Toward Better Reasoning from Natural Language

by

Adam Lee Purtee

Submitted in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Supervised by

Professor Lenhart Schubert and Professor Daniel Gildea

Department of Computer Science
Arts, Sciences and Engineering
Edmund A. Hajim School of Engineering and Applied Sciences

University of Rochester
Rochester, New York

2018

Table of Contents

Biographical Sketch **vi**

Abstract **vii**

Contributors and Funding Sources **viii**

List of Tables **ix**

List of Figures **x**

1 Introduction **1**

1.1 Motivation 1

1.2 Summary of the Dissertation 3

2 Self-Training for Parsing **5**

2.1 Parsing for Knowledge Acquisition 5

2.2 Chapter Background 6

2.2.1 Statistical Parsing 6

2.2.2 Self-training 11

2.3 Feature Exploration 13
2.3.1 Sentence Length 13
2.3.2 Observations Pertaining to Sentence Length 14
2.3.3 Bigram Novelty 16
2.4 Sample Selection for Self Training 20
2.5 Discussion 22

3 Tree Transduction **24**
3.1 Introduction 24
3.2 Related Work 27
3.3 Formal Models of Tree Transduction 29
3.4 The TTT Language 38
3.4.1 Pattern Matching 38
3.4.2 Transductions 43
3.4.3 Theoretical Properties 45
3.5 Example Applications 48
3.5.1 Linguistic 48
3.5.2 Logical 52
3.6 Chapter Conclusion 57

4 Logical Reasoning and Probability **58**
4.1 Chapter Introduction 58
4.2 Logic as Knowledge Representation 59
4.2.1 First-Order Logic 60

4.2.2 Episodic Logic 62
4.2.3 Probability, Logic, and Generalized Quantifiers 64
4.3 Challenges and Desiderata 67
4.3.1 Ambiguity and The Nixon Diamond 67
4.3.2 Specificity and The Birds 68
4.3.3 Stochastic Causal Behavior 69
4.3.4 Relational Domains 71
4.4 Evidence Combination 71
4.4.1 Causally Motivated Approaches 73
4.4.2 Entropy Maximization 83
4.5 Related Work 86
4.5.1 Random Worlds 89
4.5.2 Inheritance Networks 92
4.5.3 Markov Logic Networks 94
4.6 Algebraic Probabilities 96
4.7 Simple Rules for Probabilistic Reasoning 98
4.7.1 a-Rules 100
4.7.2 b-Rules 104
4.7.3 c-Rules 106
4.8 Inference with Our Proposed Rules 109
4.8.1 Rule Combination 109
4.8.2 Negative, Disjunctive, and Existential Rule Consequents 110

4.8.3 Obtaining Final Probabilities 111

4.9 Looping back: Tweety the Chicken 113

4.10 Looping back: The Nixon Diamond 119

4.11 Experiments 125

4.11.1 Setup 125

4.11.2 The Denormalization Problem 127

4.11.3 Results 128

5 Conclusion **129**

5.1 Concluding Remarks 129

Bibliography **134**

Appendix A Probabilistic Parameter Independence **142**

Appendix B C-Rule Convergence Example **151**

Biographical Sketch

Adam Purtee was born in Paragould, Arkansas. He attended Arkansas Tech University and graduatated with Bachelor of Science degree with a double major in Computer Science and Mathematics. He attended the University of Rochester and graduated with a Master of Science degree in Computer Science in 2012. He began dissertation work at the University of Rochester in 2012. He pursued his research under the supervision of Professor Daniel Gildea and Professor Lenhart Schubert. He began lecturing full time at the Rochester Institute of Technology in 2016.

The following publications were a result of work conducted during doctoral study:

- A. Purtee and L.K. Schubert, Simple Rules for Probabilistic Commonsense Reasoning, in Proc. Advances in Cognitive Systems, Troy, NY, May 13 2017.
- A. Purtee and L.K. Schubert, (Abstract) Simple Rules for Probabilistic Common Sense Reasoning, in Proc. of Argument Strength Workshop, Bochum, Germany, November 30th 2016.
- A. Purtee and L.K. Schubert, TTT: A tree transduction language for syntactic and semantic processing, in Proc. EACL 2012 Workshop on Applications of Tree Automata Techniques in Natural Language Processing, Avignon, France, April 24 2012

Abstract

This dissertation explores improvements to reasoning from natural language through three distinct lines of investigation. Our contributions include an empirical investigation of the problem of sample selection for self-training syntactic parsers, the development of the tree-to-tree transduction system TTT, and expanded knowledge representation for first-order probabilistic commonsense reasoning. We find that constraining the automatically labeled data by sentence length and vocabulary novelty has a strong but noisy effect on the overall accuracy of self-trained models. Applying support vector regression using these features led us to obtain a new state of the art F_1 measure of 92.52 on the standard WSJ subset of the Penn Treebank, when parsing with a discriminatively reranked generative model. The TTT language developed herein has proved to be a useful tool for manipulating both treebank parses and logical forms. The language borrows ideas from regular expressions and formal tree transducers. It is Turing complete and therefore able to compactly express a wide variety of transformations. Finally, the extensions we develop for symbolic logic are built upon set theory, causality, and an algebraic probability framework with well-defined algorithms for lifted inference. We validate the core techniques of our approach by showing favorable performance with respect to previous work in a relational logic domain. By virtue of these contributions we have pushed the field of artificial intelligence toward better reasoning from natural language.

Contributors and Funding Sources

This work was supervised by a dissertation committee consisting of Professors Lenhart Schubert, Daniel Gildea, and Daniel Stefankovic of the Department of Computer Science, and Professor Greg Carlson of the Department of Linguistics. Chapter 2 is based on Purtee and Schubert (2012). Portions of Chapter 4 are based on Purtee and Schubert (2017). This material is based upon work supported by NSF awards #IIS-1446996 & IIS-1016735, DOD/ONR #N00014-11-1-0417, SRI #1900209,The Friedland Group #003778 and IHMC Sub-contract #W911NF-15-1-0542. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of above named organizations.

