

Modeling Actions and State Changes  
for a Machine Reading Comprehension Dataset

by

Aurgho Bhattacharjee

A Thesis Presented in Partial Fulfillment  
of the Requirements for the Degree  
Master of Science

Approved July 2019 by the  
Graduate Supervisory Committee:

Chitta Baral, Chair  
Yezhou Yang  
Saadat Anwar

ARIZONA STATE UNIVERSITY

August 2019

©2019 Aurgcho Bhattacharjee  
All Rights Reserved

## ABSTRACT

Artificial general intelligence consists of many components, one of which is Natural Language Understanding (NLU). One of the applications of NLU is Reading Comprehension where it is expected that a system understand all aspects of a text. Further, understanding natural procedure-describing text that deals with existence of entities and effects of actions on these entities while doing reasoning and inference at the same time is a particularly difficult task. A recent natural language dataset by the Allen Institute of Artificial Intelligence, ProPara, attempted to address the challenges to determine entity existence and entity tracking in natural text.

As part of this work, an attempt is made to address the ProPara challenge. The Knowledge Representation and Reasoning (KRR) community has developed effective techniques for modeling and reasoning about actions and similar techniques are used in this work. A system consisting of Inductive Logic Programming (ILP) and Answer Set Programming (ASP) is used to address the challenge and achieves close to state-of-the-art results and provides an explainable model. An existing semantic role label parser is modified and used to parse the dataset.

On analysis of the learnt model, it was found that some of the rules were not generic enough. To overcome the issue, the Proposition Bank dataset is then used to add knowledge in an attempt to generalize the ILP learnt rules to possibly improve the results.

## DEDICATION

First and foremost, dedicated to Mummy and Papa, for their relentless love, support and understanding. Secondly, to all my close friends, peers, mentors and current and past teachers, whose endless discussions have always proved valuable and without whom, I'd probably never be here.

## ACKNOWLEDGMENTS

Throughout the work, I have received a great deal of support both professionally and personally and I would be amiss to not thank the people involved. First and foremost, I would like to thank my advisor, Dr. Chitta Baral, whose experience, support, encouragement and comments have been invaluable in the continuous improvement of my thesis.

I would also like to thank my colleague, mentor and friend, Arindam Mitra, for guiding, supporting and being a friend throughout the way. I would like to acknowledge my lab peers for being there for me. To be specific, I would also like to thank Ishan Shrivastava, for having endless discussions and both technical and non-technical during the course of the thesis; Tejas Gokhale, for discussions on multiple technical and sometimes philosophical doubts; Pratyay Banerjee, guiding us with minute details about frameworks and models; Arpit Sharma, Shailaja Sampat, Sam Rawal, Kuntal Pal, Vishnavi Batni, Ashok Prakash and Yiron Luo for their friendship, discussions and support.

I would also like to thank my parents for always being there for me and providing encouragement and support and being patient with me throughout the thesis. I would also like to thank my roommates and good friends, Shivam Agarwal, Sai Chavan, Pranjal Pangaokar and Akash Pawar for bearing with the “few times” I forgot to wash the utensils and for being transparent in communication whenever needed.

I would also like to thank my friends Channabasava Gola, Abhinetri Gowda, Thanmayi Rapolu, Santhoshi Inuganty, Nitin Kishore, Sai Prameela Konduru, Aditya Gopakumar, Pulkit Dobriyal, Pratyush Mahapatra and many other friends I might have forgotten to mention here for being there for me and to help provide happy distractions to rest my mind.

## TABLE OF CONTENTS

	Page
LIST OF TABLES .....	vii
LIST OF FIGURES .....	viii
CHAPTER	
1 INTRODUCTION AND MOTIVATION .....	1
1.1 About ProPara and Allen AI .....	2
1.2 Value of the Research .....	3
1.3 Research Evaluation .....	3
1.4 Contributions .....	4
1.5 Structuring of the thesis .....	4
2 BACKGROUND .....	6
2.1 Symbolic Logic .....	6
2.1.1 Propositional Logic .....	7
2.2 Answer Set Programming .....	7
2.3 Inductive Logic Programming .....	10
2.4 A Short Detailing on BERT .....	11
2.5 Machine Reading Comprehension .....	12
3 RELATED WORK .....	14
4 THE TRANSLATION LAYER .....	19
4.1 The QA-SRL parser .....	22
4.1.1 Modification to the QA-SRL Parser .....	23
4.2 Addition of Predicates .....	23
4.3 The Overall Translation Layer Flow .....	26
4.4 Reasoning .....	27

CHAPTER	Page
4.5 Critical points.....	30
5 THE LEARNING LAYER .....	32
5.1 A Brief Overview of ILP Rules .....	32
5.1.1 Learning Create Rules .....	33
5.1.2 Learning Destroy Rules .....	34
5.1.3 Learning Location Change Rules .....	36
5.2 Test-time prediction .....	37
5.3 Results .....	38
5.4 Error Analysis .....	39
6 ADDITION OF KNOWLEDGE .....	42
6.1 The Propbank Dataset.....	42
6.2 Sentence and Argument Extraction from Propbank .....	43
6.2.1 Sentences .....	44
6.2.2 Arguments and Roles .....	44
6.3 QA-SRL Parsing of Extracted Sentences .....	44
6.4 Obtaining Role Tagging of Questions .....	45
6.5 Results and Analysis .....	47
7 CONCLUSION AND FUTURE DIRECTION .....	51
7.1 Summary .....	51
7.2 Results and Discussion.....	52
7.3 Future Directions.....	53
REFERENCES .....	55
APPENDIX	
A SAMPLE CREATE RULES LEARNED BY ILP ON PROPARGA .....	60

CHAPTER	Page
B SAMPLE DESTROY RULES LEARNED BY ILP ON PROPARGA ....	63
C SAMPLE LOCATION RULES LEARNED BY ILP ON PROPARGA ...	66
D ANALYSIS OF RESULTS ON THE TEST SET OF PROPARGA .....	70
E DOMAIN KNOWLEDGE .....	78
F INERTIA RULES .....	80
G CODE REPOSITORY .....	83



## LIST OF TABLES

Table	Page
1. The Results of Various Operators on the Multiple Kinds of Values Taken by the Atomic Symbols A and B.....	7
2. Results of Models Run on the Locations Data .....	25
3. Results of Various Systems on the ProPara Dataset .....	38
4. Results of Various Systems on the ProPara Dataset .....	52

## LIST OF FIGURES

Figure	Page
1. Example Paragraph and Corresponding Location Tracking of Multiple Entities at Each Sentence .....	2
2. The Training Perspective of the Overall System .....	19
3. The Testing Perspective of the Overall System .....	20
4. A Simplified View of the Training Process .....	20
5. Example Paragraph and Corresponding Location Tracking of Multiple Entities at Each Sentence .....	22
6. QA-SRL Output for the Sentence 34The Roots Absorb Water and Minerals from the Soil 34 .....	23
7. Predicates Wrapped around the QA-SRL Outputs .....	24
8. The Translation Layer Data Flow .....	26
9. Upper Portion of the Example from before .....	27
10. Portion of the Example Discussed before .....	28
11. The QA-SRL Based Representation and the High Level Predicates of the Example Paragraph .....	29
12. A Second Example from the ProPara Dataset .....	33
13. Example of Create Event Rules .....	35
14. Example of Destroy Event Rules .....	36
15. Example of Move Event Rules .....	36
16. The Testing Perspective of the Overall System .....	37
17. The Detailed Results for ProKR Obtained Using ProPara Evaluator .....	39
18. A Snippet of the Absorb Frame XML File From Propbank .....	43

Figure	Page
19.The Roles Mentioned in the Absorb Frame File.....	43
20.The Roles and the Corresponding Phrases Extracted for 34The Habit Is Very Bad because Salt Absorbs Water from the Meat. 34 .....	44
21.The QASRL Parse of 34The Habit Is Very Bad because Salt Absorbs Water from the Meat. 34 .....	45
22.The Test Side Changes .....	45
23.Sample of PropBank Based Facts .....	46
24.Results Obtained on Modifying the Learned ILP Rules with 90% Similarity Replacement of Questions .....	46
25.Results Obtained on Modifying the Learned ILP Rules with 85% Similarity Replacement of Questions .....	47
26.Original Result without Adding Propbank Knowledge .....	48
27.Results Obtained on Appending to the Learned ILP Rules with 90% Similarity Replacement .....	48
28.Results Obtained on Appending the Learned ILP Rules with 85% Similarity Replacement of Questions .....	49
29.A Framenet Annotation for the Word Absorb .....	53

### INTRODUCTION AND MOTIVATION

There has been considerable interest in Artificial General Intelligence (AGI), a field that tries to mimic Human intelligence through machines, in the past 50 years. One of the key components in AGI that deals with processing text is that of Natural Language Processing (NLP). NLP deals with a variety of tasks such as language translation, speech recognition, etc. Natural Language Understanding (NLU), a key part of Natural Language Processing, deals with machine comprehension and understanding of a given text. There is considerable interest in NLU now due to its significant applications in “automated reasoning, question answering, news-gathering, [...] and large-scale content analysis” (*Natural-language understanding* 2019). One similar NLU task is the task of reading comprehension and its evaluation by question answering. On these lines, the ProPara dataset has been proposed by the Allen Institute of Artificial Intelligence to promote research in inference based question answering for a reading comprehension task (Mishra et al. 2018). Likewise, the main motivation to pursue this research is to come up with better inference, reasoning and comprehension based models. Current systems detect answers within paragraphs of question answering but are still unable to reason or infer well with text. Better results on this dataset can yield systems that can reason well, make good inferences and better keep track of entities or intents and their changes across time after multiple actions when performed on these entities. The developed system can incorporate background knowledge easily, reason with the constraints and the sentences and also provides an explainable model which makes it easy to decipher what the model has learnt.

		Participants:					
Paragraph (seq. of steps):		water	light	CO2	mixture	sugar	
<b>Roots absorb water from soil</b>	state0	soil	sun	?	-	-	Time ↓
<b>The water flows to the leaf.</b>	state1	roots	sun	?	-	-	
<b>Light from the sun and CO2 enter the leaf.</b>	state2	leaf	sun	?	-	-	
<b>The light, water, and CO2 combine into a mixture.</b>	state3	leaf	leaf	leaf	-	-	
<b>Mixture forms sugar.</b>	state4	-	-	-	leaf	-	
	state5	-	-	-	-	leaf	

Figure 1. Example Paragraph and Corresponding Location Tracking of Multiple Entities at Each Sentence

*Image Source:* Mishra et al. (2018)

### 1.1 About ProPara and Allen AI

The Allen Institute of Artificial Intelligence was founded by Paul Allen, Microsoft co-founder, to conduct “high impact research and engineering in the field of artificial intelligence” (*ProPara Dataset*). The authors have proposed the ProPara dataset to improve on question answering and reading comprehension. The Propara dataset “aims to promote research in Natural Language Understanding in the context of procedural text” (*Mission*). The dataset contains 488 procedural paragraphs (lending to its name) and 3300 sentences which describe specific scientific procedures.

As an example, Figure 1 shows a part of a paragraph from the training data. Here, we have the paragraph on the left and are tracking specific entities/participants in the dataset on the right. The table on the right attempts to capture the location of the entities at various points of time (Each new sentence is treated as a new time-step).

## 1.2 Value of the Research

The ultimate goal of quite a few Natural Language Processing systems is to be able to use “natural languages as effectively as humans do”[5]. Paths to this goal involve coming up with good machine comprehension and machine translation systems besides other challenges present in the community. Good machine comprehension systems should be able to understand a given paragraph of text and answer questions relevant to the text accurately as humans do. The bAbI dataset is a machine generated dataset that included questions about objects moved through a paragraph. The ProPara dataset attempts to motivate research in inference based procedural text question answering via human-authored and annotated paragraphs and question and answers. The thesis attempts to contribute to the research on Machine Comprehension methods and trying to improve performance on the ProPara dataset. Currently, the Machine Learning community is trying to figure out reasoning via statistical approaches. If accepted by the Machine Learning and NLP community, the approach will add to the toolbox of learning algorithms by adding an explainable method capable of reasoning, inference and easily incorporating background and nuanced constraints.

## 1.3 Research Evaluation

The approach will be evaluated on the ProPara test dataset of 54 paragraphs. The evaluation will be performed via a script provided by the dataset creators, expecting the input in a certain format and the script generates Precision, Recall and F1 scores based on the provided details. The expected format is a paragraph id followed by the time step and then the entity name, and then have the action and previous location

and then the next location. Each result that is obtained will be evaluated through the evaluator script.

#### 1.4 Contributions

This work makes the following contributions:

1. An approach to address the ProPara challenge using a modified semantic parser and an Inductive Logic Programming approach.
2. A knowledge base of verb-question pairs with role tags.

#### 1.5 Structuring of the thesis

The thesis is discussed into multiple chapters as follows:

1. **Background** The chapter discusses some background information that forms the basis of this work
2. **Related Work** Some additional details and most of the existing approaches for the ProPara dataset are discussed.
3. **The Translation Layer** discusses how the dataset is translated into a formal representation to process it later using the reasoning and learning system. The chapter also discusses details about some of the high level predicates used.
4. **The Reasoning and ILP Layer** discusses some details about the Learning based system.
5. **Addition of Knowledge** discusses the addition of knowledge in an attempt to improve performance.

6. **Conclusion and Future Direction** finally goes over the results and possible future works.

As mentioned, the next chapter dives into some of the background details to establish some of the foundational details.



## Chapter 2

### BACKGROUND

The previous chapter provided an introduction and motivation to the work. For this chapter, quite a few topics need to be covered to make sure that the reader is up-to-date on the topics connected together here. We need to go back all the way towards the beginning of Artificial Intelligence community efforts: towards symbolic learning and reasoning of topics. We need to do this because Artificial Intelligence was born as a field when ten researchers from multiple fields and colleges met at Dartmouth for a workshop covering intersecting interests (Russell and Norvig 2009). Later sections will then cover Answer Set Programming, Inductive Logic Programming, BERT and Machine Reading Comprehension to provide a firm base for the reader to understand the rest of the concepts. We now discuss about some of the research looked at by two of the ten scientists that were present at the Dartmouth Conference.

#### 2.1 Symbolic Logic

Two of the scientists at the Dartmouth Conference, Allen Newell and Herbert A. Simon, came up with the Physical Symbol System Hypothesis:

*“a physical symbol system has the necessary and necessary and sufficient means for general intelligent action”-* (Russell and Norvig 2009) The hypothesis tries to claim that any intelligent system should be able to manipulate relations and data structures relating to symbols that stand for real world objects. Some of the parts we

discuss next are what were referred to by John Haugeland as “Good Old-Fashioned AI”(Russell and Norvig 2009)

### 2.1.1 Propositional Logic

Propositional logic or Boolean logic, relies on truth values of variables. True and False always have the fixed boolean meaning. Meanwhile, atomic sentences are indivisible elements consist of proposition symbols that are defined by rules being defined. These rules are derived using some of the following operators:  $\neg$ ,  $\vee$ ,  $\wedge$ ,  $\Rightarrow$ ,  $\Leftrightarrow$ . The result of the operators on atomic inputs are mentioned below in table 1. The descriptive power of propositional logic increases by using predicates on top of the mentioned atomic symbols to represent more detailed description.

