge how predictable or surprising words in a sentence are, Jaeger (2010) approximates this with next-word predictions learned statistically from a corpus. Kurumada et al. (2015) found further evidence that speakers of Japanese will drop case markings if they are predictable. They use utterances that describe scenes and

---

<sup>42</sup>Examples taken from Haywood et al. (2005)

<sup>43</sup>Jaeger (2010), p. 27

find that scenes that seem more surprising make speakers use more case markings. Adjusting the use of case markings is a very common finding (see e.g. Lee (2015) for a perspective on Korean case marking). The evidence that Kurumada et al. (2015) present is especially interesting in our case because their setup targets predictability from world knowledge.

We can take this as evidence that speakers actively manage the amount of information depending on world knowledge, which is an otherwise elusive factor. Instead of world knowledge, experiments usually focus on predictability by itself, which is a very complex term (see Kuperberg et al. (2016) for a discussion). The problem with using predictability as a proxy for world knowledge is that there are many other factors besides world knowledge to shape expectations about upcoming information. There is a body evidence of production biases in pronoun production which points to a more diverse range of factors. For example Rosa et al. (2017) present evidence to argue that thematic roles influence the frequency of producing a pronoun, while Fukumura et al. (2010) found no such evidence. Fukumura et al. (2011) found evidence that similarity between potential competitors in terms of reference affects pronoun production, i.e. speakers are less likely to produce a pronoun, the more objects it can potentially refer to. Rohde et al. (2014) provide evidence that speakers are more likely to produce pronouns if the referent is the topic. These factors are not directly dependent on world knowledge, however. We can take this as evidence that corpus-based predictability measures will represent a mixture of all the factors that influence production rather than world knowledge. One of few studies that probe world knowledge more directly is described by Bunger et al. (2013). They indeed found evidence that underlying event-structures of scenes can influence production. This goes deeper than purely language-related cues; abstract event structures can be seen as world knowledge in the sense that they represent abstract knowledge about how events unfold.

A final question in forming a prediction of whether speakers will adjust their production according to world knowledge in a specific situation is whether they take into account what world knowledge the listener has. Whether speakers adjust their utterances based on their audience is the question of *audience design*. There is mixed evidence as to whether speakers do this successfully in spontaneous speech. Ferreira (2019) provides a survey on this question. Interestingly, Lane et al. (2006) provide evidence that participants are not very good at taking situation-specific knowledge into account. Even when asked not to reveal a certain piece of information to the listener, speakers tended to produce utterances that are (indirectly) revealing anyway. On the other hand, adjusting our language to our audience seems an everyday experience. Kingsbury (1968) provides evidence that speakers adjust their description when giving directions based on whether the listener seems to know the local area or not. Finally, Fukumura et al., 2012 present evidence that in producing pronouns, speakers do not account for the knowledge of the listener but simply rely on their privileged knowledge. We will assume that speakers use their own world knowledge as an approximation of

the world knowledge of the listener and thus of how easy inference will be for them.

In summary, there is evidence that, while speakers *can* avoid ambiguity, this effect is rather small. There is also evidence, however, that speakers drop or insert information into utterances based on predictability. Since speakers have, depending on the situation, limited or no access to what the listener judges predictable, there is evidence that this does not influence production very much. In our experiment, we want to propose a setup where the dependence on world knowledge is immediate.

## 13.2 The Experiment

To find out whether world knowledge influences production, we need to make sure that participants have equal and reliable world knowledge which is influenced by prior knowledge as little as possible. We further need to let the participants produce utterances that will show this influence if it exists. Traditionally, this is done in a *story continuation setup*. For example, Rosa et al. (2017) show participants pictures of real world events and provide incomplete sentences that the participants have to continue. This restricts the range of possible utterances but also limits free production. While this approach is well established, we choose a different approach.

We use the mini world of BrainControl, which was also used for modeling in this work, to confront participants with an environment they are unfamiliar with. We reduce the number of entities to three: a long blue cell, a spiky red cell and a round green virus which has a face (see the description of the experimental setup below for images). These entities have a simple base behavior in that they can move left, move right, or rest. If a green virus and one of the cell types get to close, they will pause, and one of the two will rush towards and bump into the other, bouncing off slightly. After a very short pause, one of the two will run away at an elevated speed. We will call this a *collision event*. After this collision event, the entity that stayed behind will resume its base behavior. The only pattern to this is that the green virus will run away from the red spiky cell, but the long blue cell will run away from the green virus. The two cells do not interact. In a *training phase*, the participants are shown videos where this behavior can be observed. Ideally, participants will be familiar with the simple patterns after the training phase. At this stage, we can assume that they *expect* events in the mini world to follow these patterns, i.e. they have acquired world knowledge. Additionally, we can test whether they are able to make predictions according to these patterns to be certain that they learned them.

To let the participants produce utterances like those in example (59), we simply ask them to describe short videos depicting a collision event using *spoken production*. Due to the nature of the event, participants will first have to describe that one entity attacks another and then describe that one of them flees. This two-part structure is supposed to make the participant produce one of the descriptions in (60) or a variation thereof.

- (60) a. The red cell attacks the virus and the red cell flees.



Figure 42: Entities populating the videos used in the experiment.

- b. The red cell attacks the virus and it flees.
- c. The red cell attacks the virus and flees.

While the wording and the involved entities can vary, such a description can then be annotated. (60a) would be an example of NP (noun phrase), i.e. the speaker decided to produce a full noun phrase in the second part. (60b) and (60c) would both be examples of a shortened phrase, where the speaker decided to use an ambiguous pronoun or an ellipsis to shorten the utterance. If the speaker does not use a two-part description or deviates otherwise from the expected structure, we cannot annotate that utterance and dismiss it. Since participants use free spoken production, their utterances are unbounded and immediate.

### 13.3 Experimental Setup

The participants start with being shown a short introduction and instructions. They are told that they will have to describe videos from a video game that they will be shown by talking into their microphone. They are asked to describe these videos as if they are talking to friend, i.e. not overly formal.

Each participant then has to do a microphone check. We found in pilot studies that this significantly reduces the number of unusable recordings.

The participants are then shown pictures of the three entities they will see in the videos (see figure 42) along with a description of their behavior.

This is followed by the first *training block*, which starts with videos that each show a possible interaction. These interactions are following the patterns of the world, there are four possible interactions.

1. Green virus attacks blue cell, blue cell runs away
2. Green virus attacks red cell, green virus runs away
3. Red cell attacks green virus, green virus runs away
4. Blue cell attacks green virus, blue cell runs away

After being shown the four interactions, they see a larger video with three layers, where one interaction takes place on each layer.

attacking entity	standing entity	fleeing entity	subject / object fleeing	surprise
green virus	blue cell	blue cell	object	false
green virus	blue cell	green virus	subject	true
green virus	red cell	green virus	subject	false
green virus	red cell	red cell	object	true
blue cell	green virus	green virus	object	true
blue cell	green virus	blue cell	subject	false
red cell	green virus	green virus	object	false
red cell	green virus	red cell	subject	true

Table 20: Stimuli used in the experiment. Object / subject refers to whether the fleeing entity, for which the produced referring expression is relevant, will usually have been in object or subject position in the first part, i.e. whether it was being attacked or attacking. Surprise denotes whether the stimulus follows the patterns shown and learned in the training phase (false) or reverses the pattern (true).

After this training block, they are asked to predict the patterns of the world in a *test block*. There they are shown a starting image and two possible outcomes. The starting image shows two entities before interacting. Only one of the outcomes is plausible. For example, if the starting image is a red cell and a green virus facing off, then a picture of green virus only and red cell only show possible outcomes. If the participants are able to predict the correct outcome on two such test cases, they continue with the experiment. If they are not, they do another training block followed by a test block. If they are still not able to predict the outcome correctly, they continue with the experiment but their results will be removed from the data.

After this training and test phase, they are shown another set of instructions for the *stimulus phase*. They are asked to describe what they can see by speaking into their microphone. Details of what is shown to the participants can be found in appendix C.

In the stimulus phase, each participant is shown a randomized sequence of stimuli and fillers. Fillers are videos where either one or two entities are present and moving, but no interactions happen (see appendix C for details). There are eight stimuli, the four interactions described above that follow the patterns of the world, and their unexpected version, where the object running away is exchanged. The stimuli can be seen in table 20.

All videos in the stimulus phase are connected by a short 2 second still image showing the word *break*.

## 13.4 Participants

We recruited 300 participants through the online platform *Amazon Mechanical Turk*. The complete experiment is conducted online. The validity of data obtained from Amazon Mechanical Turk has been investigated (see e.g. Schnoebelen et al. (2010)), and it is by now a very common tool for data collection. The only exclusion option provided by Amazon Mechanical Turk that was used for selecting the participants was that they are located in the US as far as Amazon Mechanical Turk is aware. This was done mainly to restrict the participants to English speakers.

We excluded participants where the audio recording could not be transcribed. We further excluded participants that did not do the correct predictions in the test phase, assuming they had not learned the patterns of the mini world.

## 13.5 Annotating Utterances and Analysis

The audio recordings were cut to the part where the description of the stimuli started. They were then transcribed by a transcription company in Stuttgart. Within the transcription, the utterances describing stimuli were extracted and then annotated by a student assistant and the author of this work. Possible annotations were *NP* for noun phrase, *pro* for pronoun and *zero* for ellipsis, (see example (60)), as well as *X* if the description did not fit one of the categories. Reasons for annotating *X* were one as follows; details can be found in appendix C.

1. The utterance had a format that did not allow for choice between pronoun, noun phrase or ellipsis. Examples of this were utterances where pronouns were used before or where the first part did not have a clear subject-object-distribution (e.g. virus and cell attack each other).
2. The utterance generally deviated from the expected schema, e.g. the cell defeats the virus.

For analysis, the annotation groups *pro* and *zero* were combined into the group *shorten*. Further, we aggregated the 8 stimuli described above into 4 groups, i.e. *object-surprise*, *object-predicted*, *subject-surprise*, *subject-predicted*. For each of these groups, each participant was shown two stimuli, one where the green virus interacted with a blue cell and one where the green virus interacted with a red cell (compare table 20). This was done to prevent within-participant comparison for two stimuli of the same type. To aggregate the annotations for each group into one value, we used the aggregation table shown in table 21.

Aggregating the two variants for each stimulus type produces what we call a shorten-vs-NP preference value. If the value is 1, the participant preferred NP for this stimulus type, if the value is 0.5, the participant was undecided, if the value is 0, the participant preferred to produce a shortened utterance. This value is thus ordered, with

stimulus 1	stimulus 2	value
NP	NP	1
NP	X	1
NP	shortened	0.5
shortened	X	0
shortened	shortened	0
X	X	excluded

Table 21: Aggregation table to compute a shorten-vs-NP preference value for each stimulus type. This table only shows ordered aggregation; permutations are aggregated equally.

1 being a preference for NP down to 0 being a preference for shortening; the exact scale, however, is meaningless. The main predictor is whether the stimulus was surprising or not. To model the data, we use an ordinal regression model (see e.g. Agresti (2013)), specifically the implementation of cumulative link mixed models implemented in the *ordinal* package in R (see Christensen (2019)). We include participants as random intercepts.