List of Tables

2.1 Advances in state-of-the-art F_1 over time on section 23 of the WSJ dataset. 11

2.2 Results of SVM-based sample selection (using unknown bigram count and sentence length as features). 22

2.3 Random Sampling Baseline for Self-trained Parsers 22

3.1 Example Bindings for TTT Variables 42

4.1 Some generalized quantifiers in English. The first two rows are adverbials, the third row exemplifies proportions and counting, and first-order quantifiers forming the last row. 63

4.2 AUC Results predicting Advisor-Advisee Relations 128

List of Figures

1.1 A Language Understanding Pipeline 4

2.1 Histogram of Sentence Length over the NANC 14

2.2 Relative quality of self-trained models when using samples of 100k length-constrained (parsed) sentences 15

2.3 Utility by Sentence Length for 2M Word Samples. 15

2.4 Utility by Unknown Bigram Count for 2M Word Samples 17

2.5 Unknown Bigram Count Correlates with Sentence Length 18

2.6 Histogram of Sentence-level Unknown Bigram Count Across the NANC. 18

2.7 Effect of Bigram Novelty Rate on F_1 for Self-trained Parsers 19

2.8 Histogram of Bigram Novelty Rate Across the NANC. 19

4.1 The Nixon Diamond 68

4.2 Specificity and the birds. 70

4.3 Each of the ϕ_n may independently cause event ψ. 74

4.4 An event with several possible effects. 81

4.5 An event with several possible causes and effects. 81

4.6 An acyclic, bipartite graph of causes and effects. 82

4.7 The “W” graph. 82

4.8 The Burglary-Alarm-Earthquake Example. 97

4.9 Chicken network. 114

4.10 Duck network. 115

4.11 The Nixon diamond. 120

4.12 Nuanced causal prevention 124

4.13 Nuanced causal prevention, alternate figure. 124

B.1 An taxonomic spectrum example. 152

1 Introduction

1.1 Motivation

Natural language is a powerful tool for knowledge representation. We use language to share details of our days with loved ones, examine morality and ethics through philosophy, and as a catalyst for political movements throughout history. Our knowledge of history, medicine, science, and numerous other fields of human endeavor are captured eternally in the written word. The benefits of natural language understanding are limitless – and the technical challenges posed by the problem are correspondingly difficult.

Although natural language is a robust and extensive system for knowledge representation, it is not the only option. Within the field of artificial intelligence, symbolic logic is often used to represent knowledge. Though lacking the expressivity of natural language, symbolic logic has substantial merits. Carefully written symbolic logic can unambiguously represent intricate subjects such as law, mathematics, and scientific arguments. Due to its long history, efficient algorithms for reasoning with symbolic logic are plentiful, and its compositional, context-free structure lends itself to translation back into natural language.

More recently, connectionist methods based on deep neural networks and correlative statistical inference have captured much of the attention of the field; however, these methods are at present vastly inadequate even for simple inferences. E.g., consider the performance of your favorite smartphone voice assistant when posed even the simple question: "If I have two apples and two oranges, how many fruits do I have?" Bag-of-words and sequence-to-sequence based methods are highly unlikely to leverage the facts that apples and oranges are both kinds of fruit, and therefore will be unable to obtain the correct answer. In contrast, when faced with such commonsense reasoning challenges, systems built upon symbolic logic are capable of actual inference and are much more likely to yield suitable answers. For a more detailed set of limitations of connecionist models, see Marcus (2018).

Our work broadens the link between language and logic. We establish a new way of representing uncertain knowledge within first-order logic, and provide an intuitive inference mechanism based on an embedded form of resolution. These extensions support applications such as problem solving, question answering, planning, and our particular focus, which is the task of commonsense reasoning.

Commonsense reasoning requires an enormous bank of both specific facts and general rules – likely hundreds of thousands, if not millions, of pieces of information. Such quantities of data are available through large corpora and internet sources (e.g., Wikipedia); however, because this knowledge is expressed as natural language, we must still meet the task of translation into symbolic logic.

One path to a solution begins with the notion of compositional semantics, which can be used to translate a sentence into logic where words become predicates and syntactic structure guides decisions such as quantifier scope, co-reference resolution, etc. Though the problem of syntactic parsing has been long investigated, it remains true

that even highly accurate parsers at the phrase level typically will have one or more errors at the sentence level. These errors become poison pills in the translation process, resulting in nonsensical and contradictory statements which severely degrade the quality of automatically constructed knowledge bases. Therefore, we also investigate the task of syntactic parsing and show that simple sample selection methods together with semi-supervised learning can be used to obtain better syntactic parsers.

1.2 Summary of the Dissertation

We aim to develop better techniques for commonsense reasoning from natural language. Non-trivial subsets of human knowledge can be expressed in symbolic logic. Automated reasoning in formal logic is a well-studied problem with industrial strength solutions; however, its expressiveness is pale in comparison to natural language. In this work, we extend first order logic to incorporate probability. We show that existing algorithms for automated reasoning can be used together with an algebraic probability framework to reason with our extended representation.

Due to the great variability of expressions in natural language, as well as the ambiguity of their meanings, a robust transformation from language to logic is difficult to obtain. We use the classic technique of divide and conquer to decompose the language understanding problem into a pipeline from sentences to logic, using syntactic parses as intermediate forms. As mentioned in the abstract, this dissertation includes contributions to three distinct stages of such a pipeline (see Figure 1.1).

Beginning with our work on syntactic parsing, we show how semi-supervised training may be used to obtain improved probabilistic grammars, and ultimately higher quality parses. We show that sentence length and bigram novelty are important predictors of

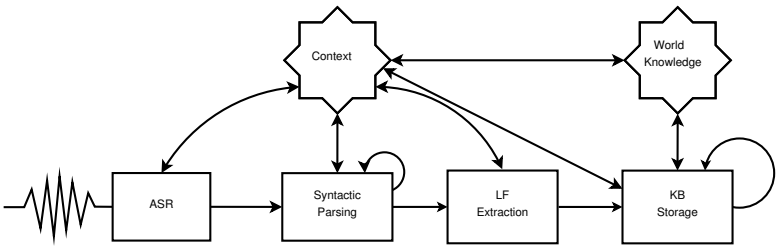


Figure 1.1: A Language Understanding Pipeline

training utility for sentences, that higher quality parsers may be obtained by simply rejecting outliers according to these features, and that due to a surprisingly high amount of variance observed during self-training, a simple random restart procedure reliably improves F_1 measure on standard datasets.

We then detail our Tree-to-Tree Transduction language (TTT), which is a pattern-based rewriting system with several applications to natural language processing and logical reasoning. We give example applications of TTT in support of parse tree repair, as an aid to the conversion from parse trees to logical forms, and for manipulation of symbolic logic. The TTT work was presented at the 2012 meeting of the European Chapter of the Association for Computational Linguistics, in Avignon, France.

Lastly, we present first-order extensions to symbolic logic together with an algebraic probability framework which support probability-aware commonsense reasoning. Building upon foundations of modal logic, nonmonotonic reasoning, graphical models, and resolution based theorem proving, we add to Episodic Logic three kinds of probabilistic quantification. Each of our quantifiers are derived from intuitive, yet distinct, methods of probabilistic reasoning. We validate our extensions using a publicly available relational logic dataset, and report results competitive with previous approaches based on Markov Logic Networks. This work led to publications at the 2016 Argument Strength workshop in Bochum, Germany and the 2017 meeting of Advances in Cognitive Systems in Troy, NY.