Table 1. The Results of Various Operators on the Multiple Kinds of Values Taken by the Atomic Symbols A and B.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

For more detailed reading about propositional logic, the reader is suggested to go through chapter 7 of the (Russell and Norvig 2009) text.

## 2.2 Answer Set Programming

Initially, Logic Programming was able to cover the requirement of a formal language being able to represent knowledge and also propagating inferences in the

knowledge base (Russell and Norvig 2009). Logic programming languages like ProLog, however, could not handle negation to discover solutions leading to the development of Answer Set Programming (ASP). The theoretical basis of handling negation came from the establishment of Herbrand models and stable model semantics as part of (Gelfond and Lifschitz 1988). In brief, canonical models are selected among models that satisfy a set of propositional rules, whereas a minimal and stable model is required to find solution to stable model semantics based rules. For example, for the following program from (Gelfond and Lifschitz 1988):

```
p(1),  
q(2),  
q(x) ← p(x)
```

$\{p(1), q(1), q(2)\}$  is a model but  $\{p(1), p(2), q(1), q(2)\}$  is a minimal model as well. A Herbrand Model is minimal, and in the example  $\{p(1), q(1), q(2)\}$  would be considered the minimal model and not the other model. To automate the process through a Programming language, multiple programming languages were spawned and Answer Set Programming is the paradigm of these languages. Some details of Answer Set Programming (hereafter, referred as ASP) have been discussed in (Lifschitz 2008) and it builds on top of the AI efforts as part of Symbolic AI, specifically on top of the stable model semantics just discussed and on answer set solving. ASP was introduced to be able to solve difficult search problems by the Knowledge Representation and Reasoning community . It generally consists of rules similar to first order logic, defining atoms, predicates and connecting atoms the operators mentioned in the previous section. A more advanced example that can help understand ASP better is the Yale Shooting Problem introduced in 1986 by Hanks and McDermott.

The problem consists of Fred the Turkey, who is initially alive and a gun is loaded at that time. We expect that loading the gun, waiting for a bit and then shooting the gun at Fred will kill the turkey. However, keeping track of the actions and determining the cause of that action was difficult at the time. Considering alive and loaded to be two predicates representing whether the turkey is alive and whether the gun is loaded; load and shoot to be two predicates that represent whether the gun is loaded and whether the gun was shot at the turkey, the set of predicates, as sourced from (Lee), will be represented as :

```
boolean(t;f).
alive(t,0).
loaded(f,0).
load(0) ⇒ loaded(1)
loaded(2) ∧ shoot(2) ⇒ alive(f,3)
```

Here, we need to make sure that the actions continue into the next time-step and is a little difficult to capture. The programming language that is chosen in this work for Answer Set Programming is Clingo (Gebser et al. 2014). For Clingo,

```
load(0) ⇒ loaded(1)
loaded(2) ∧ shoot(2) ⇒ alive(f,3)
```

would instead be written as :

```
loaded(1):-load(0).
alive(f,3) :- shoot(2).
```

In Clingo, the solution to the problem is captured as follows:

```
loaded(t,T+1) :- load(T), T=0..3.
alive(f,T+1) :- shoot(T), T=0..3.

% at any time step, an action is either true or false
!loaded(B,0):boolean(B)1.
```

```

1alive(B,0):boolean(B)1.

% at any time step, an action has to exist
:- not 1loaded(B,T):boolean(B)1, T=1..3.
:- not 1alive(B,T):boolean(B)1, T=1..3.

% action of load and shoot can either be true or false at each time step
load(T) :- T=0..3.
shoot(T) :- T=0..3.

alive(B,T+1) :- alive(B,T), T=0..3.
loaded(B,T+1) :- loaded(B,T), T=0..3.

```

Here, the final two rules are called rules of inertia because they enable states of alive and loaded to persist into the next time-step. Inertia based rules will later be used in this work as well to allow persistence of states. We now discuss the method of Inductive Logic Programming, a learning approach in the symbolic logic community.

### 2.3 Inductive Logic Programming

In Inductive Logic Programming (ILP), the objective is to be able to generalize over a given set of rules. As mentioned in (Ray 2009), ILP is “concerned with the generalisation of positive and negative examples with respect to prior background knowledge expressed in a logic program formalism” . Their XHAIL system combines abductive, inductive and deductive learning to learn and generalize rules in three phases. Here, given a background theory  $B$ , examples  $E$ , mode declarations  $M$ , it finds models such that

$$BUH \models E^+, BUH \not\models E^-$$

where  $\models$  represents the entailment operation.

The I<sup>2</sup>XHAIL system (Mitra and Baral 2018), on the other hand, converts each record in the dataset into a sample of the form  $(O_i, E_i^+, E_i^-)$  where  $O_i$  are logical representation of the records. I<sup>2</sup>XHAIL takes a sequence of samples  $S_1, S_2, \dots, S_n$  and outputs a set of rules such that the following holds:

$$O_iUBUH \models E_i^+, \forall i = 1..n$$

$$O_iUBUH \not\models E_i^-, \forall i = 1..n$$

ILP is later used as a learning framework (similar to how machine learning is used to learn over the dataset) in this work over a formal representation of the dataset.

## 2.4 A Short Detailing on BERT

When it comes to artificial neural networks, there has been significant work in the last two decades. Encoder and decoder based architectures have become very common in research now, especially attention based models. One such model which is currently very popular is that of the Transformer that combines multiple encoders and decoders (Vaswani et al. 2017). (Devlin et al. 2018) recently introduced a Transformer based model that achieved state of the art performance in eleven language based tasks. BERT can be used in multiple ways:

1. As a semantic textual similarity scorer between two sentences
2. As a Natural Language Inference model to score the possibility of a next sentence to occur compared to other sentences
3. As a text question answering model to answer questions for datasets like (Rajpurkar et al. 2016).
4. As a Parts of Speech tagger

As stated in (Devlin et al. 2018), BERT advanced the state of the art for eleven NLP tasks. BERT is later used as a natural language inference model to learn the locations of the dataset and to predict the locations during test time.

## 2.5 Machine Reading Comprehension

Machine Reading Comprehension (MRC) is currently one of the important tasks being looked into in Natural Language Understanding research to be able to judge whether human readable text is understandable by a system. The SQuAD dataset introduced by Stanford (Rajpurkar et al. 2016) presented questions where answers could be found as a span within the text. It’s successor, SQuAD 2.0 (Rajpurkar, Jia, and Liang 2018), extended the dataset by adding distractor sentences and sometimes unanswerable questions. The MS MARCO (Nguyen et al. 2016) dataset is a non-artificial MRC dataset which was introduced by Microsoft and is much larger than SQuAD. It is based on queries sampled from Microsoft Bing’s search logs with relevant passages annotated by human annotators. Some other notable recent MRC datasets are NarrativeQA (Kociský et al. 2017) and NewsQA (Trischler et al. 2016). Datasets by AllenAI are OpenBookQA (Mihaylov et al. 2018) which is a question answering task where additional knowledge is present but requires to collect and use additional knowledge to answer questions; AI2 Reasoning Challenge (Clark et al. 2018) which is the hardest dataset for knowledge and reasoning systems; SciTAIL (Khot, Sabharwal, and Clark 2018) which is a Science entailment dataset where the top model has already reached 96% accuracy and TextBook Question Answering (TQA)(Kembhavi et al. 2017) which is a multimodal (involving both images and text) dataset. The next

chapter expands on another AllenAI dataset, ProPara, how it was created and some of the recent work that was done on the dataset.



## Chapter 3

### RELATED WORK

In the previous chapter, some background topics were covered to form a foundation for this work. Towards the end of the chapter, we had discussed some reading comprehension datasets and some datasets by Allen AI as well. We now discuss one of the AllenAI datasets and some of the work done on the dataset. The motivation behind the making of the ProPara dataset was to build a Reading Comprehension system that can answer questions and may also require inference and reasoning. In essence, this procedural text dataset, is about tracking the temporal state changes of entities. It is also an attempt at creating a dataset made of natural text instead of synthetic text (example: in bAbI (Weston et al. 2015)).

**ProPara Creation** As mentioned in (Mishra et al. 2018) The dataset was created with the help of crowdsourcing workers from Mechanical Turk. The writers were first given a prompt about a process and were asked to write a sequence of sentences about the process. Each prompt was given to five annotators to produce five paragraphs but some were removed. Finally, 488 paragraphs describing 183 processes were shortlisted. These paragraphs were then given to annotators to fill participants first, creation and destruction annotations and finally, location annotations. The final number of annotations of the dataset were 81,345.

**EntNet** Recurrent Entity Networks (Henaff et al. 2016) or EntNet is a model made by the Facebook AI Research Group which is meant to track world states. It tracks states by maintaining a dynamic long-term memory allowing it to maintain and adapt a representation of the state of the world. The dynamic memory is achieved by

multiple blocks of a gated recurrent network. The modified Gated Recurrent Network is said to be sharing similarities with models such as the BiLSTM, GRU, DNC/NTM framework and the a Memory Network and its other variants. The model had achieved state-of-the-art results in the bAbI dataset during its time. EntNet was implemented as a baseline model in (Mishra et al. 2018) and the F1 was reported to be 39.40 %.

**Query Reduction Networks** or QRN is another model implemented as a baseline in (Mishra et al. 2018) and was introduced in (Seo et al. 2016) as a variant of the Recurrent Neural Network (RNN) to be able to perform in question-answering tasks. A unit of the QRN is slightly similar to a GRU it takes in input and query and outputs a hidden state. In the unit,  $\alpha$  and  $\rho$  are update gate and reduce functions respectively which are similar to gates used in LSTMs and GRUs. On a high level, the model takes in a vector representation of the question and the story and predicts an answer for it. For its time, similar to EntNet, QRN was the state-of-the art on the bAbI dataset as well. On ProPara, the model achieved 41.10 % F1, slightly better than the EntNet model’s 39.40.

**ProLocal and ProGlobal** The original paper proposed two models, ProLocal and ProGlobal, to solve the dataset (Mishra et al. 2018) after talking about the two baseline models above. ProLocal uses a biLSTM architecture for local state predictions and algorithmically propagates these predictions to perform commonsense persistence. The input takes in the vector of each word, concatenates it with a signal indicating if the word is an entity, and another signal to indicate if it’s a verb. The encodings from the LSTM are used to create the outputs by using bilinear attention.

Meanwhile, ProGlobal incorporates the persistence of the state information inside the neural network model. ProGlobal achieves an F1 accuracy of 51.9 % while ProLocal manages 50.70 %.

**ProStruct** Another paper proposed by AllenAI, proposes the model ProStruct, which uses an encoder-decoder as its core model and incorporates commonsense knowledge to improve predictions (Tandon et al. 2018). The encoder process is similar to the process involved ProLocal, where the word vectors are appended by the signals of the word being an entity or not and being a verb or now. It uses bilinear attention to generate actions. The commonsense background knowledge is incorporated using hard (boolean) constraints and soft (statistical) constraints over the dataset. Hard commonsense constraints consider commonsense facts such as the fact that an entity must exist before it can be created or destroyed. These constraints are considered by using a Boolean function. The soft constraints are incorporated through prior probabilities and are used to rank the action sequences of destroy, move and create. The decoder outputs the actions in the end. ProStruct manages to improve over ProGlobal and scores an accuracy of 54.5% on the ProPara test data, slightly better than ProGlobal’s 51.9.

**KG-MRC** Meanwhile, a recent paper by researchers from Microsoft Research Montreal and University of Massachusetts, Amherst, achieved an accuracy of 57.6 % (Das et al. 2018). In short, they encode the paragraph using a bidirectional LSTM. They use a bipartite graph to keep track of the entities and locations and keep updating the graph at each time-step. They query the Machine Reading Comprehension Module to keep updating the knowledge graph at each time step for each required entity.

**NCET** A recent paper from University of Texas at Austin achieved state of the art on the dataset at an F1 of 62.5% (Gupta and Durrett 2019) after the work in this thesis was implemented. The paper makes use of BiLSTM and Conditional Random Fields to model the dataset. The BiLSTM is used to “distill” information, i.e. it encodes the text and participants of the paragraphs in the dataset. In essence, the

model does well since it is able to mathematically capture the constraints of the problem and because it models the dataset/problem similar to that of a sequence labelling task. The significant jump could be due to the use of a CRF, helping the model learn the sequence of labels better or also because a per entity based model is used, leading to a bigger jump in accuracy. Besides, using ELMO embeddings (Peters et al. 2018) in their work contributed to a 4% jump in scores.

**AQA** or the Analogical Question Answering paper is a cognition theory based paper from NorthWestern University which is based off of the “Companion Cognitive Architecture”, consisting of a “large knowledge base and a general-purpose semantic parser” (Ribeiro et al. 2019). It is a non-neural model uses symbolic language and ontologies. The model is a combination of Step Semantics and AQA to bridge natural language semantics and task semantics. The AQA approach generates relational representations from the input sentences in the form of logical statements and uses a supervised learning approach. The system seems possibly similar to the system discussed in this work but both works have been developed independently. AQA achieves an accuracy of 52.30 %. The paper plans to improve by better handling questions requiring common-sense knowledge by using a knowledge base called NextKB .

**LaCE** LaCE or Label Consistency Explorer is an interesting approach from Allen AI as the approach better leverages training data and combines it with an unsupervised training approach (Du et al. 2019). The approach is to leverage the property that some paragraphs talk about similar process which means that there are some aspects of the process enabling the model to take advantage of label consistency across paragraphs. The LaCE framework operates on batches of grouped examples. It then tries to optimize two kinds of losses: the usual supervised loss for the primary example of the

group as well as the consistency loss for all the other members of a batch. Optimization of these two losses leads to a model that is both accurate as well as consistent across the batches with similar examples.

**Similar datasets** Some datasets related to ProPara are the bAbI dataset and the recipes dataset. The bAbI dataset was proposed by Facebook Research and consists of 1000 questions for training and 1000 questions for testing and helped advance methods of reasoning over text. The recipes dataset (Kiddon et al. 2015), meanwhile, contains 2456 recipes as paragraphs and contains multiple states such as shape, composition and location and quantity of multiple ingredients (which can be considered as entities) which have been hand annotated.

To conclude, this chapter discussed how the ProPara dataset was created and also discussed some of the approaches in solving the dataset. The next chapter dives into the details of the system used for the work in this Thesis.

## THE TRANSLATION LAYER

Having covered some background and relevant work done on the dataset until now, we begin to dive into the details of the system used to solve the dataset in this chapter. We first gloss over the high level details of the system. Section 4.1 discusses the QA-SRL parser, while section 4.2 dives into the different kinds of predicates that are considered for the system. Section 4.3 discusses the overall high level translation layer flow and how we can use these predicates to arrive at the answers that are needed. We now discuss the high level details of the system.