### 13.6 Results and Discussion

Out of initially 300 participants, each provided with 8 stimuli, i.e. 2400 items, we had to exclude 31 participants because of bad audio quality and 25 participants because they did not pass the learning test block. We further excluded 11 participants because their descriptions did not conform to our schema at all, mostly because they produced predictions instead of descriptions. This leaves 233 participants, i.e. 1864 items. Looking at the overall frequencies of annotations in figure 43, it becomes clear that participants barely use shortened phrases when referring back to the object. This is to be expected because an ellipsis is strictly speaking ungrammatical, i.e. it cannot refer to the preceding object but only the subject. Further, there is a very strong bias towards referring back to the subject when using a pronoun (see also the discussion of pronoun biases in section 3.3). Participants thus seem to have a strong preference not to use pronouns to refer back to the object.

Note further that for subject-type stimuli, participants produce less pronouns than ellipsis in general, but the raw frequency difference between the expected and the surprising case is very similar. The switch from shortened phrases to noun phrases seems to not be fueled by one type of shortening the phrase, but is distributed equally (compare also figure 44).

It is important to note that we found a strong difference between the *first* stimulus and all later stimuli, as seen in figure 44. Note that we excluded object-type stimuli in this case because participants mainly produced noun phrases for those. Participants

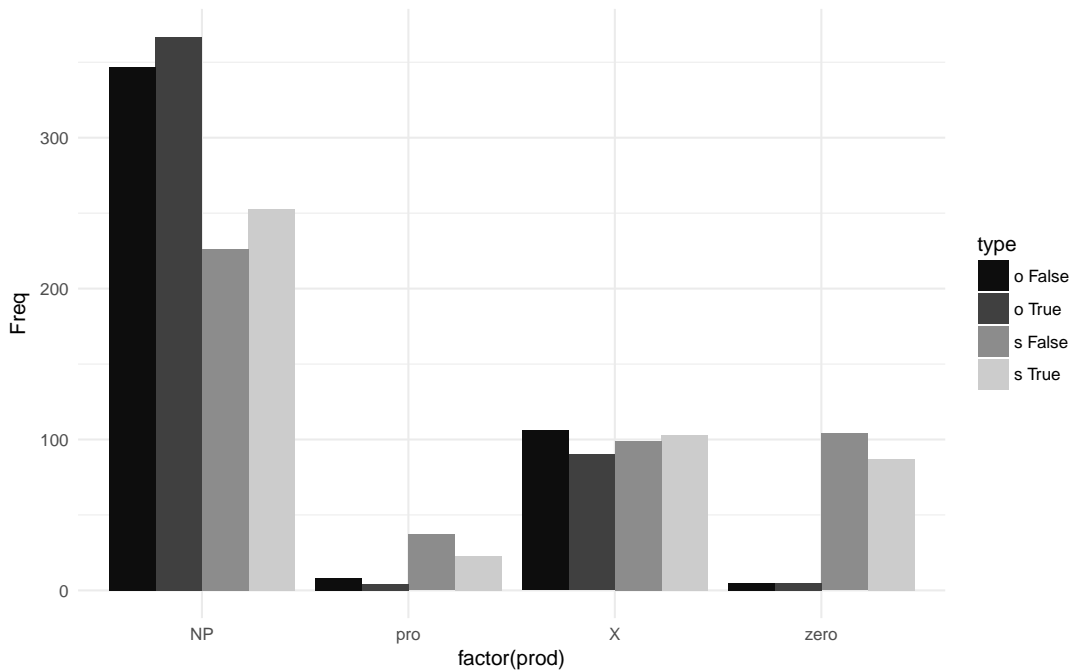


Figure 43: Distribution of different annotations split by surprising(True) vs predictable(False) and subject(s) vs object(o) position.

seem to be much more likely to produce shortened phrases for the very first stimulus. The order of the 8 stimuli was randomized for each participant but we did not ensure a uniform distribution of orderings, as can be seen in figure 45, where object-stimuli were again excluded. This also shows however that the distribution is not biased in a specific sense, i.e. we can still assume the stimuli ordering to be random.

Using the aggregation method in table 21, an item was dropped if a participant did not produce an utterance that could be annotated for a stimulus. This was true of 104 items, resulting in 828 items in the aggregated data set. Finally we excluded object stimuli since participants mostly produced noun phrases for object stimuli. This brings the number of data points in the final data set down to 412, stemming from 219 participants, i.e. 14 participants did not produce utterances that could be annotated for subject-type stimuli. For the final data set, the distribution of shortened-vs-NP preference values for surprising and predictable stimuli can be seen in figure 46. We fitted the model as described above and it showed a significant interaction ( $\beta = 0.5121$ ,  $std. error = 0.2229$ ,  $p = 0.0216$ ),  $\beta$  and  $std. error$  for surprise being *true*.



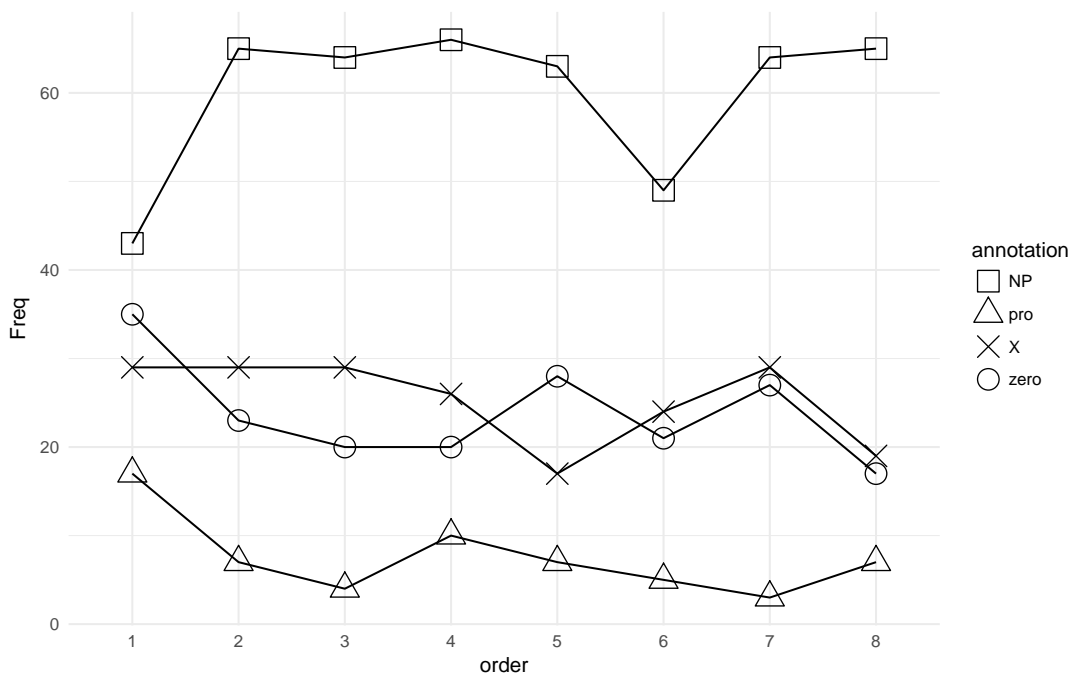


Figure 44: Distribution of different annotations vs the ordered position of stimuli. Participants are much more likely to use shortened utterances for the first stimulus while being balanced for the rest. Note that this data shows only subject-type stimuli.

### 13.7 General Discussion

The evidence presented above suggests that speakers actively manage whether to leave an information gap in an event description, depending on whether the development of the event is surprising or expected, as was our hypothesis. This is in line with our general prediction that speakers manage information gaps in their utterances. This is especially interesting because it is plausible that this behavior was triggered by world knowledge alone.

We can assume that our experiment was free of biases that stem from language statistics like word-to-word predictability. It is very unlikely that the participants had heard or produced these utterances before beginning our experiment, since the setup is artificial. Consider the example utterances (61) taken from the data. However, the only related sentence that the participants read in the instructions is shown in (62), and we explicitly avoided priming utterances that we were plausibly expecting given pilot study results. In everyday language, the beginning of each utterance in (61) may elicit continuations describing *destruction* or *dissolution*, but cells running away should be uncommon. While we did not test this specifically, we assume that participants were not biased by the thematic setup they saw and described.

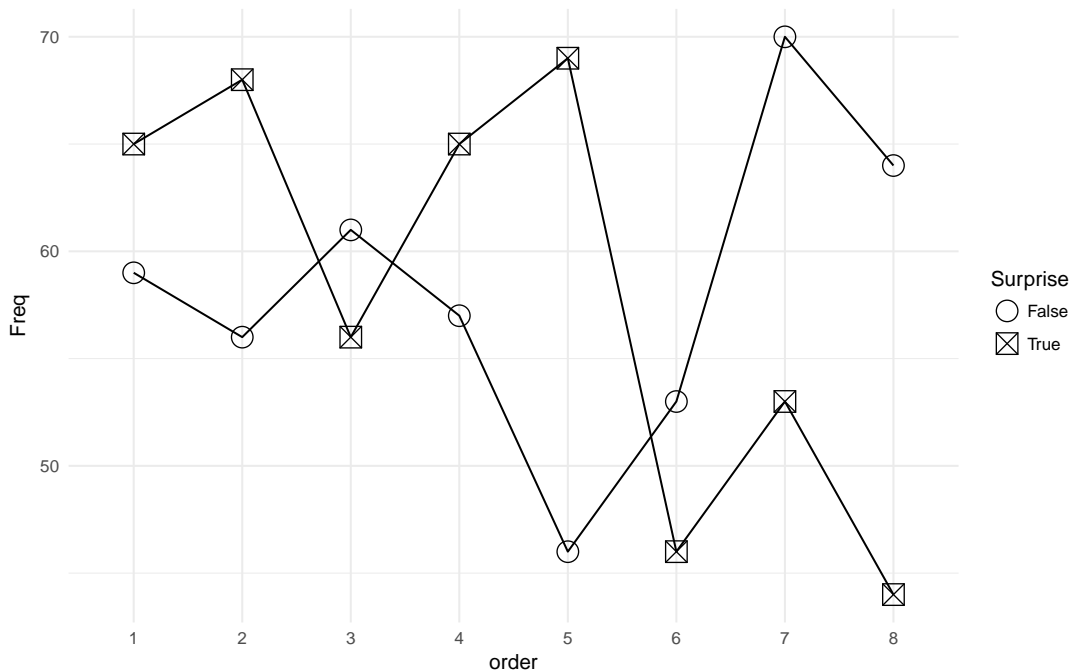


Figure 45: Distribution of stimuli types vs the ordered position of stimuli. This shows that there is no *specific* bias in the distribution; the first stimulus shows no real difference to the other stimuli. This also shows, however, that the overall distribution has a lot of variation. Note that this data shows only subject-type stimuli.

- (61) a. The red cell just attacked the virus and it ran away.  
 b. The virus attacks the blue cell, the blue cell runs away to the left.  
 c. The person attacks the virus, the virus leaves. Person wins.  
 d. Virus eats the red blood cell and then the virus runs.
- (62) Let us first look at interactions between a virus and a cell. Notice how it doesn't matter who attacks whom.