2 Self-Training for Parsing

2.1 Parsing for Knowledge Acquisition

Our aim is an automated system for commonsense reasoning. Such a system requires a great deal of background knowledge – a longstanding challenge known as the knowledge acquisition bottleneck. The wide availability of natural language text makes it an attractive potential source of such knowledge. While there exist methods for reasoning directly with language (MacCartney and Manning, 2014), symbolic logic provides a precise representation which is amenable to mechanical manipulation.

Schubert and Hwang (2000) describe a translational approach from natural language to episodic logic. Their four stage method similarly begins translation of a sentence with syntactic parsing to obtain a parse tree which is then compositionally processed with a set of match syntactic/semantic patterns yielding an intermediate, incomplete logical form. Next, quantifier scopes are determined before a final deindexing pass where relative concepts such as time and place (e.g., words such as now or here) are effectively skolemized to constants reflecting the context of the sentence. The KNEXT system uses similar techniques to obtain possibilistic factoids from syntactically parsed

text (Schubert, 2002; Van Durme and Schubert, 2008; Van Durme et al., 2009). Borrowing an example from the KNEXT authors, the sentence "The man walked through the door of the house" yields such factoids as "A man may walk" and "Houses may have doors".

Relying on syntactic parsing as an intermediate step means that parser error may become magnified into logic errors. State of the art parsers are quite good, but even with 90%+ phrase-level accuracy most parsed sentences will involve at least some incorrect annotations. Therefore, one way of indirectly attacking the knowledge acquisition bottleneck is to improve the quality of automatically obtained parses. In this work, we focus on how to make better use of the technique of self-training to establish more robust and accurate parsing models. We first discuss core methods of statistical parsing and established work on self-training for parsing. Then, we present the results of our work on feature based sample selection.

2.2 Chapter Background

2.2.1 Statistical Parsing

At the broadest level, parsers fall into either the syntactic or semantic domains. Although much work has been done to develop direct semantic parsers, there is no consensus on the preferred semantic representation for natural language. Most syntactic parsers are either dependency-based or constituency-based, according to whether or not they consider pairwise relations between words, or develop a grammatical tree, respectively.

Most constituency-based parsing methods are founded upon probabilistic context free grammars (PCFGs), which are a compact, precise means to represent a distribu-

tion over possible trees. While powerful supervised learning algorithms can be used to learn PCFG parameters from large datasets, these algorithms still primarily rely on *manually*-annotated (or at least manually corrected) data. The most commonly used set is the Wall Street Journal corpus, which is a portion of the Penn Treebank. Our knowledge extraction systems are built upon the latter due to the direct mappings between compositional syntax trees and logical forms, and we therefore focus on improving syntactic treebank parsing.

As already mentioned, an open-domain knowledge extraction system relying on compositional semantics by definition requires a robust, open-domain syntactic parser. Within the newswire domain, statistical parsers tend to be fairly reliable; however, parse quality drops off dramatically for out of domain sentences (Gildea, 2001), which means a well-performing (at the sentence level), open-domain parser is still not quite within reach. One approach to solving this problem could be create a large, open-domain training set; however, manual annotation construction of a data set spanning all genres of text and domains is impractical. Instead, we use a hybrid approach of manual and automatic annotation known as self-training (McClosky et al., 2006, 2008, 2010). The key idea is to use a standard parsing model to parse a set of unlabeled sentences, then to train a second parser model on the union of the original training data and the automatically labeled set.

The remainder of this section summarizes fundamental background on statistical parsing, and self-training. We begin with self-training and PCFGs. Then, we describe a series of sampling and feature-based approaches to the problem of sample selection, where we show that simple techniques can glean even more benefits from self-training by making better use of unlabeled data.

Probabilistic Grammars

The key formalism behind most syntactic parsers is the context-free grammar. A *probabilistic* context free grammar (PCFG) is a four tuple $G = (P, S, N, T)$, where T is a finite set of terminal symbols, N is a finite set of non-terminal symbols, $S \in N$ is the start symbol, and P is a set of finitely many weighted productions of the form $A \rightarrow^w \alpha$, where $A \in N$, $\alpha \in (N \cup T)^*$, and $0 \leq w \leq 1$. The weight of a rule r is referred to as $w(r)$ or as w_r. A PCFG licenses a set of derivations, which are sequences of productions beginning with the start symbol and ending when only terminal symbols remain at the leaves. These derivations have a natural tree structure, and the weight of a tree is product of the weights of the rules applied in its derivation. I.e., the probability of a tree t with derivation d is $P(t) = \prod_{rules\ r \in d} w(r)$. In practice, PCFG rules are further lexicalized so that nonterminals are associated with headwords further down in the derivation.

Rules can be extracted from a labeled treebank, and rule probabilities can be established maximum likelihood estimation (MLE), where the weight of a rule is the given by the ratio of the number of times the rule appears in the treebank to the number of times the lhs non-terminal appears in the treebank. E.g,

$$P(A \rightarrow BC) = \frac{\#\{A \rightarrow BC\}}{\Sigma_{B,C}\#\{A \rightarrow BC\}}$$

In practice, this training method usually overfits to the training data treebank and does not yield a model that generalizes well to new input. To overcome this, various forms of regularization or discounting are typically applied during parameter estimation.

The two most common algorithms for parsing with context free grammars are the CYK algorithm (Younger, 1967; Kasami, 1965) and the Earley method (Earley, 1970).

CYK uses bottom-up dynamic programming to compute best derivations for each sub-span, for each possible originating non-terminal. CYK is particularly easy to implement efficiently and also corresponds to Viterbi maximization. The Earley method is a left to right predictive parsing algorithm which keeps track of partially completed states by associating rules with covered spans and a "dot" reflecting the position within the rule at which the span ends. This algorithm is especially flexible when handling unknown words, because it includes explicit calls to a lexical scanner that maps directly from words to sets of possible parts of speech. Both CYK and Earley parsing methods are cubic time with respect to sentence length.

Discriminative Reranking

Our work on self-training primarily uses the Brown parser (Charniak and Johnson, 2005) which uses a lexicalized PCFG to generate a ranked list of the top-k parses according to the PCFG, and then re-ranks the parses using a separate maximum entropy model. They motivated re-ranking with the fact that, for the WSJ text, the 50 best PCFG parses have an oracle f-score of 96.8, an improvement over the 89.7 fscore for just the top parses alone.