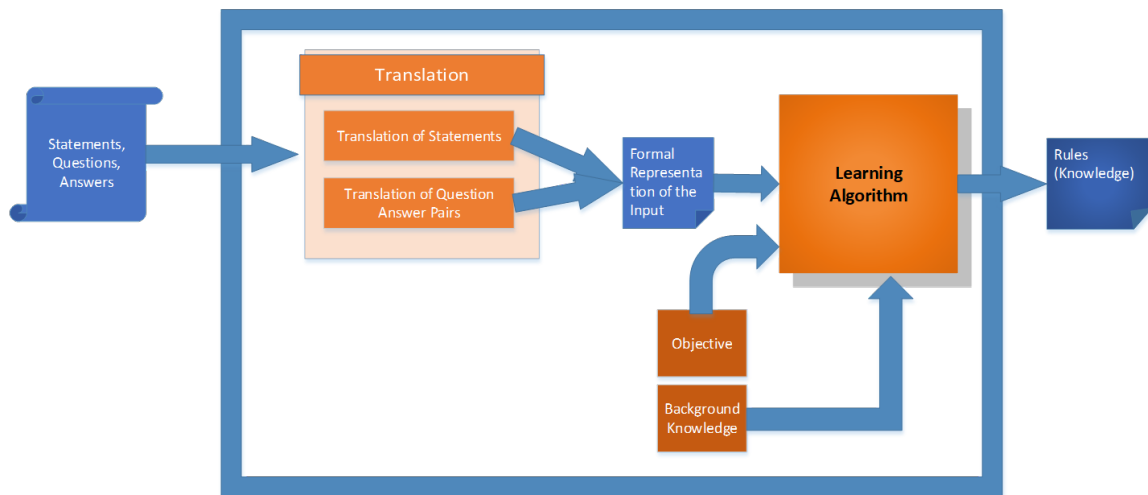


Figure 2. The training perspective of the overall system

There are two phases to the system that are shown in the figures 2 and Figure 3. Figure 2 refers to the training perspective of the system where the Translation module modifies the paragraphs into a set of facts to obtain a formal representation of the

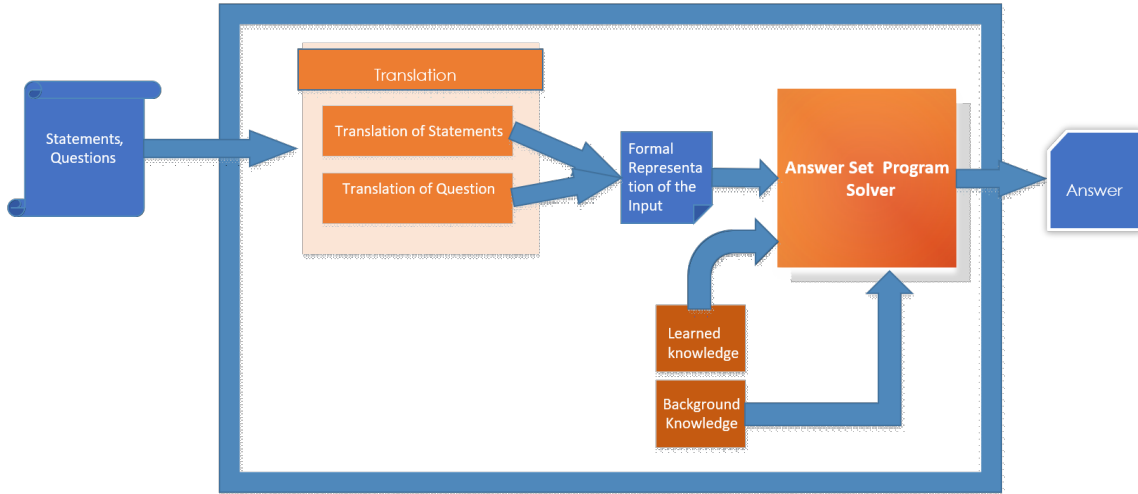


Figure 3. The testing perspective of the overall system

text. The learning algorithm, ideally, learns the right set of rules from the collection of rules, background knowledge and objectives to learn the high level rules we need. Figure 3 shows the test time perspective of the system where the learned rules are used along with the test-time translated paragraphs and background knowledge to generate the answers needed using the reasoning system.

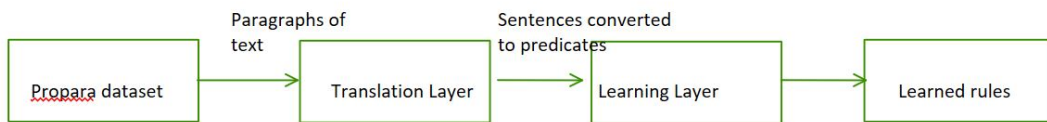


Figure 4. A simplified view of the training process

The general data flow of the system is represented above in Figure 4. In this chapter, we discuss the Translation layer module of the system while the Learning module and the reasoning module is discussed in the next chapter. The translation layer modifies the ProPara dataset, both for training and testing, to a formal representation that is consume-able by the Reasoning and the Learning layers.

There are multiple parsers and datasets that can be used to obtain a formal representation. Some choose to go with:

1. The Abstract Meaning Representation (AMR) (Banarescu et al. 2013) which has roles and sentence fragments based on verbs and consists of over 20k sentences. The AMR Corpus is made of sentence from the Proposition Bank and presents “rooted, edge-labelled, leaf-labelled graphs”. It makes use of the PropBank frames, has relations between quantities, relations for date entities and lists and includes semantic relations between entities as well. It also contains argument-predicate structures and inverse relations.
2. The semantic Role Labelling (Palmer, Gildea, and Xue 2010) based parsers or models which are based off the FrameNet database (*framenet2.icsi.berkeley.edu*) among other datasets consists of approximately 200k sentences. The FrameNet database was made to be used for training of Machine Learning models. It consists of multiple parts: 1. A lexicon of dictionary-type data, formulas, links to annotated examples and links to other machine readable resources such as WordNet. 2. Frame database consisting of frame related information such as names and descriptions of the “Frame Elements”. 3. Multiple annotated example sentences.
3. The Question Answering Semantic Role Labelling Parser (QA-SRL) (FitzGerald et al. 2018), which is trained on over 250k sentences. It generates question-answer pairs for multiple argument-predicate pairs.

The QA-SRL parser is used since the representation represents what is needed and is trained over most sentences ensuring the best possible generalizability across sentences. Initially, I make use of a semantic role label parser called QA-SRL (FitzGerald et



al. 2018) to obtain a formal representation of the dataset in terms of a question - answer meaning based representation.

For the reader’s reference and convenience, the example used in the introduction is reintroduced here.

		Participants:				
Paragraph (seq. of steps):		water	light	CO2	mixture	sugar
	state0	soil	sun	?	-	-
<b>Roots absorb water from soil</b>						
	state1	roots	sun	?	-	-
<b>The water flows to the leaf.</b>						
	state2	leaf	sun	?	-	-
<b>Light from the sun and CO2 enter the leaf.</b>						
	state3	leaf	leaf	leaf	-	-
<b>The light, water, and CO2 combine into a mixture.</b>						
	state4	-	-	-	leaf	-
<b>Mixture forms sugar.</b>						
	state5	-	-	-	-	leaf

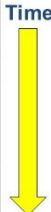


Figure 5. Example Paragraph and Corresponding Location Tracking of Multiple Entities at Each Sentence

*Image Source:* Mishra et al. (2018)

#### 4.1 The QA-SRL parser

Semantic role labelling is the process that assigns labels to words or phrases in a sentence that indicate their semantic role. To represent the text of ProPara, we use the QA-SRL parser.

The QA-SRL parser, represents a sentence in terms of multiple question-answer pairs. For a given, sentence, we obtain multiple question-answer pairs for each verb present in the sentence.

For example, in Figure 6, the QA-SRL output of the sentence input “The roots absorb water and minerals from the soil” which is similar to the sentence in the previous figure is shown.

---

```
"absorb"  
"What absorbs something?"           "The roots"  
"What does something absorb?"       "water and minerals"  
"Where does something absorb something from?" "from the soil"  
"What does something absorb something from?" "the soil"
```

Figure 6. QA-SRL output for the sentence “The roots absorb water and minerals from the soil”

#### 4.1.1 Modification to the QA-SRL Parser

The original QA-SRL parser could not produce questions for be verbs such as is, was, will do, etc. The original code of the parser was modified to now use the Stanford-Core Library (Manning et al. 2014) instead of the Spacy library to recognize “be” verbs by using the dependency parser and the part of speech tagger. The modified code is available at (Bhattacharjee 2018).

#### 4.2 Addition of Predicates

After obtaining the question-answer representation of the sentences by parsing through QA-SRL, all the question-answer pairs were programmatically wrapped under an `OberservedAt(verb, question, answer, time)` predicate where the time represents the sentence number in the paragraph. All verbs and questions were lemmatized using the Stanford CoreNLP lemmatizer to ensure a degree of uniformity for the verbs and sentences. As an example, the representation of the sentence in the previous example would be represented as shown in the figure. Here, assuming that the sentence in Figure 6 is the first sentence of the paragraph, the predicates representing it are shown in Figure 7.

```
observedAt("absorb", "what v something ?", "the roots", 1).
observedAt("absorb", "what does something v ?", "water and minerals", 1).
observedAt("absorb", "where does something v something from ?", " from the soil", 1).
observedAt("absorb", "what does something v something from ?", "the soil", 1).
```

Figure 7. Predicates wrapped around the QA-SRL outputs

The answers of the QASRL outputs and the location annotations in the dataset are used to keep a location(answer, loc) predicate and are added if any of the location text is a subphrase of the answer text. These location predicates are used to construct an “Is it a location?” model using BERT’s (Devlin et al. 2018) entailment (inference actually) model to check if a sub-phrase of an answer is a location during the testing phase. The BERT model that was used got a validation accuracy of 85.14 % and a test accuracy of 88.25 %. One example annotation that would get added here would be location(“the roots”, “roots”) where “the roots” is taken from the question-answer pair (“what absorbs something?”, “the roots”) obtained from the first sentence. Meanwhile, “roots” is an annotation for the first sentence for the “water” participant.

Just to make sure that the best entailment model was being used, the locations data was also run through some other models as well. One model that was run through the locations data was the decomposable attention model (DeCatt) which was one of the state of the art models on the SNLI dataset (Parikh et al. 2016). The model has three major components: attend, compare and aggregate. In attend, a word-to-word attention mechanism is used to calculate weights. These attention weights are used to calculate dot product similarity between word vectors of premise and hypothesis. These attention weights are used to calculate softly aligned sub-phrases. The softly aligned sub-phrases are used to perform comparisons which are aggregated to create a final inference vector. The final inference vector is fed to a feed forward neural network to make the final inference.

The ESIM, or Enhanced Sequential Inference Model (Chen et al. 2016), on the other

hand, uses bidirectional LSTMS to similarly localize, compose and pool vectors. The enhancement part refers to concatenating the element-wise difference and products with the softly aligned vectors in the compse layer. Another enhancement used was the addition of context via the biLSTM’s used. The results obtained by using ESIM, DeCatt and BERT models are present below in Table 2.

Table 2. Results of models run on the locations data

	Test Accuracy (%)
Decomposable Attention	85.81
ESIM Glove	87.58
ESIM ELMO	86.57
BERT	<b>88.25</b>

Besides, more annotations relevant to the paragraph are added : the groundings of range of time as time(t1...tn), number of participants participant(p1;..pk), descriptions of each participant description(pk; pdescription), number of annotators involved with annotator(a1;...ap) and the location annotations made by the annotators for each participant for each point of time: annotation(ap, pk, t). For the running example in Figure 5 (Image sourced from (Mishra et al. 2018)), the facts:

time(1...5).

participant(p1;p2;p3;p4;p5).

description(p1, “water”).

description(p2, “light”).

description(p3, “CO2”).

description(p4, “mixture”).

description(p5, “sugar”).

are added. For each annotator ap, an annotation for the location is added. For example, if annotator ap indicated that the participant “water” is in “soil” at timepoint

1, the predicate annotation( $ap, p1, 1$ ) is added. One may wonder what happens if some annotators vote differently for the same participant and the same time-point: in such cases, a penalty system is maintained and the higher the voting for a location, the lower the penalty and vice-versa. The paragraph representation also contains a  $refers(pi, A, T)$  predicate if the answer  $A$  contains a reference to a participant  $pi$  in the event at time  $T$ . Simple word overlap is used to generate the  $refers$  predicates. Each paragraph is represented with each of the predicates mentioned above. Thus, for the running example, there would be multiple  $observedAt$  predicates for each participant and the QA-SRL outputs.

### 4.3 The Overall Translation Layer Flow

As represented in Figure 8, a paragraph is first run through QA-SRL and then represented in predicate form. The predicate form is then fed into the Answer set programming and ILP module to produce rules generalized over the dataset.



Figure 8. The Translation Layer Data Flow

To track states of participants, it is needed to know whether the events in a sentence creates, destroys or moves a participant. The following high level predicates and their causes need to be discovered:

1. create(P,T): Participant P is created at timepoint T.
2. destroy(P,T): Participant P is destroyed at timepoint T.
3. beforeLocation(P,L,T): Participant P is at location L before timepoint T.
4. afterLocation(P,L,T): Participant P is at location L after timepoint T.
5. terminate(P,T): Participant P at timepoint T changes location but destination location is unknown.

We want to be able to discover the above mentioned high-level predicates by reasoning with the facts that were obtained in Section 4.2 above.

<b><i>The water flows to the leaf.</i></b>						
	state2	leaf	sun	?	-	-
<b><i>Light from the sun and CO2 enter the leaf.</i></b>						
	state3	leaf	leaf	leaf	-	-

Figure 9. Upper Portion of the example from before

#### 4.4 Reasoning

As an example, it can be shown how locations are propagated across participants for portion of the example from 5 shown in Figure 11. Here, the following observations will be derived after the QA-SRL parse:

```
observedAt("enter","what v something ?","light from the sun and co2",3).
```

```
observedAt("enter","what does something v ?","the leaf",3).
```

We also assume the following refers and locations to be true:

```

refers(4, "the light , water and co2", "light").
refers(4, "the light , water and co2", "co2").
refers(4, "the light , water and co2", "water").
location("the leaf", "leaf").

```

For each participant in the above snippet, a generalized reasoning for a time T would be:

```

If observedAt("enter", "what does something v?", "leaf", T) and refers(T,A,P)
then afterLocation("light", "leaf", T)

```

<b>Light from the sun and CO2 enter the leaf.</b>						
	state3	leaf	leaf	leaf	-	-
<b>The light, water, and CO2 combine into a mixture.</b>						
	state4	-	-	-	leaf	-
<b>Mixture forms sugar.</b>						
	state5	-	-	-	-	leaf

Figure 10. Portion of the example discussed before

As an example, continuing the example discussed before as shown in Figure 11, assuming that the location of the "mixture" participant is "leaf" after it's formation in time-point 4. If we take the facts and reason with them, the following fact is true:

```

afterLocation("mixture", "leaf",4).

```

which leads us to reason the following facts to be true:

```

If afterLocation("mixture, "leaf", 4) then initiate(locationOf("mixture", "leaf",5).