Further, we can assume that heuristics or biases like implicit causality did not play a major role either. The big language specific factor in our data was subject vs object position, which completely dominated production and had to be removed. If there are other heuristics that impact the participant's production decisions, these should further have the *same influence* across surprising cases and predictable cases. If, for example, *attacking* is more likely to be followed by an ellipsis than *eating*, this effect should influence the general occurrence of ellipsis, but not depend on whether the scene is surprising. While these factors are hard to measure, we can assume that their impact on our data is negligible.

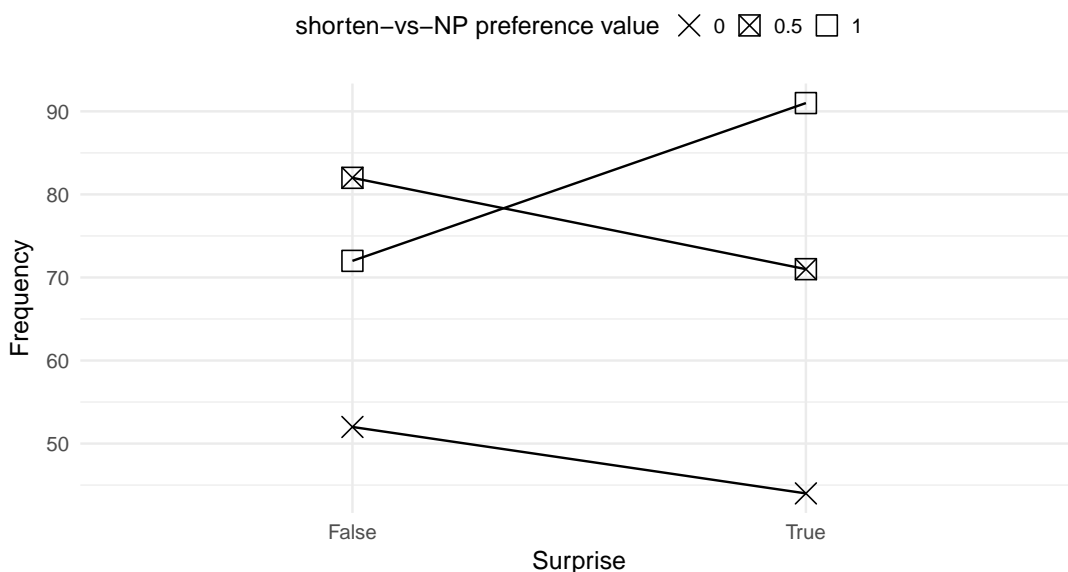


Figure 46: Difference between surprising (surprise=true) and non surprising (surprise=false) stimuli, for each shorten-vs-NP preference value. In the surprising case, participants use shortened utterances (value = 0 and 0.5) less often and noun phrases more often (value = 1). Note that there are exactly 206 surprising items and 206 non surprising items in the final data set.

The only factor that remains to explain the behavior in our specific case is *world knowledge*. This means that in the process of production, world knowledge and specifically event-predictions directly impact information gaps in the produced utterance, an effect that has not been researched to the best of our knowledge. While our experiment does not show this directly, it may be the case that speakers are actually more sensitive to information gaps in general than to ambiguity. The fact that ellipses *and* pronouns were less preferred in surprising cases may point to this, since from the perspective of ambiguity, there is no reason to avoid ellipsis even in the surprising case.

The results are also in line with the assumption that speakers use their own world knowledge as a proxy of the world knowledge of the listener. This also shows that further research is necessary that also includes dialogue to further examine how speakers adapt to the perceived world knowledge of the listener.

Finally, if the assumptions above are correct, our experimental setup may be useful as a general paradigm to examine the influence of world knowledge on language, a question at the intersection of psycho-linguistics and cognitive science. Participants adapted surprisingly well to the scenes they were shown. It is especially interesting that the majority of utterances produced were in line with our expectations, without

priming utterance structure or prescribing parts of the utterances; the production was completely free. In the pilot studies we found, however, that there was a huge difference in quality between free spoken production and free written production, which may be because of the online setting of our experiment. There is much potential for future work in these directions.

## 14 Conclusion

In summary, this work provided evidence for both main hypotheses. The structure that emerges in LEARNA from a simple event-predictive model lends itself to language processing, allowing for an easy lexical mapping between lexemes and learned structures. This effectively provides a grounding for language. The structures further allow to build an inference mechanism, which can build reconstructions of the meaning of utterances. In addition, through building a reconstruction and thus understanding an utterance, LEARNA is able to resolve ambiguity through world knowledge where possible.

Recall that this work started out with a description of SHRDLU, the language understanding system described by Winograd (1972). LEARNA is similar in spirit but implemented in a completely different manner. Whereas SHRDLU was mostly hard coded, LEARNA is designed as three systems that can be extended flexibly and show little dependencies on the exact nature of the others. World knowledge is learned and inference is done in a general and flexible manner. In principle, LEARNA presents a step forward. Even though language modeling may be dominated by deep neural network approaches, a transparent and modular system like LEARNA has the advantage of illuminating difficulties of theoretical ideas and examining the effect of visible structure. Especially the apparent absence of models of inference at the intersection between a predictive model of the world and a language module raises interesting questions for future research. The main question may be whether adaptation heuristics can also result from a general learning mechanism as proposed by PP and event-predictive theories. This will be left for future work.

Another pressing point are situation encodings, i.e. the patterns of context. When extending LEARNA, situation encodings may be the most profitable path. Better situation encodings would immediately increase the predictive power of transition models, allowing to learn an increasing range of events beyond those that this work focuses on. It seems that situation encodings are also a bottleneck in terms of the complexity that the overall system can predict. Recall that Harnad (1990) argued that the problem with grounding symbols is really the complexity of compositional expressions, i.e. the complexity of scenes where several objects are interdependent. Situation encodings aim at this interdependence.

The original set of WSs of course requires much broader world knowledge than is necessary in a mini world. Nonetheless, the problem of applying world knowledge to infer the otherwise ambiguous reference of a pronoun was successfully approached by LEARNA.

The work also provided empirical evidence that world knowledge in the form of event predictions influences language production. Speakers were more likely to shorten a description of a scene if that scene conformed to their expectations. In cases where world knowledge was violated, speakers tended to prefer more explicit descrip-

tions. Given how little the influence of world knowledge on language production is investigated in the literature, the experiment showed that its general experimental paradigm could lead to many more fruitful investigations. It seems plausible that speakers adjust their utterances in a multitude of ways in the face of surprising events. This raises the broader question of how far this influence reaches. For example, subject vs object position had a much greater influence than world knowledge. Testing the limits and exploring the influence of world knowledge should provide many interesting opportunities for future work.

## Abbreviations

**ASR** Abstract scene representation.

**EPS** Event-predictive system.

**ISS** Inference simulation system.

**LPS** Language processing system.

**PP** Predictive processing.

**WS** Winograd Schema.

## General Glossary

**Algorithmic Level Of Description** This is one of the Marr's three levels of description, also called representational level. It answers the question of how input and output are represented and what the specific algorithm is that maps input to output. On this level, the formalism is described that defines the internal algorithmic mechanics of the system Marr, 2010; Rescorla, 2020.

**Artificial System** A behaving entity that can be described as a set of models and mechanisms. These work together to exhibit a certain desired behavior. This definition is specific to this work. A chess program operating on a chess board would be a system because it produces moves (behavior) on a chessboard. A function that adds two numbers is not a system because it only produces a sum (not a behavior).

**Cognition** The overall system of processes that are responsible for processing the world and generating behavior in humans, i.e. the human mind from a mechanistic perspective. Cognition is here defined as an information processing system.

**Common Sense Reasoning** The application of world knowledge to solve problems. The efficient and appropriate application of world knowledge has to be kept apart from simply having access to world knowledge.

**Computational Level Of Description** One of Marr's three levels of description, answers the questions of what a computation is supposed to achieve and why it is done. Since Marr's three levels of description are defined in terms of information processing, the computational level should describe notable features of the mapping from input to output Marr, 2010; Rescorla, 2020.

**Encoding** An encoding carries information about the world of an artificial system and can be accessed internally by the system. An encoding can either be a model, capturing information about dynamics, or simply information about the world, e.g. about features of objects. An encoding always captures patterns of the world while a representation captures information about the current state of the world.

**Hard Coded** If a part of a system is *hard coded*, that part is unresponsive to changes in the environment and cannot be altered. When discussing learning artificial systems, hard coded behavior means that behavior is *not learned*.

**Implementation Level Of Description** One of Marr's three levels of description, answers the question of how the physical implementation is actually realized Marr, 2010; Rescorla, 2020.

**Information** In this work, something carrying meaning about the state of the domain a system operates on.

**Information Processing** Denotes that an artificial system is perceived as a mechanistic function of mapping input to output.

**LEARNNA** A loose acronym for *Learning Event Abstractions to Resolve Natural language Ambiguity*. Short name for the overall system implemented in this work.

**Mapping Language To Cognition** The process of identifying encodings within cognition that a natural language concept or a combination of concepts can be linked to, i.e. the meaning of a concept or a combination of concepts as *encoded* in cognition. This process should then work for all natural language concepts and cover compositional meaning. The symbol grounding problem proposes that this is highly non-trivial.

**Marr's Three Levels Of Description** A way of dividing the analysis of an artificial system information processing introduced by Marr (2010). See computational level of description, algorithmic level of description and implementation level of description.

**Mechanism** An operation within an artificial system that is not a model. Mechanisms operate on models and encodings. A chess computer may have models of legal moves but needs mechanisms that apply those models in a specific situation.

**Mental Model** An entity proposed to explain how cognition works. Refers to a model of an aspect of the real world used within cognition.



**Mental Simulation** A process within cognition where mental models are used on representations to simulate the dynamic development of the corresponding scene in the real world. This includes all aspects of the scene, i.e. not only spatial aspects.

**Mini World** A computer-simulated environment that shares certain features with the real world but simplifies it. This allows to test artificial systems in a complete but simpler world. Once the artificial system works, more complexity can be added.

**Model** A description of specific world dynamics formulated in a way to be able to predict those dynamics. A model's predictions should ideally match the dynamics it describes. A mathematical formula describing gravitational forces is a model of real world gravitation. Given a falling apple at some point in time, the formula produces predictions of the acceleration of the apple in the real world. A function that adds two numbers is not a model per se because it does not describe real world dynamics. The input, output and internal states of a model can be representations but also encodings.

**Representation** An entity within an artificial system that carries information about the current state of a real world entity and is used internally by the artificial system. A representation *stands in* for an entity of the real world. As opposed to encodings, a representation deals with the current state and not with learned patterns.

**Scene** A temporally and spatially bounded chunk of a world. For example, an apple falling from a tree could be a scene. In this work, a scene is a sequence of situations.

**Situation** A single timestep of a scene. A situation can be understood as a scene that is frozen at a specific point in time.

**Spatial Relational Encoding** An encoding that captures a recurring pattern of relations between objects, i.e. encodes patterns in spatial representations.

**Spatial Representation** A representation that mirrors the spatial setup of the real world entities that it corresponds to. A map or a picture are spatial representations in that sense. A list of city names with their respective distances is not a spatial representation because the spatial setup in the real world is not part of the representation, e.g. a triangle of three cities is not found in the list.

**Symbol Grounding Problem** The symbol grounding problem proposes that *mapping* encodings of an artificial system to natural language concepts, i.e. mapping language to cognition, is highly non-trivial Harnad, 1990.