Their model assigns scores to trees using a weighted feature vector: $v_\theta(t) = \theta \cdot f(t)$. The features are binned into schemas, and reflect such properties as conjunct parallelism, right branching, heaviness (count of dominated preterminals), n-grams, head word tags, and several others. Their total number of distinct features is a little over one million – which is as many words as are in the WSJ and over twenty times as many trees. The final parse is simply the one with the highest score:

$$\hat{t} = \arg\max_{t \in Y(s)} v_\theta(t).$$

The re-ranker is trained so that the sum of a loss function $L_D(\theta)$ and regularization function $R(\theta)$ applied to the entire dataset are minimized. The loss function is negative conditional likelihood of the best parses (chosen by F_1 score vs gold data) given the set of n-best parses. I.e., given a fixed n-best list from the PCFG, we want the reranking stage to put as much likelihood as possible on the true best parse from the n-best list, when comparing them by F_1 relative to the training data labels. This loss function is due to Riezler et al. (2002). The regularization function is simple sum of squares of feature weights. Unlike learning, inference with discriminative reranking is very fast and does not require normalization.

$$\hat{\theta} = \arg\min_{\theta} L_D(\theta) + R(\theta)$$

Treebanks

Finally, we discuss the most often used available training data in the literature. The Penn Treebank (PTB) (Marcus et al., 1993) is the primary data set used to train statistical parsers. Performance is usually evaluated with respect the Wall Street Journal (WSJ) subset of the PTB. The paper that systematized parser evaluation was Black et al. (1991). The two metrics of interest are *precision*, which is the percentage of constituents in a proposed parse that are present in a gold standard parse, and *recall*, which is the ratio of the total number of correct constituents in a proposed parse to the total number of constituents in a gold standard parse. A constituent is correct when it spans the right words and has the right label. Note that a correct constituent may have an incorrect sub-constituent. Because it is easiest to compare systems using a single score, the harmonic mean of precision (the fraction of reported constituents which are correct) and recall (the fraction of correct constituents which are returned) is often computed.

F_1	Citation	Restrictions
87.79	(Collins, 1997)	n$\leq$100
88.34	(Collins, 1997)	n$\leq$40
90.15	(Petrov et al., 2006)	n$\leq$40
91.02	(Charniak and Johnson, 2005)	n$\leq$100
89.70	(Petrov et al., 2006)	none
92.10	(McClosky et al., 2006)	none
92.45	(Huang et al., 2010)	none
92.52	*This Work*	none

Table 2.1: Advances in state-of-the-art F_1 over time on section 23 of the WSJ dataset.

This harmonic mean is referred to as the F_1-score. That is,

$$F_1 = \frac{2PR}{P+R}$$

The standard setup is to use sections 2-21 (of the WSJ set) as training data, section 24 for tuning, and section 23 as a held-out evaluation set. We report a timeline of progress on this task, including our results as described later in this chapter, in Table 2.1

2.2.2 Self-training

The earliest experiments with self-training treebank parsers are due to Charniak (1997), who established that the gains from self-training a pure PCFG based parser were not significant. Few results are discussed in the literature again until nearly a decade later in a paper byMcClosky et al. (2006), where self-training for parsing was observed to improve performance when using a *two stage* discriminatively re-ranked parser. They report that gains are only obtained when only the PCFG stage is self-trained. I.e., that end-to-end self-training of the PCFG and re-ranker together did not improve performance. Interestingly, this improved accuracy also holds when comparing the self-trained PCFG to the baseline PCFG models (i.e., when bypassing the re-ranker during

test time).

McClosky et al. (2006) also explored the effects of varying the amount of unlabeled data added and the degree to which the hand-labeled training data should be duplicated. They report an asymptotic effect where the benefit from additional unlabeled data tapers off after around 2 million sentences added. They also report that giving additional weight to the hand-labeled training data relative to the automatically labeled data improves performance. The overall best setup in their paper was to add 1.75 million parsed sentences and to up-weight the training data by a factor of five.

In a later paper, they examine the characteristics of sentences which tend to be more accurately parsed by self-trained models than baseline models – i.e., where the gains from self-training were going (McClosky et al., 2008). Their key observation was that those sentences were found in what they called a "Goldilocks" range of lengths. The short sentences were already easy, the longest sentences were hopeless, but sentences of moderate length had a chance of being more accurately parsed after self-training than without. They also observed that sentences with known words in novel combinations were also being parsed more accurately.

Our work examines a complementary aspect of self-training for parsing. Namely, do there exist observable properties of sentences which indicate that self-training on them would lead to better models? Although we might perhaps ideally all of the available data, practical software implementations of parsers running on physical hardware are subject to limited memory and time – and even if those are plentiful, there is still the observed asymptotic effect to contend with. The nature of self-training means that virtually any natural language source may be used as training data, so there will almost certainly be "too much" unlabeled data. This motivates the problem of sample selection for self-training. I.e., which sentences are worth parsing and incorporating into our

training data?

2.3 Feature Exploration

2.3.1 Sentence Length

We began our work with an idea that self-training could be used to bootstrap a parser, by parsing sentences of incremental difficulty. As the most obvious feature, we begin with the relationship between sentence length and self-training utility. For reference, we present a histogram of sentence length in our tokenization of the NANC in Figure 2.1.

We use F_1 measure as our utility function and aim to to find sentence-level features which predict model-level F_1; however, because of the large batch size of self training, direct evaluation of an individual sentence is not possible. Instead, we take equal sized samples from a subset of the available data which has been constrained such that the feature in question for each sentence is within a narrow window. Specifically, we take 5 samples of 100K sentences from subsets of the NANC with length $x \pm y$ for $y \in 5, 10, \ldots, 40$. We then self-train a model on each sample together with the labeled training data and evaluate labeled bracketing F_1 measure on development data. We plot the data for all of our sample points in Figure 2.2. We also include two points of reference corresponding to a baseline without self-training of 90.40 and the mean of models self-trained on random samples of 100K sentences of 90.93 (all scores are with respect to the self-training devset WSJ24).