```

Additional constraints of destruction of mixture if sugar is formed would instead override the location propagation from before, for example:

```

If observedAt("form","what does something v ?","sugar",5)
    and observedAt("form","what v something ?","mixture",5)
then initiate(locationof("mixture", "-", 5)).

```

Specific facts would help support deduce the fact that sugar was created:

```

If observedAt("form","what does something v ?","sugar",5) and refers("sugar", "sugar", 5)
then create("sugar", 5).

```

We will also need to reason out the location of where the sugar was formed:

```

If observedAt("form","what does something v ?","sugar",5)
    and observedAt("form","what v something ?","mixture",5)
    and locationOf("mixture","leaf",4)
then initiate(locationOf("sugar","leaf",5)).

```

The translation along with the high level predicates of the paragraph is shown below. The predicates besides the observedAt are excluded just to present the flow.

Sentences	QA-SRL based representation	High level representation
Roots absorb water from the soil.	observedAt("absorb", "what v something ?", "roots", 1). observedAt("absorb", "what does something v ?", "water", 1). observedAt("absorb", "where does something v something from ?", "from the soil", 1).	beforeLocation(p1, "soil", 1). afterLocation(p1, "roots", 1).
The water flows to the leaf.	observedAt("flow", "what v somewhere ?", "the water", 2). observedAt("flow", "what v ?", "water", 2). observedAt("flow", "where does something v ?", "to the leaf", 2). observedAt("flow", "what does something v to ?", "the leaf", 2).	afterLocation(p1, "leaf", 2).
Light from the sun and CO2 enter the leaf.	observedAt("enter", "what v something ?", "light from the sun and co2", 3). observedAt("enter", "what does something v ?", "the leaf", 3).	beforeLocation(p2, "sun", 3). afterLocation(p2, "leaf", 3).
The light, water and CO2 combine into a mixture.	observedAt("combine", "what v into something ?", "light, water and co2", 4). observedAt("combine", "what does something v into ?", "mixture").	create(p4, 4).
Mixture forms sugar.	observedAt("form", "what v something ?", "mixture", 5). observedAt("form", "what does something v ?", "sugar", 5).	create(p5, 5).

Figure 11. The QA-SRL based representation and the high level predicates of the example paragraph



## 4.5 Critical points

The ASP system uses the learned event-centered knowledge to first extract a high level representation from the observedAt predicates. It then uses the high level representation to predict a state sequence while making use of the notion of a *critical point*. For a participant P, a time-point T is a critical time point if any of the following is true:

1. create(P, T-1)
2. destroy(P, T-1)
3.  $\exists L$ .afterLocation(P, L, T-1)
4.  $\exists L$ .beforeLocation(P, L, T)
5. terminate(P, T-1)

The state at a non-critical time point is then computed with a set of inertia rules on a case-by-case basis. For any participant P, in a non-critical time point T only one of this must be true:

**Case 1:** There exists no critical point for P before or after T.

**Case 2:** There is no time point before T which is a critical point for P but there is one after it.

**Case 3:** There is no time point after T which is a critical point for P there is one before it.

**Case 4:** There is a critical point for P both before and after T.

In the previous section, it took quite some effort to discover the high level rules. However, using the information about critical points to come up with inertia rules

defined in Appendix F which are similar but more complicated than inertia rules discussed in 2.2 will help learn high level details of a paragraph better.

However, it won't help to generalize and learn patterns from the paragraphs. We want to be able to generalize and learn across multiple paragraphs in order to be able to discover generic high level predicates rather than reason to the high level predicates specific to each paragraph, leading us to our next chapter.

The next chapter discusses how we can automatically learn generalized rules over multiple paragraphs instead of trying to manually figure out predicates to reason to the results.

### THE LEARNING LAYER

As discussed in the last chapter, the system created uses Answer Set Programming to reason between the multiple predicates that are created from the dataset along with some background knowledge and generate the answers during testing time. However, it still needs to be able to learn to generalize across multiple paragraphs. Thus, during the training time, Inductive Logic Programming (ILP) is used to discover the causes for the higher level predicates. The ILP system, I<sup>2</sup>XHAIL, is an improvement over (Ray 2009) used and discussed in (Mitra and Baral 2018) Another example that will be referred to in this chapter is shown below in Figure 12. The chapter is structured as follows: Section 5.1 briefly goes over the ILP process, Section 5.2 discusses prediction during the test-phase of the system. Section 5.3 discusses the results obtained while Section 5.4 discusses a high level error analysis. We not dive into some of ILP process.

#### 5.1 A Brief Overview of ILP Rules

We need rules for create, destroy and for location changes as part of ProPara. ILP learns this in three phases of abduction, deduction and induction for the create and destroy rules. It learns the create-related rules in the following manner:

Paragraph ( Seq. of steps)	Participants			Time
	waves	rocks	tiny parts of rocks	
Water from the ocean washes onto beaches in waves.	ocean	beach	-	1
The waves contain sediment from the ocean.	beach	beach	-	2
The water and particles in it hit rocks and sand on the beach.	beach	beach	-	3
Tiny parts of the rocks come off the larger rocks.	beach	beach	-	4
The waves pick up sand and small rocks.	beach	beach	beach	5
The waves go back out into the ocean.	beach	beach	wave	6
	ocean	beach	ocean	7

Figure 12. A second example from the ProPara dataset

### 5.1.1 Learning Create Rules

**Abduction** In the abductive step, it finds out several minimal collection of grounded create or destroy atoms. These sets are abducibles. For the example in Figure 12, one such abducible would be `create(p3; 4)`.

**Deduction** In the deductive step, it considers the possible causes of each ground predicate in  $\Delta$  and creates the most specific rule for each of them by adding the possible causes into the body of a rule. For the example regarding rocks, the deduced causes will look like the following:

```
IF observedAt("come","what v off something ?",
  "tiny parts of the rocks",4), observedAt
  ("come","what does something v off?","the
  larger rocks",4), refers(p3,"tiny parts of
```

```

the rocks",4), refers(p2,"the larger rocks",4)
,describe(P2,"rocks"), describe(p3,"tiny
parts of the rocks")
THEN create(p3,4)

```

**Induction** In the third stage, it tries to generalize the rule as much as possible by replacing the constants by variables. For the rocks based example, it learns the following in the Inductive step.

```

IF observedAt("come","what v off something ?",X,T),
  refers(E,X,T)
THEN create(E,T)

```

Next, it takes a similar sample S2 and comes up with a similar rule. After that, it tries to come up with a better hypothesis that can satisfy both the samples. Figure 13 shows a few other examples of create rules that are learnt. In the figure, the first three rules just describe a create event while the next few rules describe the creation of a specific participant. A bigger sample of create rules that were learnt is present in Appendix A

### 5.1.2 Learning Destroy Rules

The process of learning destroy rules is similar to that of learning create rules but instead of just having just destroy rules, it is split into two parts: learning two predicates “normallyDestroys” and “exception”, which are learnt together. A generic example rule:

```

IF normallyDestroy(P,T),not exception(P,T)
THEN destroy(P,T)

```

create(P,T) IF observedAt("provide","what does something v ?",X,T), refers(P,X,T).
create(P,T) IF observedAt("form","what is being v ?",X,T),refers(P,X,T).
create(P,T) IF observedAt("make","what does someone v ?",X,T),refers(P,X,T).
create(P,T) IF observedAt("evaporate","what v ?","water",T),description(P,"water vapor").
create(P,T) IF observedAt("melt","what v something ?","the cans",T),description(P,"molten metal").
create(P,T) IF observedAt("grow","what v ?","the pupa",T),description(P,"adult butterfly").

Figure 13. Example of create event rules

In the above rule, if an event normally destroys a participant P, it is assumed to be destroyed unless it is an exceptional case. For an example from Figure 14, if an item is eaten, then it is normally destroyed. To expand further on the figure, the first three rules describe how verbs like “eat”, “decompose” and “form” normally describe events where some kind of destruction takes place. The fourth and fifth rules capture exceptions like when a caterpillar forms a cocoon, it does not destroy the caterpillar. Similarly, when caterpillar uses saliva, it does not always destroy things unlike how saliva normally destroys food. The next two rules capture the events like “pupa is destroyed when the pupa hatches” and “magma is destroyed when it flows and becomes lava”. Rules like the first three that were just discussed (rules for verbs like “eat”, “decompose”) are small in number and can be learned properly from a dataset like ProPara. A bigger background knowledge can help learn about exception rules better. A sample of more destroy rules is present in Appendix B.

---

normallyDestroy(P,T) IF observedAt("eat","what does someone v ?",X,T),  
 refers(P,X,T).

---

normallyDestroy(P,T) IF observedAt("decompose","what v ?",P,T).

---

normallyDestroy(P,T) IF observedAt("form","what v something ?",P,T).

---

exception(P,T) IF normallyDestroy(P,T), description(P,"caterpillar"),  
 create(P1,T), description(P1,"cocoon").

---

exception(P,T) IF normallyDestroy(P,T), description(P,"caterpillar"),  
 create(P1,T), description(P1,"saliva").

---

normallyDestroy(P,T) IF observedAt("hatch","what v ?","the  
 cocoon",T),description(P,"pupa").

---

normallyDestroy(P,T) IF observedAt("flow","what v from something  
 ?","magma",T), description(P,"magma"), observedAt("flow","how does  
 something v from something ?","in the form of lava",T).

---

Figure 14. Example of destroy event rules

### 5.1.3 Learning Location Change Rules

---

locationBefore(locationOf(P,L),T) IF lobservedAt("travel","what does  
 something v from ?",L,T), eobservedAt("travel","what v ?",P,T).

---

locationBefore(locationOf(P,L),T) IF lobservedAt("fill","where is something  
 being v from ?",L,T), eobservedAt("fill","what is something v with ?",P,T).

---

locationAfter(locationOf(P,L),T) IF eobservedAt("put","what is v ?",P,T),  
 lobservedAt("put","what is something v in ?",L,T).

---

terminate(locationOf(P),T) IF eobservedAt("evaporate","what v from  
 something ?",P,T).

---

terminate(locationOf(P),T) IF eobservedAt("launch","what is being v ?",P,T).

---

locationAfter(locationOf(P,L),T) IF eobservedAt("turn","what does  
 something v into ?",P,T), value(L,"atmosphere").

---

Figure 15. Example of move event rules

The location change events do not depend on the participants in the domain of ProPara unlike how the create and destroy rules do. The gold create and destroy events; the background knowledge and state description are used to identify locationAfter, terminate and locationBefore events. Some of the location change or move event rules are presented above in Figure 15. There are two kinds of move events: the first type is present in the first five rules in the figure and are normally used to describe events that involve a change in location. The second kind of rule is where a participant and

the event together determine the location of a participant in the next time-point and as an example is present in the final rule in the figure where water is the participant and the event involved is water turning into vapor which determines that the vapor would be present in the atmosphere, the location of the new participant. A sample of more location based rules is present in Appendix C.

## 5.2 Test-time prediction

During the test phase of running the system, each text paragraph is first run through the Translation layer described in the previous chapter to represent the paragraph in terms of a formal representation. The formal representation combined with the domain knowledge and inertia rules is run through the Answer Set Programming system to produce location predictions. These predictions are programmatically modified to be runnable by the ProPara evaluator. The high level test system is shown below in Figure 16.

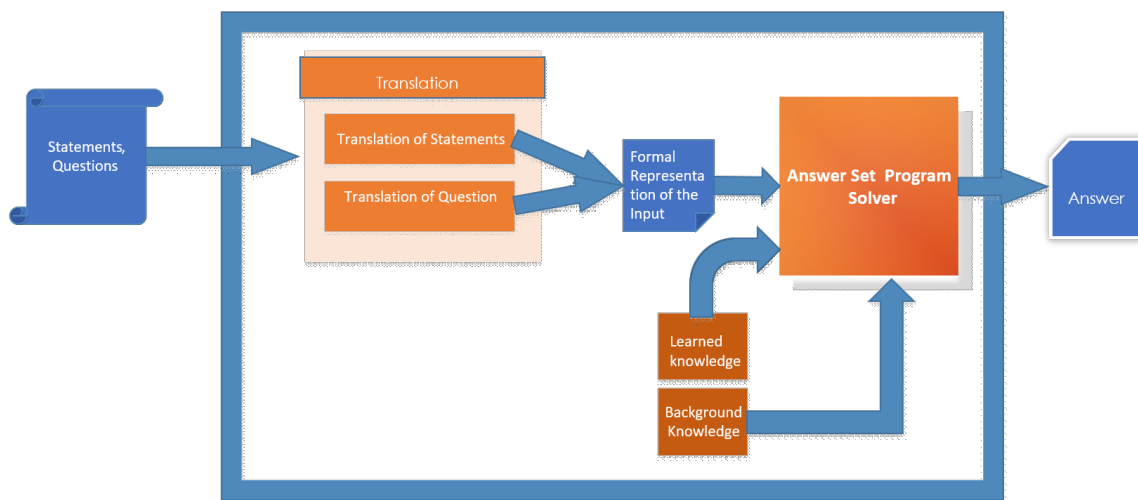


Figure 16. The testing perspective of the overall system



### 5.3 Results

The system was evaluated on the test dataset of 54 paragraphs using the creators’ evaluation script which takes location predictions of multiple entities across timepoints over the paragraphs and evaluates results based on the input objects, output objects, location changes, creation and destruction of participants. A comparison of the various systems is presented below in Table 3.

Table 3. Results of Various Systems on the ProPara Dataset

	Precision	Recall	F1
ProLocal	77.4	22.9	35.3
QRN	55.5	31.3	40.0
EntNet	50.2	33.5	40.2
ProGlobal	46.7	52.4	49.4
ProStruct	74.2	42.1	53.7
KG-MRC	64.52	50.68	56.77
<b>ProKR (This work)</b>	<b>76.00</b>	<b>45.10</b>	<b>56.60</b>
NCEt (later)	67.10	58.50	62.50
LACE (later)	75.30	45.40	56.60

The detailed results obtained through the evaluator (available at <https://github.com/allenai/aristo-leaderboard/tree/master/propara>) for the outputs generated by the system are visible below in Figure 17. Here, the results truly needed are the overall precision, recall and F1. Inputs, in the figure, refers to entities that were already present from the beginning and checks whether the system tracks them well. Similarly, outputs represent entities that were not stay and are not destroyed by the end of the process. Conversions represent entities that were converted from one form to another. Moves track only location changes along the way. All solvers should essentially be able to answer the following questions:

- (Q1) What are the inputs to the process?
- (Q2) What are the outputs of the process?
- (Q3) What conversions occur, when and where?
- (Q4) What movements occur, when and where?

From the results breakdown, it can be seen that moves and conversions recall and F1 scores are lower than 0.5 implying significant scope for improvement there.

```

=====
Question      Avg. Precision  Avg. Recall  Avg. F1
-----
Inputs        0.827         0.515       0.635
Outputs       0.744         0.577       0.650
Conversions   0.778         0.392       0.521
Moves         0.690         0.319       0.436
-----
Overall Precision 0.760
Overall Recall   0.451
Overall F1       0.566
=====

```

Figure 17. The detailed results for ProKR obtained using ProPara evaluator

#### 5.4 Error Analysis

A General Error Analysis is discussed here. A per paragraph analysis of the test dataset results is attached in the Appendix D.