**Training** The process of feeding data to an artificial system that has mechanisms to form encodings of patterns in the data. The goal of training is the formation of useful, i.e. accurate, models.

**Winograd Schema** A sentence pair similar to: *The city councilmen refused the demonstrators a permit because they [feared/advocated] violence.* The pronoun in the second part changes reference depending on which verb is inserted.

**Winograd Schema Challenge** The challenge to build a computer program that is able to resolve the reference in Winograd Schemas. A successful answer to the challenge would allegedly approximate human-like common sense reasonings abilities.

**World Knowledge** Knowledge about how the world works, i.e. how the dynamics of the world will usually unfold. Having a piece of world knowledge corresponds to being able to predict effects in the world implied by that piece of knowledge. For example, knowing that water is fluid and knowing what fluid means enables a person to predict that water will spill on a table if not held by a container. World knowledge can be distinguished from Semantic Knowledge and Syntactic Knowledge in this work.

## Cognitive Modeling Specific Glossary

**Bottom-Up Processing** The principle of processing an input encoding by passing its information onwards and changing the internal states of the artificial system accordingly. Opposite of top-down processing.

**Event** In this work, an event is a chunk of space-time where the unfolding dynamics are uniform in the sense that they follow a single specific pattern. An event always has a start and an end, i.e. event boundaries.

**Event Boundary** In this work, the point where one event ends and another begins, i.e. where the pattern governing the unfolding dynamics changes.

**Generative Model** A model that is able to actively predict encodings of the domain it operates on based on its current state.

**Prediction Error** Given a goal encoding, the prediction error is the difference of the prediction of that encoding and the actual encoding.

**Predictive Processing** A view within cognitive science that proposes that cognition can be modeled as a generative model with functional layers that predict each other in a top-down fashion, see top-down processing and processing hierarchy.

**Processing Hierarchy** A hierarchy of layers that exhibit a flow of information to support bottom-up processing or top-down processing.

**Situation Encoding** A learned pattern of the state of a scene at a specific point in time. Situation encodings are used by transition models. They include a main object to which the transition model is applied, i.e. the event dynamics models are switched. They also include objects directly touching this main object, learned by object type. Feature values are learned as gaussian distributions for these objects.

**Timestep** Since a mini world is simulated on a computer, the states can not be updated infinitely often but only a certain amount of times per second, say  $n$ . Each of these updates is a timestep.

**Top-Down Processing** The principle of generating the internal state of the artificial system, predicting what input would match this state and reacting to the difference of the prediction to the real input, i.e. the prediction error. Opposite of bottom-up processing.

## Language Specific Glossary

**Ambiguity** The phenomenon of an utterance having multiple distinct available meanings. In this work, utterances are restricted to scene description, so an utterance is ambiguous if there are at least two distinct scene reconstructions available.

**Corpus** A collection of language resources. Corpora referred to in this work are always collections of written text.

**Information Gap** A part of the meaning of an utterance that is not given through compositional structure but is to be inferred by the listener.

**Lexeme** The basic unit of meaning in the mental lexicon. LEARNA uses lexemes like *box*, *green virus*, or *blue cell*, which are not built from smaller parts but stored in the lexicon.

**Listener** The receiver of a natural language expression produced by a speaker. In this work, listener simply refers to the agent that tries to understand an utterance. It is further assumed that the listener is cooperative, i.e. tries to infer the meaning using all available resources.

**Prepositional Phrase** A phrase of the form *PREPOSITION NOUN-PHRASE*, e.g. *on the platform* or *to the red cell*.

**Scene Reconstruction** Introduced as a helper concept in this work: when a listener understood an utterance that described a scene, the internal simulated reconstruction of that scene is called the *scene reconstruction* in this work.

**Semantic Knowledge** The knowledge of the meaning of a word and how to combine it with other words, specifically including rules of applicability of the word to real world entities.

**Speaker** The sender or producer of a natural language expression directed at a listener. In this work, speaker means *producer* and simply refers to the source of an utterance. It is further assumed that the speaker wants to be understood correctly.

**Syntactic Knowledge** The tacit knowledge of syntactic rules and specifically their contribution to the compositional meaning of an utterance.

**Underspecification** Relevant information about the scene an utterance describes that was not included, see also information gap. Such information is not restricted and could, for example, be a certain detail of context. Ambiguity is a special case of underspecification where the options to fill the information gap are clearly distinct. The options to interpret the specific size of a car in the utterance *a car drove by* are taken from a continuum. This would thus be underspecification, but not ambiguity.

**Understand** In this work, we say that a listener understands an utterance if they are able to reproduce the relevant parts of the scene that the utterance describes. This reproduction is called the *scene reconstruction* in this work and takes place within cognition. In this work, only utterances that describe scenes are investigated. We assume that if the listener is able to reproduce the scene, they will also know the truth conditions of the utterance and that our definition is thus compatible with traditional truth-conditional meaning.

**Utterance** In this work restricted to the meaning: *natural language description of a scene*. In general: a natural language expression that a speaker produces and a listener receives. In this work, the distinction between *sentence*, i.e. a sentence in isolation, and *utterance*, i.e. a sentence that is uttered in a specific context, is not important.

## Glossary of Terms Specific to The Implementation

**Abstract Scene Representation** A set of requirements that a scene reconstruction has to fulfill to match an utterance. Abstract scene representations are generated by the LPS from an utterance and passed to the ISS for inference.

**Adaptation Heuristics** Heuristics applied by the ISS to adapt an incorrect scene reconstruction to make it fit the ASR provided by the LPS.

**Event Dynamics Model** A model that models the unfolding dynamics of an event. LEARNA implements these in a factorized way, using linear models for the unfolding dynamics where each model predicts the new value of a feature.

**Event-Predictive System** The part of the artificial system implemented in this work that learns and manages all predictive models.

**Inference Simulation System** The part of LEARNA where simulation and inference takes place. The inference simulation system receives the abstract requirements of an input utterance in the form of an ASR from the LPS. It then operates on the encodings of the EPS to infer a simulation that matches the ASR.

**Language Processing System** The part of LEARNA where language is processed. The language system takes utterances as input and provides the information that the utterance contains as abstract requirements in the form of an ASR.

**Transition Model** A model of the transition between two event dynamics models, which are not necessarily different. A transition model is always centered on one object and is thus specific to the corresponding object type. It builds a situation encoding to predict whether the transition should take place given the current situation.

## References

- Agresti, A. (2013). *Categorical data analysis* (3rd ed). Hoboken, NJ, Wiley.
- Baddeley, A. D. & Hitch, G. (1974). Working memory. In *Psychology of learning and motivation* (pp. 47–89). Elsevier.
- Bar-Hillel, Y. (1960). The present status of automatic translation of languages. In *Advances in computers* (pp. 91–163). Elsevier.
- Bever, T. G. (1970). The cognitive basis for linguistic structures. In R. Hayes (Ed.), *Cognition and language development* (pp. 279–362). Wiley; Sons Inc.
- Bohnemeyer, J. & Pederson, E. (Eds.). (2011). *Event representation in language and cognition*. Cambridge; New York, Cambridge University Press.
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). Language models are few-shot learners.
- Bunger, A., Papafragou, A. & Trueswell, J. C. (2013). Event structure influences language production: Evidence from structural priming in motion event description. *Journal of Memory and Language*, 69(3), 299–323.
- Butz, M. V. (2016). Toward a unified sub-symbolic computational theory of cognition. *Frontiers in Psychology*, 7.
- Butz, M. V. (2017). Which structures are out there. In T. K. Metzinger & W. Wiese (Eds.), *Philosophy and predictive processing*. Frankfurt am Main, MIND Group.
- Chomsky, N. (2002). *On nature and language* (A. Belletti & L. Rizzi, Eds.; 1st ed.). Cambridge University Press.
- Christensen, R. H. B. (2019). Ordinal—regression models for ordinal data.
- Clark, A. (2016). *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford, New York, Oxford University Press.
- Clark, H. H. & Clark, E. V. (1977). *Psychology and language: An introduction to psycholinguistics*. New York, Harcourt Brace Jovanovich.
- Cole, D. (2020). The chinese room argument. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Spring 2020). Metaphysics Research Lab, Stanford University.
- Davis, E. (2019). *Collection of winograd schemas*. Retrieved June 14, 2019, from <http://www.cs.nyu.edu/faculty/davise/papers/WinogradSchemas/WSCollection.html>
- Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford, Oxford University Press.
- Elman, J. L. & McRae, K. (2019). A model of event knowledge. *Psychological Review*, 126(2), 252–291.
- Ferreira, V. S. (2019). A mechanistic framework for explaining audience design in language production. *Annual Review of Psychology*, 70(1).

- Ferreira, V. S., Slevc, L. R. & Rogers, E. S. (2005). How do speakers avoid ambiguous linguistic expressions? *Cognition*, 96(3), 263–284.
- Fodor, J. A. (1979). *The language of thought*. Cambridge, Mass, Harvard Univ. Press.
- Fodor, J. A. (1987). *Psychosemantics: The problem of meaning in the philosophy of mind*. Cambridge, Mass, MIT Press.
- Franklin, N. T., Norman, K. A., Ranganath, C., Zacks, J. M. & Gershman, S. J. (2020). Structured event memory: A neuro-symbolic model of event cognition. *Psychological Review*, 127(3), 327–361.
- Frege, G. (1879). *Begriffsschrift, eine der arithmetischen nachgebildete formelsprache des reinen denkens*. Nebert.
- Freuder, E. C. & Mackworth, A. K. (2006). Constraint satisfaction: An emerging paradigm. In *Foundations of artificial intelligence* (pp. 13–27). Elsevier.
- Friston, K. (2010). The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, 11(2), 127–138.
- Fukumura, K. & Gompel, R. P. G. v. (2012). Producing pronouns and definite noun phrases: Do speakers use the addressee’s discourse model? *Cognitive Science*, 36(7), 1289–1311.
- Fukumura, K. & van Gompel, R. P. G. (2010). Choosing anaphoric expressions: Do people take into account likelihood of reference? *Journal of Memory and Language*, 62(1), 52–66.
- Fukumura, K., van Gompel, R. P. G., Harley, T. & Pickering, M. J. (2011). How does similarity-based interference affect the choice of referring expression? *Journal of Memory and Language*, 65(3), 331–344.
- Garvey, C. & Caramazza, A. (1974). Implicit causality in verbs. *Linguistic Inquiry*, 5(3), 459–464.
- Glenberg, A. M. & Gallese, V. (2012). Action-based language: A theory of language acquisition, comprehension, and production. *Cortex*, 48(7), 905–922.
- Grice, H. P. (1967). Logic and conversation. In P. Grice (Ed.), *Studies in the way of words* (pp. 41–58). Harvard University Press.
- Grosz, B. J., Weinstein, S. & Joshi, A. K. (1995). Centering: A framework for modeling the local coherence of discourse. *Comput. Linguist.*, 21(2), 203–225.
- Gumbsch, C., Butz, M. V. & Martius, G. (2019). Autonomous identification and goal-directed invocation of event-predictive behavioral primitives. *arXiv:1902.09948 [cs]*arxiv 1902.09948.
- Gumbsch, C., Otte, S. & Butz, M. V. (2017). A computational model for the dynamical learning of event taxonomies. In A. Papafragou, D. Grodner, D. Mirman & J. Trueswell (Eds.), *Proceedings of the 39th annual conference of the cognitive science society* (pp. 452–457). London, Cognitive Science Society.
- Hampe, B. (2005). Image schemas in cognitive linguistics: Introduction. In B. Hampe & J. E. Grady (Eds.), *From perception to meaning* (pp. 1–14). Berlin, New York, Mouton de Gruyter.

- Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1), 335–346.
- Hartshorne, J. K., O'Donnell, T. J. & Tenenbaum, J. B. (2015). The causes and consequences explicit in verbs. *Language, cognition and neuroscience*, 30(6), 716–734.
- Hayes, M. H. (1996). *Statistical digital signal processing and modeling*. New York, John Wiley & Sons.
- Haywood, S. L., Pickering, M. J. & Branigan, H. P. (2005). Do speakers avoid ambiguities during dialogue? *Psychological Science*, 16(5), 362–366.
- Hohwy, J. (2013). *The predictive mind*. Oxford, New York, Oxford University Press.
- Jackendoff, R. (2002). *Foundations of language*. Oxford University Press.
- Jaeger, F. T. (2010). Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology*, 61(1), 23–62.
- Johnson-Laird, P. (2008). *How we reason*. Oxford University Press.
- Kaup, B., Lüdtke, J. & Maienborn, C. (2010). 'The drawer is still closed': Simulating past and future actions when processing sentences that describe a state. *Brain and Language*, 112(3), 159–166.
- Kingsbury, D. (1968). *Manipulating the amount of information obtained from a person giving directions* (Doctoral dissertation). Harvard University.
- Kocijan, V., Cretu, A.-M., Camburu, O.-M., Yordanov, Y. & Lukaszewicz, T. (2019). A surprisingly robust trick for the winograd schema challenge, In *Proceedings of the 57th annual meeting of the association for computational linguistics, acl 2019, florence, italy, july 28 - august 2, 2019*. 57th Annual Meeting of the Association for Computational Linguistics, ACL, Association for Computational Linguistics.
- Korte, B. & Vygen, J. (2018). *Kombinatorische optimierung: Theorie und algorithmen*. Berlin, Heidelberg, Springer Berlin Heidelberg.
- Kuperberg, G. R. & Jaeger, T. F. (2016). What do we mean by prediction in language comprehension? *Language, Cognition and Neuroscience*, 31(1), 32–59.
- Kurby, C. A. & Zacks, J. M. (2008). Segmentation in the perception and memory of events. *Trends in cognitive sciences*, 12(2), 72–79.
- Kurumada, C. & Jaeger, T. F. (2015). Communicative efficiency in language production: Optional case-marking in japanese. *Journal of Memory and Language*, 83, 152–178.
- Lakoff, G. (1987). *Women, fire, and dangerous things: What categories reveal about the mind*. Chicago, University of Chicago Press.
- Lane, L. W., Groisman, M. & Ferreira, V. S. (2006). Don't talk about pink elephants!: Speakers' control over leaking private information during language production. *Psychological Science*, 17(4), 273–277.
- Lee, H. (2015). Information structure, topic predictability and gradients in korean case ellipsis: A probabilistic account. *Linguistic Research*, 32(3), 749–771.
- Lettvin, J., Maturana, H., McCulloch, W. & Pitts, W. (1959). What the frog's eye tells the frog's brain. *Proceedings of the IRE*, 47(11), 1940–1951.



- Levesque, H. J., Davis, E. & Morgenstern, L. (2012). The winograd schema challenge, In *Proceedings of the thirteenth international conference on principles of knowledge representation and reasoning*, Rome, Italy, AAAI Press.
- Levinson, S. C. (2000). *Presumptive meanings: The theory of generalized conversational implicature*. The MIT Press.
- Lin, S.-C., Yang, J.-H., Nogueira, R., Tsai, M.-F., Wang, C.-J. & Lin, J. (2020). TTTTackling WinoGrande schemas. *arXiv:2003.08380 [cs]arxiv 2003.08380*.
- Liu, Q., Jiang, H., Ling, Z.-H., Zhu, X., Wei, S. & Hu, Y. (2017). Combing context and commonsense knowledge through neural networks for solving winograd schema problems, In *2017 aai spring symposium series*.
- Löbner, S. (2012). *Semantik: Eine einföhrung*. Berlin [u.a., de Gruyter.
- Marr, D. (2010). *Vision: A computational investigation into the human representation and processing of visual information*. The MIT Press.
- Peng, H., Khashabi, D. & Roth, D. (2015). Solving hard coreference problems, In *Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: Human language technologies*, Denver, Colorado, Association for Computational Linguistics.
- Piantadosi, S. T., Tily, H. & Gibson, E. (2012). The communicative function of ambiguity in language. *Cognition*, 122(3), 280–291.
- Rao, R. P. N. & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1), 79–87.
- Rescorla, M. (2019). The language of thought hypothesis. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Summer 2019). Metaphysics Research Lab, Stanford University.
- Rescorla, M. (2020). The computational theory of mind. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Spring 2020). Metaphysics Research Lab, Stanford University.
- Rohde, H. & Kehler, A. (2014). Grammatical and information-structural influences on pronoun production. *Language, Cognition and Neuroscience*, 29(8), 912–927.
- Rosa, E. C. & Arnold, J. E. (2017). Predictability affects production: Thematic roles can affect reference form selection. *Journal of Memory and Language*, 94, 43–60.
- Russell, S. J., Norvig, P. & Davis, E. (2010). *Artificial intelligence: A modern approach* (3rd ed). Upper Saddle River, Prentice Hall.
- Sakaguchi, K., Bras, R. L., Bhagavatula, C. & Choi, Y. (2019). WinoGrande: An adversarial winograd schema challenge at scale.
- Schank, R. C. & Abelson, R. P. (1977). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Hillsdale, N.J., L. Erlbaum Associates.
- Schnoebelen, T. & Kuperman, V. (2010). Using amazon mechanical turk for linguistic research. *Psihologija*, 43(4), 441–464.

- Scholz, B. C., Pelletier, F. J. & Pullum, G. K. (2020). Philosophy of linguistics. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Summer 2020). Metaphysics Research Lab, Stanford University.
- Schrod, F., Kneissler, J., Ehrenfeld, S. & Butz, M. V. (2017). Mario becomes cognitive. *Topics in Cognitive Science*, 9(2), 343–373.
- Schrod, F., Röhm, Y. & Butz, M. V. (2017). An event-schematic, cooperative, cognitive architecture plays super mario. In R. Chrisley, V. C. Müller, Y. Sandamirskaya & M. Vincze (Eds.), *Proceedings of the EUCognition meeting (european society for cognitive systems) "cognitive robot architectures"* (pp. 10–15).
- Schüller, P. (2014). Tackling winograd schemas by formalizing relevance theory in knowledge graphs, In *Fourteenth international conference on the principles of knowledge representation and reasoning*.
- Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences*, 3(3), 417–424.
- Sennet, A. (2016). Ambiguity. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Spring 2016). Metaphysics Research Lab, Stanford University.
- Sharma, A., Vo, N. H., Gaur, S. & Baral, C. (2015). An approach to solve winograd schema challenge using automatically extracted commonsense knowledge, In *2015 AAAI spring symposium series*.
- Steels, L. (2008). The symbol grounding problem has been solved. so what's next? In M. d. Vega, A. M. Glenberg & A. C. Graesser (Eds.), *Symbols and embodiment: Debates on meaning and cognition*. Oxford ; New York, Oxford University Press.
- Stegemann-Philipps, C. & Butz, M. V. (2021). Learn it first: Grounding language in compositional event-predictive encodings. In *2021 IEEE International Conference on Development and Learning (ICDL)* (pp. 1–6).
- Stegemann-Philipps, C., Butz, M. V., Winkler, S. & Achimova, A. (2021). Speakers use more informative referring expressions to describe surprising events. In *Proceedings of the 43rd annual meeting of the cognitive science society*.
- Storks, S., Gao, Q. & Chai, J. Y. (2019). Commonsense reasoning for natural language understanding: A survey of benchmarks, resources, and approaches. *CoRR*, abs/1904.01172arXiv 1904.01172. <http://arxiv.org/abs/1904.01172>
- Vonk, J. M., Higby, E. & Obler, L. K. (2018). Comprehension in older adult populations: Healthy aging, aphasia, and dementia. In E. M. Fernández & H. S. Cairns (Eds.), *The handbook of psycholinguistics: Edited by eva m. fernández and helen smith cairns* (pp. 240–268). Hoboken, NJ Oxford, Wiley Blackwell.
- Wasow, T. (2015). Ambiguity avoidance is overrated. In S. Winkler (Ed.), *Ambiguity: Language and communication* (p. 29).
- Wasow, T., Perfors, A. & Beaver, D. (2005). The puzzle of ambiguity. *Morphology and the web of grammar: Essays in memory of Steven G. Lapointe*, 265–282.

- Wharton, C. & Kintsch, W. (1991). An overview of construction-integration model: A theory of comprehension as a foundation for a new cognitive architecture. *ACM SIGART Bulletin*, 2(4), 169–173.
- Wiese, W. & Metzinger, T. K. (2017). Vanilla PP for philosophers: A primer on predictive processing. In T. K. Metzinger & W. Wiese (Eds.), *Philosophy and predictive processing*. Frankfurt am Main, MIND Group.
- Wilson, M. (2002). Six views of embodied cognition. *Psychonomic Bulletin & Review*, 9(4), 625–636.
- Winkler, S. (2015). Exploring ambiguity and the ambiguity model from a transdisciplinary perspective. In S. Winkler (Ed.), *Ambiguity: Language and communication* (pp. 1–26).
- Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1), 1–191.
- Winter-Froemel, E. & Zirker, A. (2015). Ambiguity in speaker-hearer-interaction: A parameter-based model of analysis. In S. Winkler (Ed.), *Ambiguity*. Berlin, München, Boston, DE GRUYTER.
- Zacks, J. M., Speer, N. K., Swallow, K. M., Braver, T. S. & Reynolds, J. R. (2007). Event perception: A mind/brain perspective. *Psychological bulletin*, 133(2), 273–293.
- Zacks, J. M. & Tversky, B. (2001). Event structure in perception and conception. *Psychological Bulletin*, 127(1), 3–21.
- Zipf, G. (1950). Human behavior and the principle of least effort. *Social Forces*, 28(3), 340–341.
- Zwaan, R. A. (2003). The immersed experiencer: Toward an embodied theory of language comprehension. In *Psychology of learning and motivation* (pp. 35–62). Elsevier.
- Zwaan, R. A. & Radvansky, G. A. (1998). Situation models in language comprehension and memory. *Psychological Bulletin*, 123(2), 162–185.

## Appendix A Reference to Implemented BrainControl Code

The code used in this work can be found in a private github repository within the cognitive modeling group's github area. It can be reached via [github.com/ CognitiveModeling/ BrainControl-Research](https://github.com/CognitiveModeling/BrainControl-Research) (without spaces). The repository contains several branches; the code used in this work is stored in the branch *trymerge*. While there were adjustments made to original BrainControl files, LEARNA consists of two packages, namely `braincontrolAI.eventSequencing` for the EPS and the ISS and `braincontrolAI.language` for the LPS.