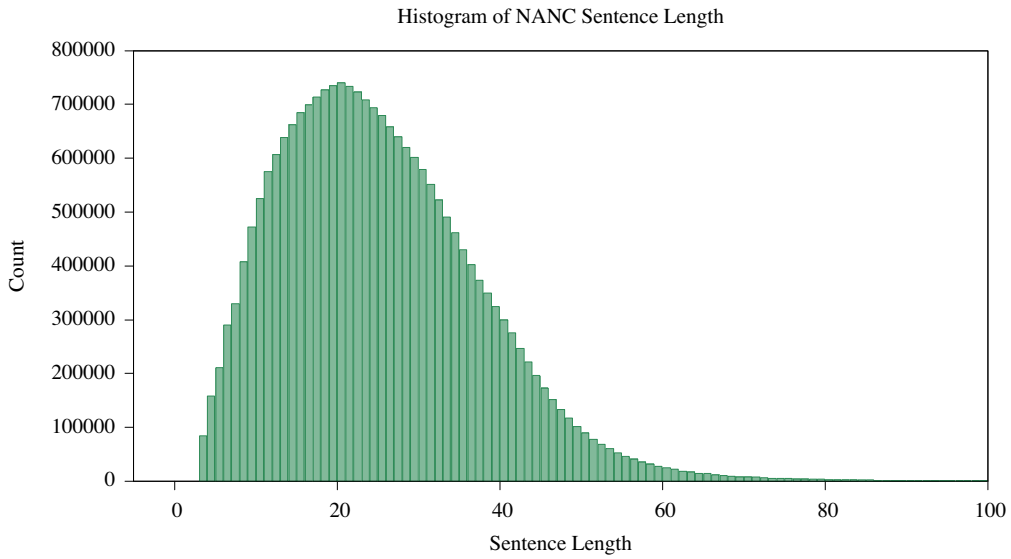


Figure 2.1: Histogram of Sentence Length over the NANC

2.3.2 Observations Pertaining to Sentence Length

The results are counter-intuitive. Rather than being of extra benefit to self-training, shorter sentences tend to actually *harm* performance relative to a baseline without any self-training at all. As can be seen in Figure 2.2, the labeled bracketing f-measure on held-out development data after self-training entirely on sentences with fewer than 10 words is distinctly worse than the baseline performance of 90.40 without any self-training at all.

Another distinct observable property is that even among sets of sentences with very narrow ranges of lengths, their self-training utility varies significantly. We emphasize that the methods of self-training (and correspondingly, PCFG rule learning) involve no uncertainty. As a sanity check, we self-trained on individual sets of sentences more than once and observed exactly zero performance difference, as expected. It is therefore clear that sample selection for self-training induces a significant amount of variance in the quality of the model obtained.

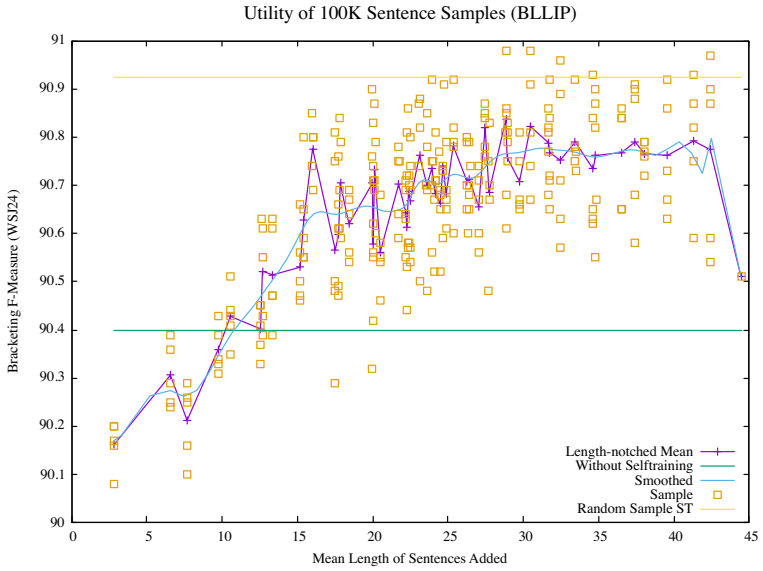


Figure 2.2: Relative quality of self-trained models when using samples of 100k length-constrained (parsed) sentences

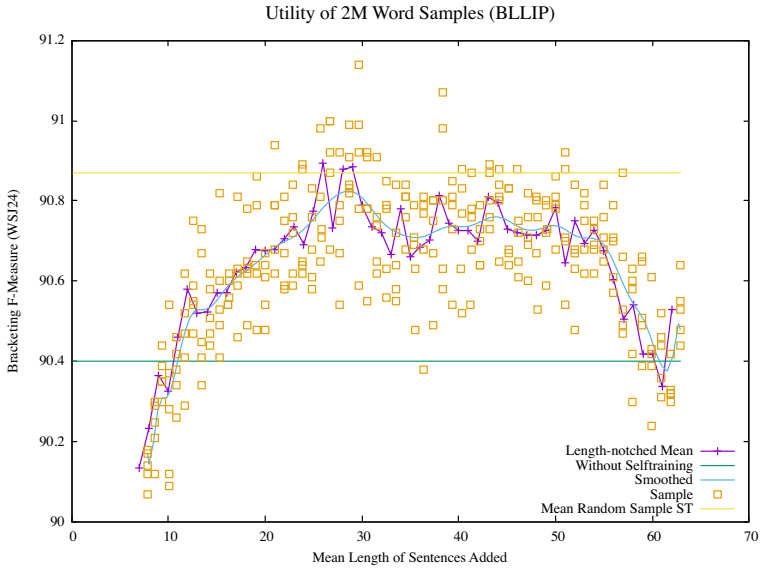


Figure 2.3: Utility by Sentence Length for 2M Word Samples.

A potential explanation for the harmfulness of short sentences is that they do not contain sufficient data. Therefore, we sample by sentence length again but this time hold constant the total number of words at 2M in Figure 2.3. Once again, we find that self-training on short sentences leads to worse performance than the baseline without any self-training; however, in this setup a clear quadratic "Goldilocks" effect can be observed where the most utility comes from sentences which are neither too long nor too short. This result complements the earlier results of (McClosky et al., 2008) who report that sentences in this range are more accurately parsed by self-trained models. Here, we find that such sentences are also more useful during the training phase.

Finally, we note that the majority of length-restricted sample points tend to be above the baseline without self-training, but below the means of models self-trained on random samples (of 90.93 for samples of 100K sentences and 90.87 for samples of 2M words). From this, we conclude that sentence length does not give a strong indication of utility, but it can be used to filter out useless and potentially harmful samples.

2.3.3 Bigram Novelty

We investigate the idea of using the sentence level counts of unknown bigrams as a sample selection method. As discussed in Section 2.2.2, McClosky et al (2008) report that self-training tends to yield better parses of sentences which have known words in unknown combinations. I.e., sentences containing novel bigrams but few unknown words. In this section, we do not consider the effect of unknown words, and focus entirely on unknown bigrams.