**Imperfectly formed sentences** There are examples in the dataset where a verb is not well defined and a sentence may be improperly framed. For example, “The air travels through your windpipe. Into your lungs.” Here, the second sentence, is an incomplete sentence and should ideally be part of the first sentence but because of the way the dataset was created, some of the imperfections have come through but causes error as the system cannot understand the sentence.

**Symbolic Interpretation of Questions** In the current system, each question is treated to be different but some of the questions such as “What formed?”, “What is formed?”, “What has been formed?” should be treated to be similar question but aren’t treated as such.

**Coreference Resolution** The dataset has multiple instances which require simple to complex coreference resolution of phrases or words. For example, when an entity such as air masses gets converted to wind in paragraph 582 of propara: the system does not detect the similarity between the entities “air masses” and “the masses”, where, when referring to “air masses” as the masses, it cannot resolve “the masses” to mean “air masses” leading to incorrect predictions. Improvements in coreference resolution could possibly improve performance with paragraphs having imperfectly formed sentences as well such as the example: “Trash is removed from everything else. Goes to a landfill”.

**QA-SRL related errors** While QA-SRL is a great parser, it is not without its faults. There are multiple times where it cannot coreference resolve details within a sentence. It also replaces item names or person names with “someone” or “something” which can sometimes lead to losing information about a participant for the dataset.

**Impact of the locations model** Though, the BERT model that was used to learn the locations annotations was the best among the models used, one can speculate if the model caused any errors in the locations predicted since the bert model has an accuracy of 88.25 % (accuracy mentioned in previous chapter). The ProPara evaluator seemed to be immune to minor changes in location text. A major analysis of the impact of this model on the results remains pending during the time of submission of this work.

Until now, we have discussed the translation and, learning and the reasoning

process. This chapter then discussed the results and some error analysis. As part of the next chapter, we make an attempt to improve over one of the issues discussed here in the error analysis.

### ADDITION OF KNOWLEDGE

As discussed at the end of the last chapter, after generation of the learned ILP rules, it was necessary to be able to generalize over the rules, to get over issues caused by some questions that were similar. Based on the analysis of the results, generalization would have a slightly improved performance on the test side since there were some rules that were expected to trigger that just did not. The rest of the chapter goes into the effort of how the generalization of the rules is achieved.

#### 6.1 The Propbank Dataset

(Marcus, Santorini, and Marcinkiewicz 1993) originally introduced the Penn Treebank corpus, which consisted of close to 4.5 million words and multiple sentences that have been tagged by part of speech and the syntax trees of the sentences. The Proposition bank (Palmer, Gildea, and Kingsbury 2005) adds an additional layer of annotation to a portion of the Penn Treebank project with predicate-argument relations consisting of semantic roles of the words. Semlink (Palmer, Bonial, and McCarthy 2014) is a project which combined FrameNet, VerbNet and OntoNote sense groupings. There are multiple verbs in the Propbank dataset and each verb has a xml file associated with it which includes multiple roles of the verb used and example sentences which use the verb in that role. An example xml snippet is shown below in Figure 18

```

<frameset>
  <predicate lemma="absorb">
    <roleset id="absorb.01" name="suck up">
      <aliases>
        <alias framenet="" pos="n" verbnet="">absorption</alias>
        <alias framenet="" pos="v" verbnet="">absorb</alias>
      </aliases>
      <note>ABSORB-V NOTES: Frames file for 'absorb' based on sentences in financial subcorpus. No Verbnet entry. No comparison. (from absorb.01-v predicate notes)</note>
      <note>ABSORPTION-N NOTES: Roleset based on verb entry absorb.01. Framed by Anwen, Feb 2011. (from absorption.01-n)</note>
      <roles>
        <role descr="absorber, cause or agent" f="pag" n="0"/>
        <role descr="absorbed, us. liquid" f="ppt" n="1"/>
        <role descr="sponge, arg 1 is absorbed into this, when separate from cause" f="gol" n="2"/>
        <role descr="Source of liquid, absorbed from what?" f="dix" n="3"/>
      </roles>
      <example name="absorb losses" src="absorb losses" type="absorb losses">
        <inflection aspect="ns" form="infinitive" person="ns" tense="ns" voice="ns"/>
        <text>[Allstate]-1 is expected *trace*-1 to absorb another big hit in the fourth quarter as claims pour in from the San Francisco earthquake.</text>
        <arg f="" n="0">*trace*-1</arg>
        <rel f="">absorb</rel>
        <arg f="" n="1">another big hit</arg>
        <arg f="tmp" n="m">in the fourth quarter</arg>
        <arg f="tmp" n="m">as claims pour in from the San Francisco earthquake</arg>
      </example>
      <example name="Descriptive passive, with arg 2" src="Descriptive, with arg 2" type="Descriptive, with arg 2">
        <inflection aspect="ns" form="ns" person="ns" tense="ns" voice="ns"/>
        <text>George W. Bush, still by his own account given to "heavy drinking," absorbed *none* in changing the name of his

```

Figure 18. A Snippet of the Absorb Frame XML File From Propbank

```

absorb

roles:
0: "absorber, cause or agent"
1: "absorbed, us. liquid"
2: "sponge, arg 1 is absorbed into this, when separate from cause"
3: "Source of liquid, absorbed from what?"

```

Figure 19. The roles mentioned in the absorb frame file

The roles that are extracted from the frame file of absorb are shown in Figure 19.

## 6.2 Sentence and Argument Extraction from Propbank

Two kinds of information were extracted for usage from the propbank frame xml files. The below information can later be used to obtain a generalization over the ILP generated rules from before.

### 6.2.1 Sentences

All sentences from all the verb frame files were extracted and cleaned of the Treebank annotations to be used later. The number of sentences extracted were 22,443. An example sentence from the absorb frame file is “The habit is very bad because salt absorbs water from the meat.”

### 6.2.2 Arguments and Roles

The arguments and semantic roles were extracted from the frame files to later be used. All the extracted verbs, arguments and roles were represented in tuples as (verb, phrase, semantic role). The phrases and respective roles extracted for the running example are shown below in Figure 20. The corresponding tuples would then be (“absorb”, “salt”, “absorber, cause or agent”), (“absorb”, “water”, “absorbed, us. liquid”) and (“absorb”, “from the meat”, “Source of liquid, absorbed from what?”).

"absorber, cause or agent"	"salt"
"absorbed, us. liquid"	"water"
"Source of liquid, absorbed from what?"	"from the meat"

Figure 20. The roles and the corresponding phrases extracted for “The habit is very bad because salt absorbs water from the meat.”

## 6.3 QA-SRL Parsing of Extracted Sentences

The extracted sentences as part of Section 6.2.1 were then run through the QA-SRL parser (FitzGerald et al. 2018) (details of the parser were covered in Section 4.1). For the running example, the qasrl parse is shown below in Figure 21. The extracted

QA-SRL output was then represented as a tuple of (verb, question, answer). The tuples for the example would accordingly be: (“absorb”, “what absorbs something?”, “salt”), (“absorb”, “what does something absorb?”, “water”), (“absorb”, “where does something absorb something from?”, “from the meat”) and (“absorb”, “what does something absorb something from?”, “the meat”).

"What absorbs something?"	"salt"
"What does something absorb?"	"water"
"Where does something absorb something from?"	"from the meat"
"What does something absorb something from?"	"the meat"

Figure 21. The QASRL parse of “The habit is very bad because salt absorbs water from the meat.”

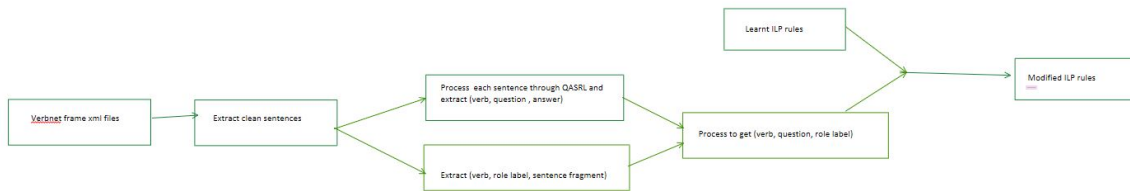


Figure 22. The Test Side Changes

#### 6.4 Obtaining Role Tagging of Questions

The argument phrase tuples of Section 6.2.2 and the questions-answer tuples of Section 6.3 were combined to form new tuples of (verb, question, semantic role) to enable us to now tag the questions with specific semantic roles. The overall changes of the process is shown above in Figure 22. The new tuples formed for the running example by combining the tuples in Figure 20 and Figure 21 are (“absorb”, “what v something?”, “absorber, cause or agent”), (“absorb”, “what does something v?”,



“absorbed, us. Liquid”) and (“absorb”, “where does something v something from?”, “Source of liquid, absorbed from what?”). To have an approximate match between the argument texts and the QA-SRL answers while combining the QA-SRL tuples and the propbank tuples, the fuzzywuzzy library’s fuzz.ratio function is used. A 75% or above match for the ratio was allowed for matching between the questions. The questions were then lemmatized using the StanfordCore NLP parser and Figure 23 shows a sample of the facts generated. The number of facts created is 25662.

```
propFact("absorb","what is being v ?","absorbed, us. liquid").
propFact("absorb","when is something being v ?","tmp").
propFact("absorb","why is something being v ?","tmp").
propFact("absorb","what v something ?","absorber, cause or agent").
propFact("absorb","what does something v ?","absorbed, us. liquid").
propFact("absorb","where does something v something from ?","source of liquid, absorbed from what?").
propFact("absorb","what does something v something from ?","source of liquid, absorbed from what?").
propFact("absorb","what is v ?","absorbed, us. liquid").
propFact("absorb","what is something v by ?","absorber, cause or agent").
propFact("abstain","who v ?","entity abstaining").
propFact("abstract","where did someone v something ?","adv").
propFact("abstract","who v something ?","entity abstracting").
```

Figure 23. Sample of PropBank based facts

The role tagged question tuples were then used to modify the ILP generated rules that were generated in Section 5.1. The rules that had questions that can be replaced were either replaced by the PropBank roles or appended to the original rules and the results were re-evaluated.

Question	Avg. Precision	Avg. Recall	Avg. F1
Inputs	0.904	0.502	0.646
Outputs	0.735	0.475	0.577
Conversions	0.810	0.286	0.423
Moves	0.730	0.279	0.404
Overall Precision	0.795		
Overall Recall	0.385		
Overall F1	0.519		

Figure 24. Results obtained on modifying the learned ILP rules with 90% similarity replacement of questions

In Figure 24, the results for the following process are presented: the PropFacts from above are used to replace the rules based on a 90% similarity between the question in the rules and the questions present in the ProPosition Bank facts and a matching of the verb being used. The rule is modified such that the question is replaced with the corresponding role in the ProPosition fact. Figure 25 represents the same process above but for an 85% similarity of questions. For the test time paragraph, new predicates are generated in run-time using ASP such that  $\text{observedAt}(V,Q,A,T):-\text{observedAt}(V,R,A,T)$  where R is the corresponding role to replace (Questions are matched exactly and not approximately here) when V and Q are present in the ProPosition facts.

```

=====
Question      Avg. Precision  Avg. Recall  Avg. F1
-----
Inputs          0.918         0.497       0.645
Outputs         0.673         0.401       0.503
Conversions     0.827         0.248       0.382
Moves           0.733         0.270       0.395
-----
Overall Precision 0.788
Overall Recall   0.354
Overall F1       0.488
=====

```

Figure 25. Results obtained on modifying the learned ILP rules with 85% similarity replacement of questions

## 6.5 Results and Analysis

When comparing the results in Figure 24 compared to the original results of Chapter 5 present above in Figure 26, we see that the precision increases from 0.756

```

=====
Question      Avg. Precision  Avg. Recall  Avg. F1
-----
Inputs        0.827          0.515        0.635
Outputs       0.744          0.577        0.650
Conversions   0.778          0.392        0.521
Moves         0.690          0.319        0.436
-----
Overall Precision 0.760
Overall Recall   0.451
Overall F1      0.566
=====

```

Figure 26. Original Result without adding Propbank Knowledge

to 0.795 while the recall decreases from 0.451 to 0.385. The F1 decreases from 0.566 to 0.519. The decrease in the F1 is possibly because of the multiple rule changes that occur leading to the formation of more false positives and false negatives. An increase in precision would imply that false positives decreased or true positives increase or both. A decrease in recall implies that false negatives have increased for sure. Thus, it is highly likely that the decreased F1 is more due to an increasing occurrence of false negatives than an increasing occurrence of false positives. The results obtained on using

```

=====
Question      Avg. Precision  Avg. Recall  Avg. F1
-----
Inputs        0.827          0.520        0.639
Outputs       0.744          0.564        0.642
Conversions   0.781          0.395        0.525
Moves         0.674          0.321        0.435
-----
Overall Precision 0.756
Overall Recall   0.450
Overall F1      0.564
=====

```

Figure 27. Results obtained on appending to the learned ILP rules with 90% similarity replacement

Meanwhile, instead of modifying the learned rules , if the rules are appended to,

the results do not change a lot because very few paragraph predictions are significantly modified (paragraph numbers 37 and 99) while others only change very little (249, 502, 533, 660, 725, 927). The precision, recall and F1 are slightly worse than the original results and are shown above in Figure 27. Similarly, the results obtained when an 85% similarity of questions is used is present below in Figure 28. We can see that the 85% question similarity's results lie in between the 90% similarity and the original results from Chapter 5.

```

=====
Question      Avg. Precision  Avg. Recall  Avg. F1
-----
Inputs        0.827          0.520       0.639
Outputs       0.706          0.555       0.621
Conversions   0.781          0.395       0.525
Moves         0.672          0.328       0.441
-----
Overall Precision 0.747
Overall Recall   0.450
Overall F1       0.561
=====

```

Figure 28. Results obtained on appending the learned ILP rules with 85% similarity replacement of questions

**Why use fuzzywuzzy?** One might argue the validity of using the fuzzywuzzy library for string matching. The fuzzywuzzy library bases the ratio calculation off of the Levenshtein distance used in Information Theory which is calculated by the algorithm of the same name. The Levenshtein distance is a string edit distance metric whose algorithm was presented in (Levenshtein 1966) and is a well known and well cited paper.

**Why adding knowledge didn't work well** It was expected that the addition of knowledge should have improved the results.

Finally, The cleaned Propbank dataset, the QA-SRL resulting tuples , the

probank tuples and the final set of tuples are available for download at <https://github.com/aorghob/ThesisMaterial> .

### CONCLUSION AND FUTURE DIRECTION

Chapter 4 set up the translation of the ProPara dataset to make it usable in the method briefly explained in Chapter 5. A brief summary of the work is present in Section 7.1 while Section 7.2 discusses the Results obtained. Finally, Section 7.3 discusses possible future directions.