To run the program, follow the description in the readme file of the public BrainControl repository: [github.com/ CognitiveModeling/ BrainControl](https://github.com/CognitiveModeling/BrainControl) (without spaces). In case of technical difficulties, contact [christian.stegemann@uni-tuebingen.de](mailto:christian.stegemann@uni-tuebingen.de)

## Appendix B Vocabulary in final evaluation

Table 22: Mappings for nouns known to the system at the time of final evaluation

Nouns	Mapped Set
light bulb	BULB_FLOWER
arrow	BULLET_WILLY
box	IRON_BLOCK1, MOVABLE_BOX
cell	CELL_LONG_BLUE, CELL_SPIKES_RED
virus	GREEN_VIRUS, RED_VIRUS
red box	MOVABLE_BOX
platform	TOP_CENTER_WRENCH_HEAD, TOP_RIGHT_WRENCH_HEAD, TOP_LEFT_WRENCH_HEAD
green arrow	BULLET_WILLY
robot	CLARK, TALKING_ROBOT, PETER
ball	VIRUS_BALL
red square	MOVABLE_BOX
peter	PETER
block	SIMPLE_BLOCK1
virus ball	VIRUS_BALL
blue cell	CELL_LONG_BLUE
green block	IRON_BLOCK1
red cell	CELL_SPIKES_RED
red virus	RED_VIRUS
green virus	GREEN_VIRUS
bulb	BULB_FLOWER

blue block	SIMPLE_BLOCK1
it	NONE
grey robot	CLARK, TALKING_ROBOT
bullet willy	BULLET_WILLY
wrench	WRENCH
she	NONE
clark	CLARK, TALKING_ROBOT
blue robot	PETER
green box	IRON_BLOCK1
ground	TOP_DIRT_BORDER_GROUND
blue square	SIMPLE_BLOCK1
wall	RIGHT_DIRT_BORDER, LEFT_DIRT_BORDER
he	NONE

Table 23: Mappings for verbs known to the system at the time of final evaluation

Verbs	Mapped Set
flies	112, 134
flees	475, 476, 477, 699, 481, 482
falls	275, 245, 224, 225, 229, 283
breaks	1811, 1799, 2068, 2056, 1807, 1804, 2060
attacks	474, 377, 366, 367, 380, 372, 384, 450
moves	88, 89, 122, 124, 90, 91, 119, 120
rests	99, 44, 14, 26, 38, 193, 262, 110, 199, 2, 268, 8, 81, 50, 20, 86, 32, 76
pushes	365, 370, 294, 2859, 382, 295, 296
consumes	551, 531, 546, 529

Table 24: Mappings for prepositions known to the system at the time of final evaluation

Prepositions	Mapped Set
from	24, 14, 16, 8
(direct object)	23, 2, 4, 18, 7
to	11, 12, 15, 19

Table 25: Mappings for conjunctions known to the system at the time of final evaluation

Conjunctions	Mapped Set
after, when, if	after
while	during
and, before	before

## Appendix C Details of Production Experiment

To save space, all data relevant to the experiment was moved to a private github repository within the cognitive modeling group’s github area. It can be reached via [github.com/ CognitiveModeling/ 2020\\_CSt\\_ProductionExperiment](https://github.com/CognitiveModeling/2020_CSt_ProductionExperiment) (without spaces). To request access, please send an email to [christian.stegemann@uni-tuebingen.de](mailto:christian.stegemann@uni-tuebingen.de).

The repository contains the experiment as it was seen by participants as well as the raw data (except for audio files) and the script used for analysis.

## Appendix D Models Learned By Final Implemented System

Table 26: Known predictive models at the time of final evaluation

ID	Predictive Model
1	$P(x)=0$
2	$P(vx)=0$
3	$P(y)=0$
4	$P(vy)=0$
5	$P(h)=h*1$
6	$P(e)=0$
7	$P(vx)=vx*-0.015+Mxdir*-0.151+Mvx*-1.754+Mvy*-1.928+Mact*0.202+h*-0.087+e*0.085$
8	$P(e)=0.025+Mvx*-0.03+Mvy*0.006+Mact*-0.3+e*1$
9	$P(vx)=vx*-0.233+Mxdir*-0.084+Mvx*1.304+Mact*0.577+h*0.145+e*0.085$
10	$P(vx)=vx*0.366+vy*-0.022+Mxdir*1.588+Mvx*-0.088+Mvy*-0.689+Mact*-5.897+h*-0.026+e*0.091$
11	$P(vx)=Mvx*-2.25$

12  $P(vx)=vx*-0.009+Mxdir*-0.054+Mvx*2.583+Mvy*2.231+Mact*-0.235+h*0.004+e*-0.005$   
13  $P(vx)=vx*0.001+vy*0+Mxdir*-0.005+Mvx*1.361+Mact*0.004$   
14  $P(vx)=0.028+vx*0+vy*-0+Mxdir*-0.001+Mvx*-1.363+Mact*-0.011+h*-0+e*0$   
15  $P(e)=Mvy*0+Mact*0.001+e*1$   
16  $P(e)=0.014+vx*-0.001+Mxdir*0.008+Mvx*-0.042+Mvy*0.024+Mact*0.211+h*0+e*1$   
17  $P(vx)=Mvx*-4.5$   
18  $P(vx)=vx*-0.018+Mxdir*0.198+Mvx*2.791+Mvy*2.269+Mact*0.176+h*0.01+e*-0.006$   
19  $P(vx)=vx*0.019+Mxdir*0.177+Mvx*3.103+Mvy*2.176+Mact*-0.787+h*-0.009+e*0.002$   
20  $P(vx)=vx*0.04+Mxdir*0.202+Mvx*3.907+Mvy*1.929+Mact*-2.446+h*0.021+e*0.021$   
21  $P(h)=Mact*0.001$   
22  $P(vx)=vx*0.016+Mxdir*2.592+Mvx*2.075+Mact*-0.182+h*-0.005+e*-0.03$   
23  $P(vx)=0.031+vx*0.002+Mxdir*0.005+Mvx*-1.36+Mact*-0.006+h*-0+e*0$   
24  $P(vx)=Mvx*4.5$   
25  $P(vx)=Mvx*2.25$   
26  $P(vx)=0.022+vx*0.001+Mxdir*-0.006+Mvx*-1.362+Mact*-0+h*-0.001+e*0$   
27  $P(vx)=vx*0.003+Mxdir*0.003+Mvx*1.359+Mact*-0.011+h*-0+e*0$   
28  $P(vx)=0.022+vx*0.003+Mxdir*-0.002+Mvx*-1.36+Mact*-0.004+h*0+e*-0$   
29  $P(vx)=vx*0.004+Mxdir*-0.001+Mvx*1.357+Mact*-0.003$   
30  $P(vx)=0.046+vx*0.001+vy*-0.202+Mxdir*-0.044+Mvx*-1.351+Mact*0.068+h*-0.003+e*-0.009$   
31  $P(h)=vx*0.004+Mxdir*-0.154+Mvx*0.067+Mact*0.449+h*1+e*0.015$   
32  $P(e)=0.684+vx*-0.005+Mxdir*0.146+Mvx*-0.035+Mact*-0.179+h*-0.167+e*1.081$   
33  $P(vy)=0.276+vx*-0.001+Mxdir*0.001+Mvx*0+Mact*0.002+h*0+e*0$   
34  $P(vx)=vx*0.514+vy*0.11+Mxdir*-0.038+Mvx*-0.09+Mact*0.55+h*-0.028+e*-0.054$   
35  $P(vy)=0.28+vy*1$   
36  $P(vy)=0.015+vy*1+Mxdir*0+Mvx*0+Mact*-0+h*0.043+e*-0$   
37  $P(vx)=0.005+vx*0.002+vy*-0.001+Mxdir*-0.007+Mvx*-1.36+Mact*-0.011+h*0.001+e*0$   
38  $P(vx)=vx*0.95$   
39  $P(vx)=0.757+vx*0.291+vy*-0.222+Mxdir*-0.028+Mvx*0.941+Mact*-0.075+h*-0.05+e*-0.043$   
40  $P(vy)=0.271+vx*0.001+vy*1+Mvx*-0+e*0.001$   
41  $P(vy)=0.262+vx*-0.007+vy*1.002+Mxdir*-0.003+Mvx*0.001+Mact*0.001+h*0+e*-0.001$   
42  $P(vx)=vx*0.003+Mxdir*0.006+Mvx*1.344+Mact*0.002+h*0.011+e*0.024$   
43  $P(h)=vx*-0.016+vy*-0.057+Mxdir*-0.009+Mvx*0.146+Mact*0.026+h*1.027+e*-0.001$   
44  $P(vx)=Mxdir*0.096+Mvx*-0.008+Mact*-0.024+e*-0.017$   
45  $P(vx)=0.447+vx*-0.003+vy*-0.172+h*0.07$   
46  $P(h)=vx*-0.037+vy*0.017+Mxdir*-0.027+Mvx*0.022+Mact*-0.006+h*1.014+e*-0.004$   
47  $P(vx)=0$   
48  $P(vx)=vx*0.005+vy*-0.168+Mxdir*0.005+Mvx*0.012+Mact*0.014+h*0.395+e*0.02$   
49  $P(h)=vy*0.001+h*1$   
50  $P(vx)=2.929+vx*0.008+vy*0.142+h*-0.34$   
51  $P(vx)=1.181+vx*0.001+Mxdir*0.198+Mvx*0.061+Mact*0.068+h*0.002+e*0.003$   
52  $P(vx)=0.2+vx*0.004+Mxdir*0.131+Mvx*1.203+Mact*-0.722+h*0+e*-0.003$   
53  $P(e)=0.026+vx*-0.003+Mxdir*0.016+Mvx*-0.067+Mact*0.163+h*0.011+e*0.992$   
54  $P(h)=vx*0.262+vy*0.036+Mact*0.073+h*0.987+e*-0.007$   
55  $P(vx)=1.5$   
56  $P(h)=0.155+vx*0.012+vy*0.008+Mxdir*-0.25+Mvx*0.187+Mact*0.36+h*0.855+e*0.029$   
57  $P(e)=0.028+vx*0.002+Mxdir*-0.002+Mvx*-0.051+Mact*0.198+h*0+e*0.999$   
58  $P(h)=vx*0.126+vy*0.02+Mxdir*0.065+Mvx*0.015+Mact*0.074+h*1.054+e*0.004$   
59  $P(vx)=Mvx*-1.35$   
60  $P(vx)=Mvx*1.35$   
61  $P(vx)=1.2$   
62  $P(h)=vx*-0.004+Mxdir*0.037+Mvx*0.328+Mact*-0.168+h*1.011+e*0.026$