We consider a bigram to be unknown if it does not appear in the training data subset of the WSJ. Similar to our exploration of sentence length, we begin by filtering out all sentences containing a number of unknown bigrams outside a small window (+/- 1)

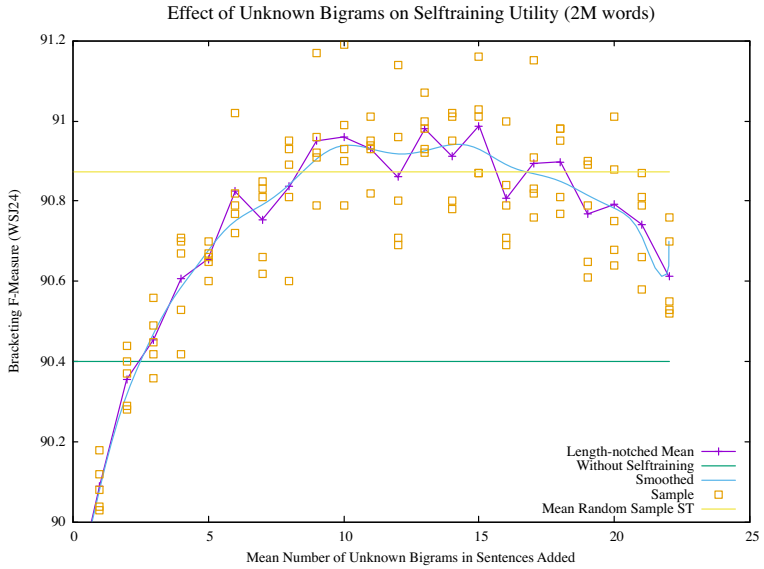


Figure 2.4: Utility by Unknown Bigram Count for 2M Word Samples

and move that window along the available data. We then randomly sample 2M words worth of sentences, use the associated baseline parses for self-training (along with the original training data), and evaluate the F_1-measure on our held-out development data (WSJ 24). Results are presented in Figure 2.4. The distribution of bigram novelty across the complete NANC is presented in Figure 2.6.

We immediately found the number of unknown bigrams to correlate almost exactly with sentence length (see Figure 2.5). In an effort to obtain a more independent view of the data, we also sample by the ratio of unknown bigrams to total sentence length. We call this the bigram novelty rate (bnr). Once again, we sample from within narrow windows of +/- 0.1 bnr, self-train on those samples, and evaluate F_1 on development data; however, here we also restrict the sentences to those with a length above 20. Results are presented in Figure 2.7, and the distribution of bigram novelty rate across the NANC is presented in Figure 2.8.

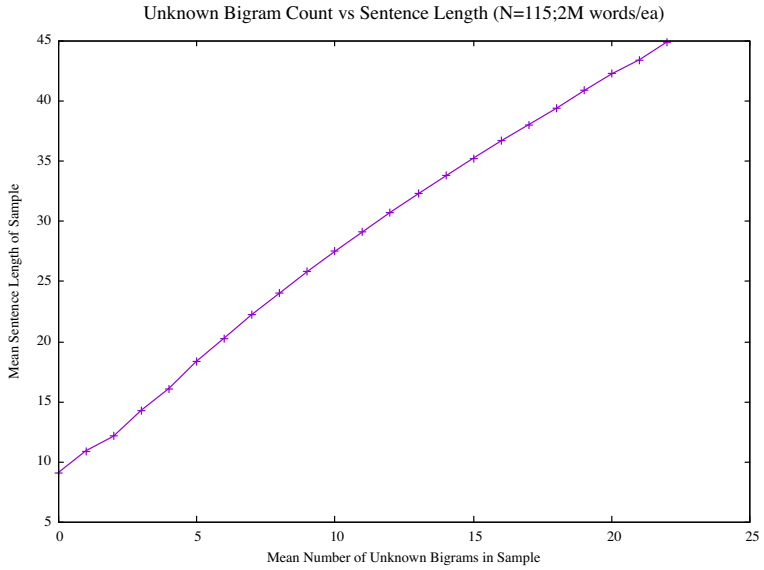


Figure 2.5: Unknown Bigram Count Correlates with Sentence Length

.

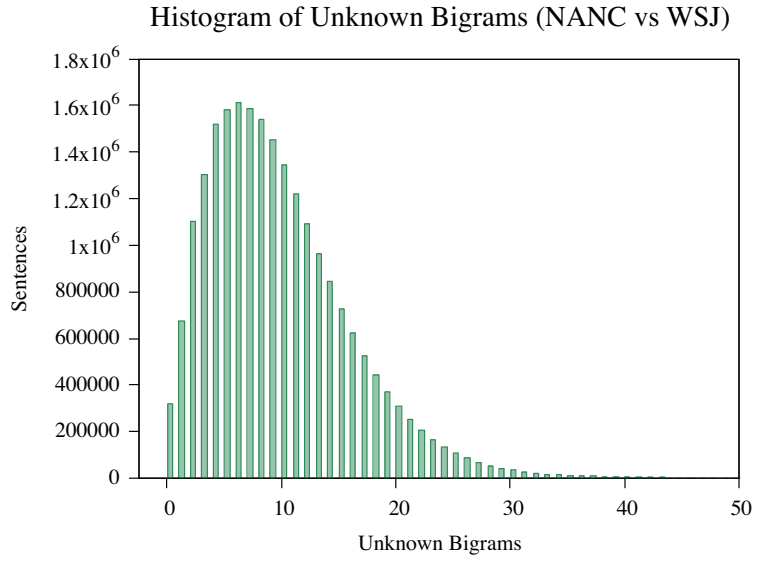


Figure 2.6: Histogram of Sentence-level Unknown Bigram Count Across the NANC.