#### 7.1 Summary

Artificial Intelligence, in the last two decades has become a rapidly developing field, especially due to the availability of powerful hardware and the consequent rise of popularity in Neural Network Architectures. Multiple innovations in Neural Architectures has led to rising interest within the Machine Learning, Computer Vision and Natural Language Processing Communities in the past decade. The AllenAI Institute of Artificial Intelligence has come up with multiple datasets. The dataset, ProPara, targeted to improve reading comprehension was introduced in the beginning of the thesis. The details of the method used are discussed in Chapter 5. A test-side modification of the rules obtained from Inductive Logic Programming were discussed in Chapter 6 in an attempt to add knowledge to improve performance but unexpectedly, did not lead to better results.

## 7.2 Results and Discussion

The results obtained from the initial translation and use of Inductive Logic Programming and ASP are mentioned in Table 4. On Analysis, it was found that coreference resolution, formation of rules with similar questions but different high level rule assignments and some inference errors cause errors with a per-paragraph detail presented in Appendix D.

Table 4. Results of Various Systems on the ProPara Dataset

	Precision	Recall	F1
ProLocal	77.4	22.9	35.3
QRN	55.5	31.3	40.0
EntNet	50.2	33.5	40.2
ProGlobal	46.7	52.4	49.4
ProStruct	74.2	42.1	53.7
KG-MRC	64.52	50.68	56.77
<b>ProKR (This work)</b>	<b>76.00</b>	<b>45.10</b>	<b>56.60</b>
NCET (later)	67.10	58.50	62.50
LACE (later)	75.30	45.40	56.60

The original ILP generated rules with the reasoning system gives an accuracy of 56.60% The modification of the ILP generated rules gives a maximum accuracy of 56.40 % as discussed in the previous chapter. Thus, we take the accuracy to be considered as 56.60 %. It can be seen from Table 4 that the model performs on-par or better than other deep learning based models, reasoning and ILP based approaches can yield great results as well.

A criticism of ILP based approaches would generally be that the high-level rules and inertia and background have to be defined manually before proceeding. The flip side is that more specific rules can be defined.

### 7.3 Future Directions

The first step to improvement that needs to be considered is to overcome coreference resolution related errors that were faced as mentioned at the end of Chapter 5. This can be done in adding additional inertia or domain based ASP rules to be able to handle the resolution.

Another improvement that can be tried is to be able to fuzzily compare questions in the clingo side predicate that was added in Chapter 6.

Frame Element	Core Type
Degree	Peripheral
Duration	Peripheral
Item	Core
Manner	Peripheral
Place	Peripheral
Substance	Core
Time	Peripheral

[Turn Colors Off](#)

- moisture-(1)
  1. Pointing the mortar joints had to be done long after laying the last blocks of the day , as **the concrete** **ABSORBS** **very little moisture** .
  2. Store the envelopes in an airtight container , and include **a sachet of silica gel** to **ABSORB** **moisture** ( if the gel turns pink , dry gently in the oven until it turns blue again ) .

Figure 29. A Framenet Annotation for the word absorb

Also, in Chapter 6, PropBank was used to add role based knowledge to the dataset. FrameNet can also be used to add knowledge as it is also a role based knowledge of more than 200,000 sentences linked with more than 1200 semantic frames. Each frame can be used to describe a type of event, relation or entity in a sentence. The words that evoke the frame are known as lexical units. For example, in Figure 29 (Image sourced as a screenshot from FrameNet’s website at (*framenet2.icsi.berkeley.edu*)), the absorb frame file is shown from framenet’s website and the frame elements are the multiple roles that can be considered. Two sentences are annotated with the roles highlighted by color and these sentences are not present in the PropBank absorb frame file.



Another step to confirm the generalizability of the work is to try it out on similar process or procedure based datasets such as the Recipes dataset (Kiddon et al. 2015).

Meanwhile, two other approaches came to mind while working on this dataset. One approach that stands out due to the presence of actions and state changes is that of Reinforcement Learning where actions need to be learnt. A Deep Reinforcement Learning approach which incorporates word encodings and constraints of the problem would probably work.

The other approach comes from the recent state of the art results with the Transformer and the BERT architectures. BERT has achieved better results than BiLSTM-CRF, as experimented in (Peters et al. 2017) at Parts of Speech Tagging, in the CoNLL 2013 dataset, which can be considered a sequence labelling problem. One of the reasons that BERT performs so well is possibly because the model is trained on a huge dataset. On similar lines, XL-NET (Yang et al. 2019) was a recent state of the art language model released by Google Brain and is built on top of the TransformerXL (Dai et al. 2019) architecture. It is considered to be an improvement over BERT. Combining BERT or XL-NET (Yang et al. 2019) or a bidirectional Transformer approach with natural language abduction and some commonsense knowledge and also incorporating coreference resolution will probably lead to a good solution for the ProPara dataset. BERT and coreference resolution would have to be trained in a multitask manner or perhaps learn coreference resolution as an auxiliary task (Ruder 2017). Until human accuracies are beaten for NLP datasets (83.9% for ProPara), research into models trying to beat these datasets will continue since the Artificial General Intelligence will want to be able to replicate human performance on multiple tasks before integrating everything together to help humanity.

## REFERENCES

- Mission*. <https://allenai.org/about.html>.
- Banarescu, Laura, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. “Abstract meaning representation for sembanking.” In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, 178–186.
- Bhattacharjee, Aurgho. 2018. *aurghob/nrl-qasrl*, October. <https://github.com/aurghob/nrl-qasrl/tree/propara>.
- Chen, Qian, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. “Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference.” *CoRR* abs/1609.06038. arXiv: 1609.06038. <http://arxiv.org/abs/1609.06038>.
- Clark, Peter, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. “Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge.” *CoRR* abs/1803.05457. arXiv: 1803.05457. <http://arxiv.org/abs/1803.05457>.
- Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context.” *CoRR* abs/1901.02860. arXiv: 1901.02860. <http://arxiv.org/abs/1901.02860>.
- Das, Rajarshi, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018. “Building Dynamic Knowledge Graphs from Text using Machine Reading Comprehension.” *arXiv preprint arXiv:1810.05682*.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” *CoRR* abs/1810.04805. arXiv: 1810.04805. <http://arxiv.org/abs/1810.04805>.
- Du, Xinya, Bhavana Dalvi, Niket Tandon, Antoine Bosselut, Wen-tau Yih, Peter Clark, and Claire Cardie. 2019. “Be Consistent! Improving Procedural Text Comprehension using Label Consistency.” In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2347–2356. Minneapolis, Minnesota: Association for Computational Linguistics, June. <https://www.aclweb.org/anthology/N19-1244>.

- FitzGerald, Nicholas, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. “Large-scale qa-srl parsing.” *arXiv preprint arXiv:1805.05377*.
- framenet2.icsi.berkeley.edu*. <https://framenet2.icsi.berkeley.edu/fnReports/data/lu/lu10687.xml?mode=annotation>.
- Gebser, Martin, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. 2014. “Clingo = ASP + Control: Preliminary Report.” *CoRR* abs/1405.3694.
- Gelfond, Michael, and Vladimir Lifschitz. 1988. “The Stable Model Semantics For Logic Programming,” 1070–1080. MIT Press.
- Gupta, Aditya, and Greg Durrett. 2019. “Tracking Discrete and Continuous Entity State for Process Understanding.” *arXiv preprint arXiv:1904.03518*.
- Henaff, Mikael, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2016. “Tracking the World State with Recurrent Entity Networks.” *CoRR* abs/1612.03969. arXiv: 1612.03969. <http://arxiv.org/abs/1612.03969>.
- Kembhavi, Aniruddha, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. “Are You Smarter Than A Sixth Grader? Textbook Question Answering for Multimodal Machine Comprehension.” In *Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Khot, Tushar, Ashish Sabharwal, and Peter Clark. 2018. “Scitail: A textual entailment dataset from science question answering.” In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Kiddon, Chloé, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. “Mise en place: Unsupervised interpretation of instructional recipes.” In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 982–992.
- Kociský, Tomáš, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. “The NarrativeQA Reading Comprehension Challenge.” *CoRR* abs/1712.07040. arXiv: 1712.07040. <http://arxiv.org/abs/1712.07040>.
- Lee, Joohyung. *Lecture 14*.
- Levenshtein, Vladimir I. 1966. “Binary codes capable of correcting deletions, insertions, and reversals.” In *Soviet physics doklady*, 10:707–710. 8.

- Lifschitz, Vladimir. 2008. “What Is Answer Set Programming?” In *AAAI*, 8:1594–1597. 2008.
- Manning, Christopher, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. “The Stanford CoreNLP natural language processing toolkit.” In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60.
- Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. “Building a large annotated corpus of English: The Penn Treebank.”
- Mihaylov, Todor, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. “Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering.” In *EMNLP*.
- Mishra, Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. “Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension.” *arXiv preprint arXiv:1805.06975*.
- Mitra, Arindam, and Chitta Baral. 2018. “Incremental and Iterative Learning of Answer Set Programs from Mutually Distinct Examples.” *CoRR* abs/1802.07966. arXiv: 1802.07966. <http://arxiv.org/abs/1802.07966>.
- Nguyen, Tri, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. “MS MARCO: A Human Generated MACHine Reading COMprehension Dataset.” *CoRR* abs/1611.09268. arXiv: 1611.09268. <http://arxiv.org/abs/1611.09268>.
- Palmer, Martha, Claire Bonial, and Diana McCarthy. 2014. “Semlink+: Framenet, verbnet and event ontologies.” In *Proceedings of Frame Semantics in NLP: A Workshop in Honor of Chuck Fillmore (1929-2014)*, 13–17.
- Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. “The proposition bank: An annotated corpus of semantic roles.” *Computational linguistics* 31 (1): 71–106.
- Palmer, Martha, Daniel Gildea, and Nianwen Xue. 2010. “Semantic Role Labeling.” *Synthesis Lectures on Human Language Technologies* 3 (1): 1–103. doi:10.2200/S00239ED1V01Y200912HLT006. eprint: <https://doi.org/10.2200/S00239ED1V01Y200912HLT006>.
- Parikh, Ankur P., Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. “A Decomposable Attention Model for Natural Language Inference.” In *EMNLP*.

- Peters, Matthew E., Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. “Semi-supervised sequence tagging with bidirectional language models.” *CoRR* abs/1705.00108. arXiv: 1705.00108. <http://arxiv.org/abs/1705.00108>.
- Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. “Deep contextualized word representations.” *CoRR* abs/1802.05365. arXiv: 1802.05365. <http://arxiv.org/abs/1802.05365>.
- ProPara Dataset*. <http://data.allenai.org/propara/>.
- Rajpurkar, Pranav, Robin Jia, and Percy Liang. 2018. “Know What You Don’t Know: Unanswerable Questions for SQuAD.” *CoRR* abs/1806.03822. arXiv: 1806.03822. <http://arxiv.org/abs/1806.03822>.
- Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. “SQuAD: 100, 000+ Questions for Machine Comprehension of Text.” *CoRR* abs/1606.05250. arXiv: 1606.05250. <http://arxiv.org/abs/1606.05250>.
- Ray, Oliver. 2009. “Nonmonotonic abductive inductive learning.” *Journal of Applied Logic* 7 (3): 329–340.
- Ribeiro, Danilo, Thomas Hinrichs, Maxwell Crouse, Kenneth Forbus, Maria Chang, and Michael Witbrock. 2019. “Predicting State Changes in Procedural Text using Analogical Question Answering.” *Advances in Cognitive Systems* (August). doi:[http://www.qrg.northwestern.edu/papers/Files/QRG\\_Dist\\_Files/QRG\\_2019/RibeiroACS2019.pdf](http://www.qrg.northwestern.edu/papers/Files/QRG_Dist_Files/QRG_2019/RibeiroACS2019.pdf).
- Ruder, Sebastian. 2017. “An Overview of Multi-Task Learning in Deep Neural Networks.” *CoRR* abs/1706.05098. arXiv: 1706.05098. <http://arxiv.org/abs/1706.05098>.
- Russell, Stuart, and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press.
- Seo, Min Joon, Sewon Min, Ali Farhadi, and Hannaneh Hajishirzi. 2016. “Query-Reduction Networks for Question Answering.” In *ICLR*.
- Tandon, Niket, Bhavana Dalvi Mishra, Joel Grus, Wen-tau Yih, Antoine Bosselut, and Peter Clark. 2018. “Reasoning about actions and state changes by injecting commonsense knowledge.” *arXiv preprint arXiv:1808.10012*.

- Trischler, Adam, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. “NewsQA: A Machine Comprehension Dataset.” *CoRR* abs/1611.09830. arXiv: 1611.09830. <http://arxiv.org/abs/1611.09830>.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. “Attention Is All You Need.” *CoRR* abs/1706.03762. arXiv: 1706.03762. <http://arxiv.org/abs/1706.03762>.
- Weston, Jason, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. 2015. “Towards ai-complete question answering: A set of prerequisite toy tasks.” *arXiv preprint arXiv:1502.05698*.
- Natural-language understanding*. 2019, April. [https://en.wikipedia.org/wiki/Natural-language\\_understanding](https://en.wikipedia.org/wiki/Natural-language_understanding).
- Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. “XLNet: Generalized Autoregressive Pretraining for Language Understanding.” *CoRR* abs/1906.08237. arXiv: 1906.08237. <http://arxiv.org/abs/1906.08237>.

## APPENDIX A

### SAMPLE CREATE RULES LEARNED BY ILP ON PROPORA

create(V1, V2):-eobservedAt("provide", "what does something v ?", V1, V2),  
entity(V1), time(V2).

create(V1, V2):-eobservedAt("turn", "what v something ?", V1, V2), doesNotexists(V1,  
V2), entity(V1), time(V2).

create(V1, V2):-eobservedAt("support", "what is v ?", V1, V2), eobservedAt("be",  
"what v something ?", V1, V2), entity(V1), time(V2).

create(V1, V2):-entityObservation("incubate", "what v ?", "the eggs", V2), does-  
Notexists(V1, V2), entity(V1), time(V2).

create(V1, V2):-entityObservation("filter", "what does something v ?", "the blood",  
V2), doesNotexists(V1, V2), description(V1, "waste"), entity(V1), time(V2).

create(V1, V2):-eobservedAt("gain", "what does something v ?", V1, V2), does-  
Notexists(V1, V2), entity(V1), time(V2).

create(V1, V2):-eobservedAt("rise", "what v ?", V1, V2), doesNotexists(V1, V2),  
entity(V1), time(V2).

create(V1, V2):-eobservedAt("go", "how does something v somewhere ?", V1, V2),  
entity(V1), time(V2).

create(V1, V2):-eobservedAt("consider", "what is something v ?", V1, V2), doesNotex-  
ists(V1, V2), entity(V1), time(V2).

create(V1, V2):-eobservedAt("be", "what does something v ?", V1, V2), does-  
Notexists(V1, V2), description(V1, "oil"), entity(V1), time(V2).

create(V1, V2):-eobservedAt("drip", "how is something v somewhere ?", V1, V2),  
entity(V1), time(V2).

create(V1, V2):-eobservedAt("break", "what v ?", V1, V2), doesNotexists(V1,  
V2), entity(V1), time(V2).

create(V1, V2):-eobservedAt("break", "how does something v ?", V1, V2), does-  
Notexists(V1, V2), entity(V1), time(V2).

create(V1, V2):-doesNotexists(V1, V2), eobservedAt("release", "what is being v  
?", V1, V2), entity(V1), time(V2).



create(V1, V2):-eobservedAt("develop", "what v with something ?", V1, V2), entity(V1), time(V2).

create(V1, V2):-eobservedAt("have", "what does something v ?", V1, V2), doesNotexists(V1, V2), entity(V1), time(V2).

create(V1, V2):-groundObservation("water", "what does something v to ?", "evaporation", V2), description(V1, "water vapor"), entity(V1), time(V2).

create(V1, V2):-eobservedAt("copy", "where is something v ?", V1, V2), entity(V1), time(V2).

create(V1, V2):-groundObservation("grow", "what is being v into ?", "new trees", V2), doesNotexists(V1, V2), entity(V1), time(V2).

create(V1, V2):-doesNotexists(V1, V2), description(V1, "Seeds"), entity(V1), time(V2).

create(V1, V2):-description(V1, "thicker coat"), eobservedAt("grow", "what does something v ?", V1, V2), entity(V1), time(V2).

create(V1, V2):-eobservedAt("compose", "how is something v ?", V1, V2), entity(V1), time(V2).