63  $P(h)=vx*0.002+Mxdir*0.048+Mvx*0.343+Mact*0.057+h*1.022+e*0.003$   
64  $P(h)=vx*0.001+Mxdir*0.026+Mvx*-0.028+Mact*-0.051+h*1.017+e*-0.003$   
65  $P(h)=Mxdir*0.02+Mvx*0.402+Mact*-0.08+h*1.007+e*0.015$   
66  $P(h)=vx*-0.003+Mxdir*0.022+Mvx*0.027+Mact*0.083+h*1.011+e*0.002$   
67  $P(h)=vx*-0.004+Mxdir*0.021+Mvx*-0.005+Mact*0.056+h*1.009+e*0.004$   
68  $P(vx)=Mvx*1.8$   
69  $P(vx)=Mvx*-1.8$   
70  $P(h)=vx*-0.015+Mxdir*0.024+Mvx*0.008+Mact*0.032+h*1.008+e*-0$   
71  $P(h)=vx*-0.009+Mxdir*-0.003+Mvx*0.051+Mact*0.321+h*0.99+e*0.033$   
72  $P(h)=vx*-0.001+vy*0.002+Mxdir*0.008+Mvx*0.038+Mact*0.02+h*1.033+e*-0.002$   
73  $P(e)=0.025+Mxdir*0.002+Mvx*-0.052+Mact*0.2+e*1$   
74  $P(h)=1.531+vx*-0.005+Mxdir*-0.001+Mvx*0.003+Mact*0.011+h*-0.048+e*-0.001$   
75  $P(h)=vx*-0.015+Mxdir*0.166+Mvx*0.003+Mact*-0.052+h*0.948+e*0.077$   
76  $P(h)=vx*-0.001+Mxdir*-0.166+Mvx*0.024+Mact*0.447+h*1.001+e*0.011$   
77  $P(h)=vx*-0+Mxdir*0.003+Mvx*-0.004+Mact*0.207+h*1$   
78  $P(h)=vy*-0+Mxdir*0.004+Mvx*0.149+Mact*0.416+h*0.997+e*0.019$   
79  $P(h)=0.207+vx*-0.001+vy*0+Mxdir*0.001+Mvx*-0.001+Mact*0.202+h*0.985+e*0$   
80  $P(h)=vx*0.01+Mxdir*-0.009+Mvx*0.269+Mact*-0.452+h*0.992+e*0.012$   
81  $P(h)=Mxdir*0.001+Mvx*0.11+Mact*0.68+h*1.009+e*0.003$   
82  $P(h)=vx*-0.002+Mvx*0.583+Mact*-0.466+h*1.008+e*0.004$   
83  $P(h)=vx*0.001+Mxdir*0.006+Mvx*0.928+Mact*0.01+h*1.009+e*0.004$   
84  $P(h)=vx*-0.007+Mxdir*0.014+Mvx*0.028+Mact*-0.107+h*1.01+e*-0.001$   
85  $P(h)=vx*0.005+Mxdir*-0.008+Mvx*-0.017+Mact*0.047+h*1.024+e*-0.012$   
86  $P(h)=vx*0.04+Mxdir*-0+Mvx*0.094+Mact*-0.102+h*1.019+e*0.004$   
87  $P(h)=vx*0.001+Mxdir*-0.001+Mvx*0.026+Mact*-0.018+h*0.972+e*0$   
88  $P(h)=vx*-0.045+vy*-0.002+Mxdir*0.076+Mvx*0.211+Mact*0.884+h*0.972+e*0.118$   
89  $P(h)=0.032+vx*0.011+Mxdir*0.01+Mvx*0.227+Mact*0.155+h*1.01+e*-0.028$   
90  $P(h)=vx*0.001+Mxdir*0.011+Mvx*0.295+Mact*0.056+h*1.014+e*-0.015$   
91  $P(h)=Mxdir*-0.002+Mvx*0.308+Mact*-0.064+h*1.015+e*-0.014$   
92  $P(h)=0.14+Mxdir*-0+Mvx*-0.617+Mact*0.01+h*0.995+e*-0.017$   
93  $P(h)=0.469+vx*0+Mxdir*-0.109+Mvx*0.103+Mact*-0.078+h*1.003+e*-0.066$   
94  $P(h)=vx*0+Mxdir*0.014+Mvx*0.199+Mact*0.011+h*1.012+e*-0.004$   
95  $P(h)=vx*-0+Mxdir*-0.002+Mvx*0.036+Mact*0.249+h*1.008+e*-0.001$   
96  $P(h)=vx*0.019+Mxdir*-0.244+Mvx*0.084+Mact*0.477+h*1+e*0.013$   
97  $P(h)=0.339+vx*-0.088+Mxdir*0.064+Mvx*0.201+Mact*0.078+h*0.996+e*-0.037$   
98  $P(h)=vx*0.024+Mxdir*0.025+Mvx*0.09+Mact*-0.267+h*1.01+e*-0$   
99  $P(h)=vx*0.016+Mxdir*-0.014+Mvx*0.006+Mact*-0.018+h*1.007+e*0.017$   
100  $P(h)=vx*0.024+Mxdir*0.015+Mvx*0.074+Mact*-0.409+h*1.006+e*0.002$   
101  $P(h)=vx*-0.005+Mxdir*-0.301+Mvx*0.314+Mact*0.01+h*1.016+e*-0.02$   
102  $P(h)=vx*-0.007+Mxdir*0.038+Mvx*0.045+Mact*0.331+h*1.012+e*-0.006$   
103  $P(h)=vx*0.003+Mvx*0.283+Mact*1.248+h*1.021+e*-0.016$   
104  $P(h)=Mxdir*-0.009+Mvx*0.159+Mact*0.082+h*1.019+e*-0.004$   
105  $P(h)=Mxdir*-0.003+Mvx*0.283+Mact*0.009+h*1.021+e*-0.001$   
106  $P(h)=Mxdir*-0+Mvx*0.256+Mact*0.032+h*1.017+e*-0.001$   
107  $P(h)=vx*-0+Mxdir*-0.005+Mvx*-0.025+Mact*-0.032+h*1.012+e*-0.001$   
108  $P(h)=vx*0.001+Mxdir*-0.001+Mvx*0.032+Mact*-0.009+h*1.008+e*-0.007$   
109  $P(h)=vx*0.004+Mxdir*-0.021+Mvx*0.02+Mact*0.034+h*1.002$   
110  $P(h)=vx*-0.018+Mxdir*0.003+Mvx*0.052+Mact*0.283+h*0.998+e*0.013$   
111  $P(h)=0+vx*0.025+Mxdir*-0.036+Mvx*0.079+Mact*0.246+h*1.001+e*-0.008$   
112  $P(h)=vx*-0+Mxdir*-0.062+Mvx*-0.004+Mact*0.219+h*1.002+e*0.016$   
113  $P(h)=7.668+vx*-0.015+Mxdir*-0.071+Mvx*0.026+Mact*0.535+h*0.477+e*0.075$



114  $P(h)=0.15+vx*0.032+Mxdir*-0.058+Mvx*0.289+Mact*-1.089+h*0.998+e*-0.031$   
115  $P(h)=vx*0.041+Mxdir*-0.089+Mvx*0.458+Mact*-0.287+h*1.005+e*0.033$   
116  $P(h)=0.005+vx*0.01+Mxdir*0.012+Mvx*0.035+Mact*0.241+h*1.004+e*-0.013$   
117  $P(h)=vx*-0.023+Mxdir*-0.02+Mvx*0.031+Mact*-0.248+h*1.006+e*-0.007$   
118  $P(h)=vx*0.004+Mxdir*-0.033+Mvx*0.242+Mact*-0.159+h*1.015+e*0.024$   
119  $P(h)=Mxdir*-0.012+Mvx*0.075+Mact*0.039+h*1.007+e*0.013$   
120  $P(h)=vx*-0.006+Mxdir*-0.015+Mvx*0.005+Mact*-0.034+h*1.006+e*0.013$   
121  $P(h)=vx*-0+Mxdir*-0.046+Mvx*-0.005+Mact*0.012+h*1.002+e*0.002$   
122  $P(h)=vx*-0.002+vy*-0.001+Mact*0.454+h*0.995+e*0.017$   
123  $P(h)=vx*-0.001+Mxdir*0.005+Mvx*0.058+Mact*0.127+h*1.003+e*0.005$   
124  $P(h)=vx*0.062+Mxdir*0.018+Mvx*0.213+Mact*0.285+h*1.004+e*-0.008$   
125  $P(h)=vx*-0.035+Mxdir*-0.005+Mvx*0.43+Mact*1.231+h*1.088+e*0.097$   
126  $P(h)=0.131+vx*-0.01+Mxdir*0.015+Mvx*0.272+Mact*0.645+h*0.989+e*-0.013$   
127  $P(h)=Mxdir*0+Mvx*-0.439+Mact*0.043+h*1.005+e*-0.002$   
128  $P(h)=vx*0.001+Mxdir*-0.037+Mvx*-0.272+Mact*-0.055+h*1.004+e*0.001$   
129  $P(h)=vx*-0+Mxdir*0.003+Mvx*0.52+Mact*0.035+h*1.002+e*-0.001$   
130  $P(h)=vx*0.005+Mxdir*-0.029+Mvx*0.025+Mact*0.091+h*1.007+e*0.003$   
131  $P(h)=9.727+vx*0.004+Mxdir*-0.009+Mvx*0.02+Mact*-0.095+h*0.277+e*0.006$   
132  $P(h)=vx*-0.006+Mxdir*0.005+Mvx*0.084+Mact*0.146+h*1.009+e*0.001$   
133  $P(h)=0.001+vx*-0.046+Mxdir*-0.011+Mvx*0.153+Mact*-0.145+h*1.004+e*-0.014$   
134  $P(h)=vx*0.004+Mxdir*-0.001+Mvx*0.081+Mact*-0.06+h*1.004+e*-0.001$   
135  $P(h)=vx*0.01+Mxdir*0.003+Mvx*0.207+Mact*-0.207+h*1.006+e*0.009$   
136  $P(h)=vx*0+Mxdir*0.032+Mvx*0.688+Mact*0.182+h*1.003+e*-0.004$   
137  $P(h)=vx*0.001+Mxdir*0.013+Mvx*0.569+Mact*-0.244+h*1.015+e*0.004$   
138  $P(h)=vx*-0+Mxdir*0.018+Mvx*-0.674+Mact*0.062+h*1.014+e*-0.001$   
139  $P(h)=vx*-0.005+Mxdir*0.016+Mvx*0.032+Mact*-0.051+h*1.012+e*-0.001$   
140  $P(h)=vx*0+Mxdir*0.055+Mvx*-0.005+Mact*-0.01+h*1+e*0$   
141  $P(h)=0+vx*-0.019+vy*0+Mxdir*-0.026+Mvx*0.127+Mact*0.812+h*0.867+e*0.304$   
142  $P(h)=2.42+vx*0.017+Mxdir*0.021+Mvx*0.081+Mact*0.368+h*0.682+e*0.165$   
143  $P(h)=0.107+Mxdir*0+Mvx*-0.001+Mact*0.152+h*0.969+e*0.002$   
144  $P(h)=vx*0+Mxdir*-0+Mvx*-0.009+Mact*-0.034+h*0.997+e*-0.004$   
145  $P(vx)=vx*-0.001+h*-0.073+e*-0.019$   
146  $P(h)=0.156+vx*0.012+Mxdir*-0.035+Mvx*0.461+Mact*0.062+h*0.985+e*0.001$   
147  $P(h)=Mxdir*0.002+Mvx*-0.347+Mact*0.063+h*1.013+e*0.009$   
148  $P(h)=vx*0.006+Mxdir*-0.01+Mvx*0.013+Mact*0.011+h*1.011+e*0.012$   
149  $P(h)=vx*0.011+Mxdir*-0.002+Mvx*0.017+Mact*0.02+h*1.01+e*0.002$   
150  $P(h)=vx*-0.033+vy*0.002+Mxdir*0.115+Mvx*0.125+Mact*0.104+h*1.019+e*0.034$   
151  $P(h)=6.784+vx*0.001+Mxdir*-0.045+Mvx*0.031+Mact*-0.14+h*0.355+e*0.2$   
152  $P(h)=vx*0.006+Mxdir*0.185+Mvx*0.505+Mact*0.453+h*0.977+e*0.024$   
153  $P(h)=vx*-0.034+Mxdir*0.054+Mvx*0.555+Mact*0.398+h*0.997+e*0.002$   
154  $P(h)=0$   
155  $P(h)=0.1+vx*0.033+Mxdir*-0.049+Mvx*0.313+Mact*-0.587+h*0.998+e*-0.015$   
156  $P(h)=vx*0+Mxdir*-0.055+Mvx*-0.257+Mact*0.062+h*1.016+e*-0.01$   
157  $P(h)=vx*0+Mxdir*-0.013+Mvx*0.046+Mact*0.135+h*1.013+e*-0.005$   
158  $P(h)=vx*-0.009+Mxdir*0.019+Mvx*0.02+Mact*0.007+h*1.007+e*0.011$   
159  $P(h)=vx*-0.029+Mxdir*0.215+Mvx*0.375+Mact*0.204+h*1.001+e*0.001$   
160  $P(h)=vx*-0.09+Mxdir*-0.114+Mvx*0.265+Mact*0.263+h*0.996+e*0.019$   
161  $P(h)=8.526+vx*-0.001+Mxdir*-0.028+Mvx*0.019+Mact*-0.468+h*0.296+e*-0.13$   
162  $P(h)=0.085+vx*0+Mxdir*-0.001+Mvx*-0+Mact*0.006+h*0.992$   
163  $P(h)=7.215+vx*-0+Mxdir*-0.515+Mvx*0.034+Mact*-0.204+h*0.3+e*0.101$   
164  $P(h)=vx*-0.011+Mxdir*-0.047+Mvx*0+Mact*0.009+h*0.992+e*0.012$