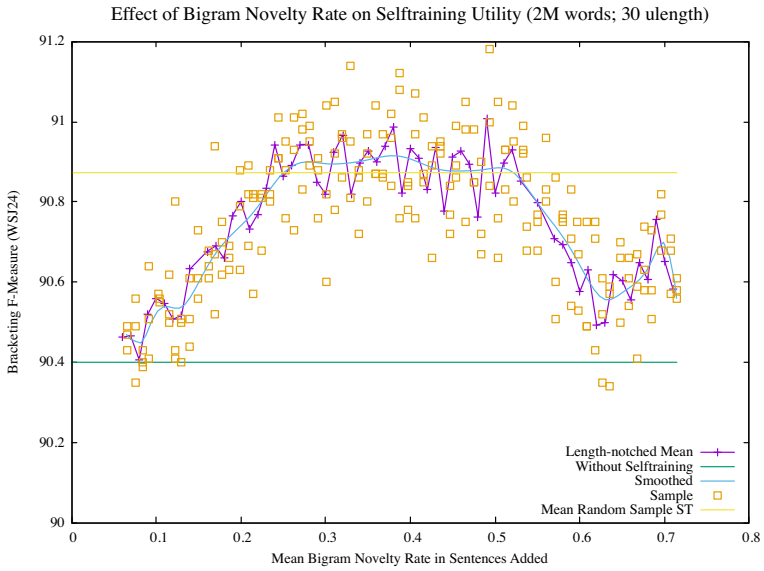


Figure 2.7: Effect of Bigram Novelty Rate on F_1 for Self-trained Parsers

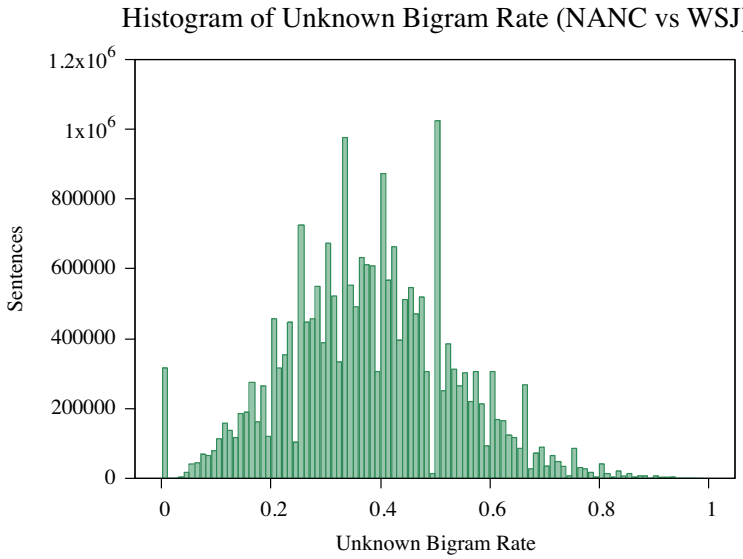


Figure 2.8: Histogram of Bigram Novelty Rate Across the NANC.

2.4 Sample Selection for Self Training

In this experiment, we seek sample selection critera to make the best use of unlabeled data. To do so, we train a support vector regression model on means of sample sentence features and the F_1 measure of their associated self-trained models on development data (WSJ24). We then rank all of the automatically parsed data by predicted F_1, and self-train models using the top $100K$ and top $5M$ sentences.

We show that our method yields reliable improvements in parser accuracy and finally discuss some limitations and possible extensions of our work.

Experiment Setup

We use the BLLIP parser (Charniak and Johnson, 2005), which is a two stage pipeline built from a lexicalized PCFG and a discriminative re-ranking model. This is the same parser as was used in self-training experiments in the literature discussed earlier. We use the WSJ portion of the Penn Treebank as our labeled data. We use sections 2–21 as our baseline parser training data and sections 22 for tuning the parser and re-ranker hyperparameters (Dev1), and section 24 as our primary development set (Dev2). We use the North American News Text Corpus (NANC) (Graff, 1995) as our source of unlabeled data. We segmented the NANC into sentences using Splitta (Gillick, 2009) and tokenized the sentences using a script supplied on the Penn Treebank Project website. Our segmentation of the NANC yielded approximately 20M sentences.

Our training data consists of the union of all evaluated samples in Section 2.3.1 and Section 2.3.3, as well as additional samples obtained similarly using 1M words and 5M words. In total, we had 444 sample points. The samples consisted of mean feature values of sets of sentences of various sizes and F_1 measures for parser models which

had been obtained through one step of self-training. I.e., the models were trained on the concatenation of one copy of the labeled training data and a subset of the baseline model parses of the NANC. We only used two features corresponding to sentence length and the number of novel bigrams. A bigram is considered novel if it does not appear in the sentences of the labeled training data.

Our regression target is labeled bracketing F_1 measure on development data. We used a radial basis function kernel with hyperparameters $C = 1.0$, $\gamma = 0.002$, $\epsilon = 0.1$. We did not perform any feature scaling. We used the Scikit-Learn implementation of support vector regression (Pedregosa et al., 2011). We then used the trained support vector regression model to predict self-trained F_1 for each sentence of the NANC, using the individual sentence features in the same manner as the sample feature means used during regression training. We then chose the top $100K$ and $5M$ sentences as semi-supervised data for self-training for evaluation.

Results

We report the evaluation set (WSJ23) labeled bracketing F_1 in Table 2.2. We also include results obtained by randomly sampling the same number of sentences from the available unlabeled data and evaluating a self-trained model on development data. At both levels evaluated, we obtained significant improvement from the baselines. Our best setup included 5M words with a 15x gold data upweighting to achieve an F_1 score of 92.52. We give additional details regarding the random baselines in Table 2.3.

Method	Sentences Added	F_1
Random	100K	90.64
SVR	100K	91.55
Random	5M	92.42
SVR	5M	92.52

Table 2.2: Results of SVM-based sample selection (using unknown bigram count and sentence length as features).

Sentences Added	Min	Mean	Max	stdDev	N
100K	90.50	90.64	90.78	0.0824	10
5M	92.37	92.42	92.45	0.0307	5

Table 2.3: Random Sampling Baseline for Self-trained Parsers

2.5 Discussion

Viewing sample selection as a problem of independently ranking the potential contributions of individual parses is challenging due to the batch evaluation nature of the task. Moreover, the learning utility contributions of individual parses are not truly independent. To see this, one can consider a training set consisting of arbitrarily many duplications of a single parse with positive utility. Moreover, the sum of the utilities of each parse is not bounded, while the actual utility of any set of parses is bounded above by the gap between the baseline parser score and 1. So it is clear that the true utility of a set of parses cannot accurately be modeled as a sum of independent values. Regardless, we have shown that a feature based approach can yield more accurate parsers than random sampling. We also obtained similar datasets with additional features, such as PCFG score, n-gram perplexity, and others; however, we did not find these features to be as strong predictors of utility as sentence length and bigram novelty, and therefore do not discuss them at length. We believe that the poor results of SVM regression with such features is due to overfitting of the training data with the higher dimensional sets.

The most practical conclusions of our work are that shorter sentences tend to be

harmful, and novel bigrams may be indicative of good utility. We have also established that there is significant variance in self-trained model F_1 so that a simple random restart with selection by F_1 on held out tuning data would result in improved parser performance with minimal effort. Our work obtains a new state of the art score (92.52) on a well-known and established data set (WSJ23) by selecting sentences of moderate length with many novel bigrams, and when using a traditional PCFG based parsing model.

In the future, we are most interested in exploring iterated self-training and how lessons learned during this work may be applied to bootstrapping other semi-supervised natural language processing models. The growing availability of extremely large text corpora such as CommonCrawl and Gigaword make it likely that there will continue be more data than can be feasibly used for training. This justfies careful sample selection for self-training.