## APPENDIX B

### SAMPLE DESTROY RULES LEARNED BY ILP ON PROPORA

mayDestroy(V1, V2):-observedAt(“form”, “what v somewhere ?”, “the butterfly”, V2), description(V1, “pupa”), entity(V1), time(V2).

mayDestroy(V1, V2):-observedAt(“hatch”, “what v ?”, “the cocoon”, V2), description(V1, “pupa”), entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation(“complete”, “what does something v ?”, “its metamorphosis”, V2), description(V1, “pupa”), entity(V1), time(V2).

mayDestroy(V1, V2):-somethingGotCreated(V2), observedAt(“form”, “what is v ?”, “crystals”, V2), description(V1, “magma”), entity(V1), time(V2).

mayDestroy(V1, V2):-somethingGotCreated(V2), observedAt(“flow”, “what v from something ?”, “magma”, V2), description(V1, “magma”), observedAt(“flow”, “how does something v from something ?”, “in the form of lava”, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation(“go”, “what v into something ?”, “some water”, V2), description(V1, “rain”), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt(“dissolve”, “what does something v ?”, V1, V2), groundObservation(“dissolve”, “what does something v ?”, “calcite”, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt(“energize”, “what is v ?”, V1, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt(“die”, “who v ?”, V1, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt(“convert”, “what does something v ?”, V1, V2), somethingGotCreated(V2), entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation(“build”, “where does something v up ?”, “behind the dam”, V2), description(V1, “rain”), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt(“enter”, “what v something ?”, V1, V2), description(V1, “sugars”), entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation(“enter”, “what v something ?”, “the larva”, V2), eobservedAt(“enter”, “what v something ?”, V1, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt(“break”, “what is v ?”, V1, V2), description(V1, “coal”), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt("combine", "what is v with something ?", V1, V2),  
entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation("incorporate", "what is v into something ?",  
"the recycled materials", V2), entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation("become", "what is being v ?", "urine",  
V2), description(V1, "water"), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt("give", "what v something ?", V1, V2), eob-  
servedAt("cause", "what does something v ?", V1, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-description(V1, "metamorphic rock"), entity(V1), time(V2).

mayDestroy(V1, V2):-groundObservation("break", "what is v ?", "the food", V2),  
eobservedAt("break", "what is v ?", V1, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt("die", "what v ?", V1, V2), entity(V1), time(V2).

mayDestroy(V1, V2):-eobservedAt("split", "what does something v ?", V1, V2),  
entity(V1), time(V2).

## APPENDIX C

### SAMPLE LOCATION RULES LEARNED BY ILP ON PROPORA

initiate(locationOf(V1, V2), V3):-eobservedAt(“replace”, “what is being v ?”, V1, V3), lobservedAt(“replace”, “what is something being v on ?”, V2, V3), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-groundObservation(“lit”, “what is v ?”, “the oil”, V3), value(V2, “stove”), description(V1, “warm air”), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-eobservedAt(“put”, “what is v ?”, V1, V3), lobservedAt(“put”, “what is something v in ?”, V2, V3), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-eobservedAt(“call”, “what is something v ?”, V1, V3), lobservedAt(“make”, “what v something ?”, V2, V3), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt(“combine”, “what v with something ?”, V2, V3), description(V1, “sulfur”), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt(“react”, “what does something v with ?”, V2, V3), eobservedAt(“react”, “what does something v with ?”, V1, V3), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt(“move”, “what does something v from ?”, V2, V3), eobservedAt(“move”, “what v ?”, V1, V3), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-value(V2, “cloud”), eobservedAt(“call”, “what is something v ?”, V1, V3), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt(“bump”, “what v something ?”, V2, V3), description(V1, “ice”), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-eobservedAt(“know”, “what is v as something ?”, V1, V3), value(V2, “bottom of ocean , riverbed or swamp”), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-eobservedAt(“rise”, “what v ?”, V1, V3), value(V2, “bloodstream”), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt(“burn”, “what v somewhere ?”, V2, V3), description(V1, “hydrogen”), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-value(V2, “air”), eobservedAt(“release”, “what

is v ?", V1, V3), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt("be", "where does something v ?", V2, V3), eobservedAt("be", "what v somewhere ?", V1, V3), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-description(V1, "water vapors"), lobservedAt("form", "how is something being v ?", V2, V3), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt("form", "what v with something ?", V2, V3), eobservedAt("translate", "where is something v ?", V1, V3), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-lobservedAt("hide", "where does something v something ?", V2, V3), description(V1, "food stores"), entity(V1), location(V2), time(V3).

initiate(locationOf(V1, V2), V3):-lobservedAt("start", "what v ?", V2, V3), description(V1, "forest fire"), entity(V1), location(V2), time(V3).

before(locationOf(V1, V2), V3):-lobservedAt("push", "what v into something ?", V2, V3), eobservedAt("push", "what v into something ?", V1, V3), entity(V1), location(V2), time(V3).

terminate(locationOf(V1), V2):-groundObservation("start", "what does something v doing ?", "growing", V2), entity(V1), time(V2).

terminate(locationOf(V1), V2):-eobservedAt("make", "what v something ?", V1, V2), groundObservation("call", "what is something v ?", "honey", V2), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("put", "when is something being v ?", "when you are finished", V2), description(V1, "food"), entity(V1), time(V2).

terminate(locationOf(V1), V2):-eobservedAt("exchange", "what is v ?", V1, V2), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("put", "where is something being v ?", "away", V2), eobservedAt("put", "what is being v ?", V1, V2), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("split", "what is v ?", "the

atoms", V2), description(V1, "neutron"), entity(V1), time(V2).

terminate(locationOf(V1), V2):-eobservedAt("surprise", "who v someone ?", V1, V2), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("reuse", "what v something ?", "the plant", V2), description(V1, "oxygen"), entity(V1), time(V2).

terminate(locationOf(V1), V2):-eobservedAt("remove", "what is being v ?", V1, V2), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("launch", "what is being v ?", "rocket", V2), description(V1, "satellite"), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("ignite", "where is something v ?", "in the combustion chamber", V2), description(V1, "heating oil"), entity(V1), time(V2).

terminate(locationOf(V1), V2):-groundObservation("start", "what does something v doing ?", "growing", V2), entity(V1), time(V2).



APPENDIX D

ANALYSIS OF RESULTS ON THE TEST SET OF PROPARA

qasrl errors: 7,  
includes incorrect question-answer pair: 1146  
answers not containing entity/location : 1 - ex: 903  
no rule present with verb:10 - ex: 460, 1188, 70  
coreference resolution: 16 - also, topic needs to be included for coreference resolution  
(ex: 400)  
more inference/commonsense needed: 7 - 401, 896, 927, 1031, 1190  
maydestroy but not destroyed:6 - ex: 1146  
terminate but not destroyed: 1  
creation verb not present: 1 - ex: 410  
generalization may help - 3 - ex: 465  
initate but not create rule - 1 - ex: 660  
destroy rule not present for verb - 2 - ex : 1146  
destroy rule not triggerred - 1 - ex: 661  
create rule not triggerred - 2 - ex : 695, 697  
create rule not present for verb - 2 - ex: 1031, 38  
annotation problem - 1 - ex: 1145  
no phrase based rules present - 1 - ex : 1147  
verb not learnt properly - 1 - ex: 67

interesting test to look at : 697, 791, 896, 1032, 1033

37 - qasrl, input/output; plant/animal null and match; soft tissues destroy matched;  
bones: missed destroy in step 5; fossil: got create in step 6;

38 - missed 2 inputs; missed 1 output; both conversions missed and got something else;  
animal/body: missed destroy in step 6 because form has maydestroy but destroy not triggerred;  
soil: missed destroy in step 5 because assumption is that rocks develop from the soil and got extra create in step 4 but seems fine;  
rock: missed create in step 5 because no create rule present for form; fossil: matched create in step 6;

67 - inputs missed-plants or dead animals, mistake in destroy and create, create for coal correct time but incorrect location, destroy and then create recognized,  
area is a wierd location, coreference resolution required for same area to be bottom of the swamps,  
join not learnt properly probably;

68 - water:location not found leading to incorrect input,destroy of coal not detected as coal replaced by something in qasrl - qasrl problem

69 - inputs are alright, pressure not detected as output but detected peat instead,  
coreference resolution should have detected destroy of peat, cause indicates location sometimes:alright to miss

70 - no inputs, outputs and creations or destroys detected here, rain destroy not detected due to no rule for stop, unsure what should be happening with water: location detected as water but should be stream or vanish because of evaporation: difficult. Why is precision for creation and destroy coming to 1?

99 - inputs are accurate, outputs: two correct but missed rain, create worked well

152 - input, output, create and destroy works fine

249 - got one input but missed "rocks OR smaller pieces", outputs: correct, destroyed: got one and missed one, creation fine. formation leading to destruction: slightly difficult

310 - inputs: missed carbon dioxide, outputs: fine, conversions: fine, destroy did not work for carbon dioxide - terminate not treated similar to destroy? Why did destroy not fire?

400 - inputs and outputs match, creation and destroy seems fine, problem with location. Dataset problem and coreference resolution here. Topic needs to be considered for coreference resolution.

401 - both inputs and outputs are incorrect here, alternating current could've faced problems because of unsanitized quotes, also need commonsense that some sunlight comes from sun shining.

409 - inputs and outputs are good. create and destroys are good too.

410 - inputs and outputs are both wrong. creating for one case should not have been there and destroy for other. dataset assumes when something is being talked about, the process is starting but ILP expects a creation related verb - light seems fine.

411 - inputs and outputs good, light destroy with incorrect timestamp but seems better in our case.

429 - inputs: got one and missed one, outputs: incorrect, possibly missed one due to not satisfaction of quotes, not able to distinguish between nitrogen and fixable or used nitrogen, destroy incorrect for plants and nitrogen.

460 - inputs: correct, outputs: incorrect, carbon predictions are wierd as there are multiple create and destroys, unable to detect creation of fossil fuel at right time - possibly because rule not present without verbs but requires one here - discovered

may be considered

463 - inputs, outputs and conversions all missed but why? Raindrops is usually missed. No create and destroy found.

465 - inputs, outputs and conversions missed again, did not properly learn that freeze can end and create something - generalization here probably can help. commonsense could help too.

502 - inputs and outputs appropriate, destroy works fine but create missed for one, piston motion creation missed.

503 - inputs are alright, more outputs generated than true values, spark-incorrect time created, locations incorrect.

533 - inputs, outputs and conversions don't match. creates being detected - missed one, but destroys being missed - why? sprout is a maydestroy but not detected as a destroy, not sure what rule detected for becoming. sprouts is wierdly a participant. Reaching maturity could be a problem - does not directly specify a creation. Besides, there seems to be incorrect coreference resolution happening.

534 - inputs, outputs dont match but conversions match. create majority being missed. destroy working. text incorrect here - so flowers missed. coreference resolution problem mostly. become and maydestroy again due to definition of become. QASRL made wrong coreference resolution.

582 - input don't match. outputs: got two out of three. conversions don't match. 1-2 creates out of 3 detected. destroys not tracked well - why?. Air masses does not have a creation verb - meet. no destruction verb present as well. wind creation went wrong. coreference resolution for masses, air masses, warm wind and cold wind not detected well.

583 - inputs match, outputs - 2 out of 3 match, conversions none and match. Creation assumed but not for warm humid air. updraft creation correct. tornado funnel creation missed.

600 - all inputs predicted and 1 extra. all 2 outputs missed. conversion missed. energy creation missed and incorrectly destroyed. creation can be accounted for here using a generalized rule...create(V1,V2):-eobservedAt("convert","what is something v into ?",V1,V2),entity(V1),time(V2). for observedAt("convert","what is something v to ?","to energy",1). understandable why hot gas creation is not detected.

653 - input wrong, output: 1 extra, other 3 right. conversions: one conversion captured other extra. seed destroy missed due to missed coreference resolution, fruit and flower tracked well.

654 - inputs match, outputs match, conversions: creation match while destroy dont, seedling multiple times created and destroyed, other entities tracked well.

659 - missed 1 out of 2 inputs, outputs: predicted one extra than none, missed multiple conversions, missed one nitrogen create, ammonium destroy and creation missed. missed because incorrect question asked by qasrl : observedAt(“turn”,“where does something v something ?”,“back into ammonium”,4).

Not sure about whether maydestroy to be triggered or not. similar for create.

660 - got all 3 inputs, missed 1 out of 2 outputs, missed 1 out of 5 conversions, plants destroy off by one step, wastes creation missed. Here, expel is probably missed because expel has initiate but no create rule. what is difference between create and initiate?

661 - got 1 out of 2 inputs, outputs none and matched, got 2 out of 4 conversions, only one destroy matches but 3 extra creates and destroys present, step 2 nitrates creation missed and step 5 nitrates destruction missed, for plant : one destroy is correct, otherwise: create destroy create destroy create is wrong, for nitrates destroy: no absorb destroy rule present, create rule was probably not triggered for nitrates - generalization may help here.

695 - inputs 1 and didn't match, outputs got 4 out of 5, got 1 out of 2 conversions. water: destroy missed. vapor: create missed. clouds: create missed by 1 step and extra destroy present, coreference resolution failed for water and did not recognize that water had turned leading to miss in destroy, vapor: create rule does not seem to have triggered but why? not sure: maybe wrong thing was looked into.

696 - inputs 1 and didn't match, outputs: missed all 2, conversions none and match. water droplets: missed create, rain: missed step 6 create and labelled destroy instead. water droplets creation missed due to qasrl answer miss, not sure why rain labelled as destroy even if create rule present.