165	$P(h)=vx^*-0.003+Mxdir^*-0.001+Mvx^*0.341+Mact^*-0.185+h^*1.009+e^*0.014$
166	$P(h)=vx^*-0.001+Mxdir^*-0.039+Mvx^*-0.003+Mact^*-0.012+h^*1.011+e^*0.016$
167	$P(h)=vx^*-0.001+Mxdir^*0.005+Mvx^*0.032+Mact^*-0+h^*1$
168	$P(h)=8.791+vx^*-0.004+Mxdir^*0.006+Mvx^*0.015+Mact^*-0.223+h^*0.2+e^*0.002$
169	$P(h)=vx^*0.035+vy^*-0.002+Mxdir^*0.005+Mvx^*0.344+Mact^*0.479+h^*1.006+e^*0.012$
170	$P(h)=9.979+vx^*-0.01+Mxdir^*-0.065+Mvx^*0.008+Mact^*-0.081+h^*0.351+e^*0$
171	$P(h)=vx^*0.011+Mxdir^*-0.022+Mvx^*0.291+Mact^*-0.114+h^*1.029+e^*0.009$
172	$P(h)=Mxdir^*-0.001+Mvx^*-0.265+Mact^*-0.162+h^*1.022+e^*0.005$
173	$P(h)=Mxdir^*0.018+Mvx^*0.052+Mact^*0.005+h^*1.019+e^*0.002$
174	$P(h)=vx^*-0+Mxdir^*-0.027+Mvx^*0.015+Mact^*0.001+h^*1$
175	$P(h)=vx^*-0.021+Mxdir^*-0.158+Mvx^*0.474+Mact^*0.263+h^*0.999+e^*0.051$
176	$P(h)=8.18+vx^*0.013+Mxdir^*-0.024+Mvx^*0.057+Mact^*0.217+h^*0.357+e^*0.218$
177	$P(h)=vx^*-0+Mxdir^*0.038+Mvx^*0.014+Mact^*0.201+h^*1.007+e^*-0.001$
178	$P(h)=5.133+vx^*0.026+Mxdir^*-0.087+Mvx^*0.041+Mact^*-0.227+h^*0.281+e^*0.839$
179	$P(h)=vx^*0.007+Mxdir^*0.01+Mvx^*0.282+Mact^*0.444+h^*1.012+e^*0$
180	$P(h)=Mxdir^*0+Mvx^*0.067+Mact^*-0.006+h^*1.017+e^*-0.015$
181	$P(h)=Mxdir^*0.01+Mvx^*1.028+Mact^*-0.064+h^*1.013+e^*-0.002$
182	$P(h)=0.105+vx^*-0.006+Mxdir^*0.025+Mvx^*0.339+Mact^*0.074+h^*1.008+e^*-0.033$
183	$P(h)=vx^*0.033+Mxdir^*-0.232+Mvx^*-0.048+Mact^*-0.183+h^*1.023+e^*0.055$
184	$P(h)=0.761+vx^*0.024+Mxdir^*0.107+Mvx^*0.031+Mact^*0.317+h^*0.855+e^*0.113$
185	$P(h)=vx^*0.002+Mxdir^*-0.013+Mvx^*0.063+Mact^*0.945+h^*1.002+e^*0.136$
186	$P(h)=vx^*-0.011+Mxdir^*0.004+Mvx^*0.246+Mact^*-0.293+h^*1.004+e^*0.01$
187	$P(h)=0.002+Mvx^*0.38+h^*1$
188	$P(h)=0.634+vx^*0+Mxdir^*-0.001+Mvx^*0.001+Mact^*0.005+h^*0.433+e^*-0$
189	$P(h)=vx^*0.001+Mxdir^*-0.135+Mvx^*0.011+Mact^*0.367+h^*0.985+e^*0.021$
190	$P(h)=9.113+vx^*-0.001+Mxdir^*0.002+Mvx^*0.013+Mact^*0.054+h^*0.191+e^*0.005$
191	$P(h)=3.339+vx^*-0.017+Mxdir^*0.074+Mvx^*0.007+Mact^*-0.013+h^*0.553+e^*0.245$
192	$P(h)=vx^*0.042+Mxdir^*-0.261+Mvx^*0.38+Mact^*0.421+h^*0.968+e^*0.029$
193	$P(h)=0.01+vx^*-0.005+Mxdir^*-0.324+Mvx^*0.314+Mact^*0.18+h^*0.999+e^*0.002$
194	$P(h)=vx^*0+Mxdir^*0.05+Mact^*-0.005+h^*1$
195	$P(h)=6.807+vx^*-0.003+Mxdir^*0.14+Mvx^*0.032+Mact^*0.066+h^*0.186+e^*0.008$
196	$P(h)=7.706+vx^*0.002+Mxdir^*-0.005+Mvx^*0.012+Mact^*-0.146+h^*0.199+e^*0.002$
197	$P(h)=0.11+vx^*-0.028+Mxdir^*-0.12+Mvx^*0.178+Mact^*0.203+h^*0.942+e^*0.049$
198	$P(h)=h^*1$
199	$P(h)=9.104+vx^*0.004+Mxdir^*-0.008+Mvx^*0.016+Mact^*0.102+h^*0.239+e^*0.006$
200	$P(h)=0+Mvx^*0.5+Mact^*0+h^*1$

Table 27: Known relation models at the time of final evaluation

ID	Relation Model
1	distdiff: stable, touch: false
2	distdiff: decreasing, touch: false
3	distdiff: increasing, touch: false
4	distdiff: stable, touch: true, dir: BOTTOM
5	distdiff: stable, touch: true, dir: LEFT
6	distdiff: stable, touch: true, dir: RIGHT
7	distdiff: stable, touch: true, dir: OVERLAP

Table 28: Known relation transition models at the time of final evaluation

ID	Relation Transition Model
1	from: 1, to: 1, t: 11938823
2	from: 2, to: 2, t: 2793329
3	from: 3, to: 3, t: 2942352
4	from: 4, to: 4, t: 1316310
5	from: 2, to: 1, t: 113142
6	from: 3, to: 1, t: 44829
7	from: 5, to: 5, t: 66949
8	from: 4, to: 3, t: 36109
9	from: 2, to: 3, t: 84694
10	from: 1, to: 3, t: 117186
11	from: 2, to: 6, t: 3336
12	from: 2, to: 4, t: 35517
13	from: 3, to: 2, t: 204417
14	from: 5, to: 3, t: 3432
15	from: 2, to: 5, t: 3331
16	from: 6, to: 3, t: 3447
17	from: 1, to: 2, t: 42228
18	from: 6, to: 6, t: 60825
19	from: 2, to: 7, t: 2021
20	from: 3, to: 4, t: 258
21	from: 1, to: 4, t: 422
22	from: 1, to: 6, t: 51
23	from: 7, to: 7, t: 29808
24	from: 7, to: 3, t: 1372
26	from: 1, to: 5, t: 50
28	from: 3, to: 7, t: 14
30	from: 3, to: 5, t: 22
31	from: 3, to: 6, t: 45
32	from: 5, to: 7, t: 1
34	from: 1, to: 7, t: 4



## Zusammenfassung in deutscher Sprache

In dieser Arbeit wird der Zusammenhang zwischen natürlicher Sprache und Weltwissen untersucht. Weltwissen bezieht sich dabei auf Wissen, das beschreibt, wie sich Dinge in der realen Welt normalerweise verhalten. Die Arbeit startet mit einem breit angelegten theoretischen Überblick über Sprache, Ambiguität, Bedeutung, Kognition und Weltwissen. Dabei wird die Hypothese formuliert, dass bei der Auflösung von Ambiguität und generell dem Verstehen natürlicher Sprache sogenannte *Information Gaps*, also Informationslücken auftreten, die eine Hörerin während der Sprachverarbeitung per Inferenz schließen muss. Statt Ambiguität sind Informationslücken während der Inferenz das eigentlich Problem und führen zu Missverständnissen.

Die Arbeit beschreibt LEARNA, ein Computersystem, das Event-Prädiktive Strukturen benutzt um Weltwissen zu sammeln und anzuwenden. LEARNA enthält außerdem ein einfaches Sprachverarbeitungssystem sowie ein Inferenzsystem. Zusammengekommen können mit diesen Systemen einfache Sätze natürlicher Sprache als Simulation reproduziert und in diesem Sinne *verstanden* werden. Insbesondere schafft es LEARNA Ambiguität aufzulösen, wo dies mithilfe von Weltwissen möglich ist.

Schließlich beschreibt diese Arbeit ein Experiment zu menschlicher Sprachproduktion. Die Studienteilnehmer haben dabei Weltwissen über eine einfache Spielwelt gelernt und anschließend Szenen aus dieser Spielwelt beschrieben. Die Hypothese war, dass Beschreibungen verkürzt werden, falls die beschriebene Szene den Erwartungen per Weltwissen entspricht. Falls die Szene überraschend ist, sollte die Beschreibung verlängert werden. Dieser Effekt konnte im Experiment nachgewiesen werden.