697 - inputs none and match, outputs: got 1 out of 2, conversions none and match, rain: missed step 5 create because , not sure why rain is not being detected as created.

725 - inputs: missed all 4, outputs: got 1 out of 2, conversions missed the only 1

mentioned, plant: missed step 2 destroy, soft tissues: predicted 2 extra create and destroy, sediment: predicted create instead of destroy, rock: missed create, bones: predicted extra create and missed required destroy. why bones missed ? minerals: predicted destroy extra. stone replica: missed final step create and predicted in the beginning: why?

726 - inputs: got 1 out of 3, outputs: got 1 out of 2, conversions: got 1 out of 2. animal skeleton: has extra create and missed destroy in step 9, mudsand: have 2 extra creates and misplaced destroy by 1 step, hole: missed step 9 create

727 - inputs: got 2 out of 4, outputs: got 1 out of 2, conversions: missed the only 1, animal: destroy off by 2 steps, bones: have extra create but missed destroy, mudsilt - missed destroy in step 6, rock: missed create in step 6

791 - inputs: none and match, outputs: missed the only 1, conversions: missed the only 1, DC electricity: missed destroy in step 4 because maydestroy exists but destroy not triggered and speculate that coreference resolution did not work, AC electricity: missed create in step 4 because create rule does not seem to have triggered.

896 - inputs: missed the only 1, outputs: missed the only 1, conversions: missed the only 1, oxygen-depleted blood: missed destroy in step 7 because common-sense and coreference resolution failed and because no rules present for oxygenate, oxygenated blood: missed create in step 7 because of similar reasons

903 - inputs: missed the only 1, outputs: got 1 out of 2, conversions: missed the only 1, visible light: missed destroy in step 4 because sentence is incomplete leading to no coreference resolution possibility, image: missed create in step 5 because interpret does not have a rule and answers do not contain image though it seems like it should in some way, oxygenated blood create is fine.

904 - inputs: missed the only 1, outputs: got the only 1, conversions: missed all the 2, electrical impulses: create is fine, light: missed destroy in step 1 because refract does not have a rule and doesn't really destroy something, small image: missed create in step 1 because refract does not have a rule and destroy in step 3 because coreference resolution missed for small image, retinas: none and matches

927 - inputs: missed all 4, outputs: missed the only 1, conversions: missed the only 1, rain: missed destroy in step 2 because inference/commonsense is needed that rain becomes rainwater, rain water: none and matches, land: missed destroy in step 5 because qasrl question and answer slightly wrong - should be based on phrase instead of verb, plants: missed destroy in step 5 because question and answer slightly wrong, trees: missed destroy in step 5 because question and answer slightly wrong.

932 - inputs: none and matched, outputs: none and matched, conversions: none and matched, water: matches, rocks: none and matches, chemicals: none and matches, oxygen: none and matches

933 - inputs: none but predicted 1 instead, outputs: none and matched, conversions: none and matched, water: extra destroy present, carbon dioxide: none and matches, mineral based material: none and matched, rock: none and matched

1031 - inputs: got the only 1 but also predicted 1 extra, outputs: got 1 and matched, conversions: got 2 out of 4, plant or animal: got the one destroy, bones: create missed in step 1 because inference/abduction and backtracking from step 2 needed and destroy missed in step 3 because cover rule had maydestroy but was not triggered and seems fair, sediment: create missed in step 3 because cover does not have associated create rule but got destroy, rock: got create and destroy, fossil: got the one create

1032 - inputs: got 2 inputs and matched, outputs: missed the only 1, conversions: missed all 3, animals: destroy off by two steps, plants: off by two steps, bones and hard tissues: missed create in step 3 because some abduction is needed from next step regarding bones and hard tissues remaining and missed destroy in step 5 because inference or coreference resolution needed as it isn't directly indicated that bones and hard tissues become part of sediment, sediment: got create but missed destroy because harden has no rule, rock: missed create on step 5

1033 - inputs: predicted something else than the 1 actual input, output: got all 2 i.e. matched, conversions: got 1 out of 2, animal: destroy missed in step 3 because coreference resolution missed and because die maydestroy rule not triggered and rot has no rule, skeleton: missed create because inference from future steps needed and destroy off by one step, rock: got create, mold: missed create by one step

1145 - inputs: got both 2 and matched, outputs: got 1 out of 2, conversions: got the only 1, hydrogen fuel: matched destroy, hydrogen atoms: matched destroy, helium atoms: matched create, red giant star: missed create in step 5 because no proper evidence present for creation - annotation not the best.

1146 - inputs: got 2 out of 4, outputs: missed both 2 and predicted some other 1, conversions: got 1 out of 2, star: destroy missed in step 1 as burn has no associated destroy rule, hydrogen fuel: got destroy in step 2, hydrogen: got destroy in step 3, helium: got create but missed destroy in step 7 has maydestroy but not destroy rule meaning destroy was probably not triggered, energy: missed destroy in step 4 because qasrl generated incorrect question answer pair, carbon atoms: missed create in step 7 because no create rule present for combine, red giant star: missed create in step 10

because no proper evidence for creation present,

1147 - inputs: got 1 out of 2, outputs: got 1.5 out of 3, conversions: got 2.5 out of 3, hydrogen atoms: got step 1 destroy, helium atoms: got create but predicted extra destroy in step 4 because combine with does not have an associated rule - generalization might help here as there are other rules for combine, hydrogen: destroy off by one step and predicted extra create and destroy

1188 - inputs: missed the only 1, outputs: got the only 1, conversions: none and matched, coment: missed destroy in step 4 because no rule present with crash, crater: got create

1189 - inputs: none and match, outputs: missed the only 1, conversions: missed the only 1, chunks of rock: missed create in step 1 because there are no create rules with break and missed destroy in step 4 because chunks probably not related to chunks of rock - complex coreference resolution, crater: missed create in step 5 because seems like no verb present to indicate creation.

1190 - inputs: missed the only 1, outputs: none and match, conversions: missed the only 1, deoxygenated blood: missed destroy and create, oxygen: missed destroy in step 6, oxygenated blood: missed create and destroy, pick does not help recognize that an entity is converted leading miss of oxygenated blood creation, for destroy of oxygenated blood: it needs to be deduced that the oxygenated blood got destroyed - hard to deduce. deoxygenated blood: similarly destroy and creation of oxygenated blood missed.

1240 - inputs: missed 2 and predicted none, outputs: matched the only 1, conversions: missed all 2, Air: no problem with create or destroy, refrigerant liquid: destroy missed in step 3, liquid: create off by one step, refrigerant gas: create and destroy missed, for refrigerant liquid: creation missed, for refrigerant gas: gas probably not recognized as refrigerant gas (missed coreference resolution or entity recognition?) so creation and destroy missed.



APPENDIX E  
DOMAIN KNOWLEDGE

```

time(T):-observedAt(V,Q,A,T).
time(T+1):-observedAt(V,Q,A,T).

entity(E):-participant(E).
location(L):-lvalue(P,L).
location("?").
location("-").
answer(L):-observedAt(V,Q,L,T).
event(V):-observedAt(V,Q,L,T).
question(Q):-observedAt(V,Q,L,T).
groundObservation(V,Q,L,T):-observedAt(V,Q,L,T).
entityObservation(V,Q,L,T):-observedAt(V,Q,L,T), refers(T,P,L).

name(X):-description(N,X).
value(L,L):-location(L).
value(L,L):-answer(L).

%% observedAt
eobservedAt(V,Q,P,T):-observedAt(V,Q,L,T), refers(T,P,L).
lobservedAt(V,Q,L,T):-observedAt(V,Q,P,T), lvalue(P,L).

%colocation
%%cosituation(E,P,T):- create(E,T), eobservedAt(V,Q,P,T), E!=P, not create(P,T),exists(P,T).
cosituation(E,P,T):- create(E,T), mayDestroy(P,T), eobservedAt(V,Q,P,T), E!=P,
not create(P,T),holdsAt(locationOf(P,L),T),L!="-".

```

APPENDIX F  
INERTIA RULES

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input predicates:
% 1. time(1..n).
% 2. entity(1..m): the set of entities.
% 3. description(I,V) : string description of i-th entity
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% you observe the location at certain timepoints which are critical points
% for the points that are not critical points you decide the location based
% on the neighbour critical points

criticalPoint(E,T)      :- before(locationOf(E,L),T).
criticalPoint(E,T+1):- initiate(locationOf(E,L),T).
criticalPoint(E,T+1):- terminate(locationOf(E),T).
criticalPoint(E,T+1):- destroy(E,T).
criticalPoint(E,T+1):- create(E,T).

% assume the leftmost point is a critical point
criticalPoint(E,0)      :- entity(E).

% define non critical point
nonCriticalPoint(E,T):- entity(E), time(T), not criticalPoint(E,T).

% define the left endpoint of a non critical point
leftCriticalPoint(E,T,T1):- nonCriticalPoint(E,T), criticalPoint(E,T1), T1<T,
    not nleftCriticalPoint(E,T,T1).
nleftCriticalPoint(E,T,T1):- nonCriticalPoint(E,T),
    criticalPoint(E,T1), T1<T, criticalPoint(E,T2),
    T1<T2, T2<T.

% define the right endpoint of a entity at each point
rightCriticalPoint(E,T,T1):- entity(E), time(T), T1>T, before(locationOf(E,L),T1),
    not nRightCriticalPoint(E,T,T1).
nRightCriticalPoint(E,T,T1):- entity(E), time(T), before(locationOf(E,L),T1), T1>T,
    criticalPoint(E,T2), T1>T2, T2>T.
nRightCriticalPoint(E,T,T1):- entity(E), time(T), initiate(locationOf(E,L),T1-1), T1>T, time(T1).

% define has a right critical point
hasARightCriticalPoint(E,T):- rightCriticalPoint(E,T,T1).
hasNoRightCriticalPoint(E,T):- not hasARightCriticalPoint(E,T), entity(E), time(T).

% define first critical point is not created type
firstCriticalPoint(E,T):- criticalPoint(E,T), time(T), not nFirstCriticalPoint(E,T).
nFirstCriticalPoint(E,T):- criticalPoint(E,T), criticalPoint(E,T1), T1<T, time(T1), time(T).
nUsedBeforeCreation(E):- firstCriticalPoint(E,T), create(E,T-1).
isUsedBeforeCreation(E):- entity(E), not nUsedBeforeCreation(E).
firstCriticalPointTypeBefore(E):- firstCriticalPoint(E,T), before(locationOf(E,L),T),
    not initiate(locationOf(E,L1),T-1):location(L1).

% initialization (we can do this better with checking if it has a before)
holdsAt(locationOf(E,"?"),0):- isUsedBeforeCreation(E), not firstCriticalPointTypeBefore(E).
holdsAt(locationOf(E,L),1):- isUsedBeforeCreation(E),
    firstCriticalPointTypeBefore(E),before(locationOf(E,L),T).
holdsAt(locationOf(E,"-"),0):- entity(E), not isUsedBeforeCreation(E).

% define location at critical point
holdsAt(locationOf(E,L),T):- criticalPoint(E,T), before(locationOf(E,L),T).
%what about does not exist?
holdsAt(locationOf(E,L),T):-
    criticalPoint(E,T), initiate(locationOf(E,L),T-1),
    not before(locationOf(E,L1),T):location(L1):L1!=L.

```

```

holdsAt(locationOf(E,"-"),T):-criticalPoint(E,T), destroy(E,T-1), time(T-1).
holdsAt(locationOf(E,"?"),T):-criticalPoint(E,T), terminate(locationOf(E),T-1), time(T-1), time(T),
    not initiate(locationOf(E,L),T-1):location(L), hasNoRightCriticalPoint(E,T-1).
holdsAt(locationOf(E,"?"),T):-criticalPoint(E,T), create(E,T-1), time(T-1), time(T),
    not initiate(locationOf(E,L),T-1):location(L), hasNoRightCriticalPoint(E,T-1).
% fix this : the value when terminate || create followed by before
holdsAt(locationOf(E,L),T):-criticalPoint(E,T), terminate(locationOf(E),T-1), time(T-1),
    not initiate(locationOf(E,L1),T-1):location(L1),
    rightCriticalPoint(E,T,T1), holdsAt(locationOf(E,L),T1).
holdsAt(locationOf(E,L),T):-criticalPoint(E,T), create(E,T-1), time(T-1),
    not initiate(locationOf(E,L1),T-1):location(L1),
    rightCriticalPoint(E,T,T1), holdsAt(locationOf(E,L),T1).

% define propagation
% If the right critical point is not backward type
% propagate the value from left (i.e. also why left must exist)
holdsAt(locationOf(E,L),T+1):- nonCriticalPoint(E,T+1), hasNoRightCriticalPoint(E,T+1),
    holdsAt(locationOf(E,L),T).

% If the right is backward type
% if the left type is termination || creation with no initiation Then propagate back
holdsAt(locationOf(E,L),T):- nonCriticalPoint(E,T), rightCriticalPoint(E,T,T1),
    holdsAt(locationOf(E,L),T1),leftCriticalPoint(E,T,T2), T2>0, create(E,T2-1),
    not initiate(locationOf(E,L1),T2-1):location(L1).

holdsAt(locationOf(E,L),T):- nonCriticalPoint(E,T), rightCriticalPoint(E,T,T1),
    holdsAt(locationOf(E,L),T1),leftCriticalPoint(E,T,T2), T2>0, terminate(locationOf(E),T2-1),
    not initiate(locationOf(E,L1),T2-1):location(L1).

% If the left has a known value then there are possibilities
% case 3: conflicting propagation
% base case (the point before the right critical can take any value of left/right)
holdsAt(locationOf(E,L),T-1):- nonCriticalPoint(E,T-1), rightCriticalPoint(E,T-1,T),
    holdsAt(locationOf(E,L),T), not holdsAt(locationOf(E,L1),T-1):location(L1):L1!=L.

holdsAt(locationOf(E,L),T-1):- nonCriticalPoint(E,T-1), rightCriticalPoint(E,T-1,T),
    holdsAt(locationOf(E,L),T1),
    not holdsAt(locationOf(E,L1),T-1):location(L1):L1!=L, leftCriticalPoint(E,T-1,T1).

holdsAt(locationOf(E,L),T):- nonCriticalPoint(E,T), rightCriticalPoint(E,T,T1), T<T1-1,
    holdsAt(locationOf(E,L),T+1), holdsAt(locationOf(E,L),T1),
    not holdsAt(locationOf(E,L1),T):location(L1):L1!=L.
holdsAt(locationOf(E,L),T):- nonCriticalPoint(E,T), rightCriticalPoint(E,T,T1), T<T1-1,
    leftCriticalPoint(E,T,T2), holdsAt(locationOf(E,L),T2),
    not holdsAt(locationOf(E,L1),T):location(L1):L1!=L.

%%% constraints %%%%
:- initiate(locationOf(E,L),T), destroy(E,T), time(T), entity(E).
:- before(locationOf(E,L),T+1), destroy(E,T), time(T), entity(E).

```

## APPENDIX G

### CODE REPOSITORY

The Code and intermediate data is available at  
<https://github.com/aurghob/ThesisMaterial> .