3 Tree Transduction

In this chapter we present our tree to tree transduction language, TTT. We motivate the overall "template-to-template" approach to the design of the language, and outline its constructs, also providing some examples. We then show that TTT allows transparent formalization of rules for parse tree refinement and correction, logical form refinement and predicate disambiguation, inference, and verbalization of logical forms. Much of the text of this chapter was originally presented in the Proceedings of the Workshop on Applications of Tree Automata Techniques in Natural Language Processing (Purtee and Schubert, 2012).

3.1 Introduction

Pattern matching and pattern-driven transformations of list-structured symbolic expressions or trees are fundamental tools in AI. They facilitate many symbol manipulation tasks, including operations on parse trees and logical forms, and even inference and aspects of dialogue and translation.

The TTT system allows concise and transparent specification of rules for such tasks,

in particular (as we will show), parse tree refinement, parse tree correction, predicate disambiguation, logical form refinement, inference, and verbalization into English.

In parse tree refinement, our particular focus has been on repair of malformed parses of image captions, as obtained by the Charniak-Johnson parser (Charniak and Johnson, 2005). This has encompassed such tasks as distinguishing passive participles from past participles and temporal nominals from non-temporal ones, assimilation of verb particles into single constituents, deleting empty constituents, and particularizing prepositions. For example, standard treebank parses tag both past participles (as in "has written") and passive participles (as in "was written") as VBN. This is undesirable for subsequent compositional interpretation, as the meanings of past and passive participles are distinct. We can easily relabel the past participles as VBEN by looking for parse tree subexpressions where a VBN is preceded by a form of "have", either immediately or with an intervening adverb or adverbial, and replacing VBN by VBEN in such subexpressions. Of course this can be accomplished in a standard symbol manipulation language like Lisp, but the requisite multiple lines of code obscure the simple nature of the transduction.

We have also been able to repair systematic PP (prepositional phrase) misattachments, at least in the limited domain of image captions. For example, a common error is attachment of a PP to the last conjunct of a conjunction, where instead the entire conjunction should be modified by the PP. Thus when a statistically obtained parse of the sentence " Tanya and Grandma Lillian at her highschool graduation party" brackets as "Tanya and (Grandma Lillian (at her highschool graduation party.))", we want to lift the PP so that "at her highschool graduation party" modifies "Tanya and Grandma Lillian".

Another systematic error is faulty classification of relative pronouns/determiners

as wh-question pronouns/determiners, e.g., "the student *whose* mother contacted you" vs. "I know *whose* mother contacted you" – an important distinction in compositional semantics. (Note that only the first occurrence, i.e., the relative determiner, can be paraphrased as *with the property that his*, and only the second occurrence, in which *whose* forms a wh-nominal, can be paraphrased as *the person with the property that his*.) An important point here is that detecting the relative-determiner status of a wh-word like *whose* may require taking account of an arbitrarily deep context. For example, in the phrase "the student in front of whose parents you are standing", *whose* lies two levels of phrasal structure below the nominal it is semantically bound to. Such phenomena motivate the devices in TTT for detecting "vertical patterns" of arbitrary depth. Furthermore, we need to be able to make local changes "on the fly" in matching vertical patterns, because the full set of tree fragments flanking a vertical match cannot in general be saved using match variables. In the case of a wh-word that is to be re-tagged as a relative word, we need to rewrite it *at the point where the vertical pattern matches it*, rather than in a separate tree-(re)construction phase following the tree-matching phase.

An example of a discourse phenomenon that requires vertical matching is anaphoric referent determination. In particular, consider the well-known rule that a viable referent for an anaphoric pronoun is an NP that C-commands it, i.e., that is a (usually left) sibling of an ancestor of the pronoun. For example, in the sentence "John shows Lillian the snowman that he built", the NP for *John* C-commands the pronominal NP for *he*, and thus is a viable referent for it (modulo gender and number agreement). We will later show a simple TTT rule that tags such an anaphoric pronoun with the indices of its C-commanding NP nodes, thus setting the stage for semantic interpretation. We have been also able to perform Skolemization, conjunct separation, simple inference, and logical form verbalization with TTT and suspect its utility to logic tasks will increase

as development continues.

The rest of the chapter is as follows: we discuss related work in Section 3.2, discuss the TTT language (including pattern matching and transduction syntax, and some theoretical properties) in Section 3.4, and give several example applications in Section 3.5. The system can be found at *http://www.cs.rochester.edu/research/ttt/*.

3.2 Related Work

There are several pattern matching facilities available; however, none proved sufficiently general and perspicuous to serve our various purposes.

The three related tools Tgrep, Tregex, and Tsurgeon provide powerful tree matching and restructuring capabilities (Levy and Andrew, 2006). However, Tgrep and Tregex provide no transduction mechanism, and Tsurgeon's modifications are limited to local transformations on trees. Also, it presupposes list structures that begin with an atom (as in Treebank trees, but not in parse trees with explicit phrasal features), and its patterns are fundamentally tree traversal patterns rather than tree templates, and can be quite hard to read.

Peter Norvig's pattern matching language, "pat-match", from Norvig (1992) provides a nice pattern matching facility within the Lisp environment, allowing for explicit templates with variables (that can bind subexpressions or sequences of them), and including ways to apply arbitrary tests to expressions and to match boolean combinations of patterns. However, there is no provision for "vertical" pattern matching or subexpression replacement "on the fly". TTT supports both horizontal and vertical pattern matching, and both global (output template) and local (on the fly) tree transduction. Also the notation for alternatives, along with exclusions, is more concise than in

Norvig's matcher, for instance not requiring explicit ORs. While pat-match supports matching multi-level structures, the pattern operators are not composable – a feature present in TTT that we have found to be quite useful.

Mathematica also allows for sophisticated pattern matching, including matching of sequences and trees. It also includes an expression rewriting system that is capable of rewriting sequences of expressions. It provides functions to apply patterns to arbitrary subtrees of a tree until all matches have been found or some threshold count is reached, and it can return all possible ways of applying a set of rules to an expression. However, as in the case of Norvig's matcher there is no provision for vertical patterns or on-the-fly transduction (Wolfram Research, Inc., 2017).

Snobol, originally developed in the 1960's, is a language focused on string patterns and string transformations (Griswold, 1978). It has a notably different flavor to the other transformation systems. Its concepts of cursor and needle support pattern-based transformations that rely on the current position in a string at pattern matching time, and on the strings that were matched by preceding patterns up to the current point. Snobol also supports named and thereby recursive patterns. While it includes recognition of balanced parentheses, the expected data type for Snobol is the string – leaving it a less than direct tool for intricate manipulation of trees.

Haskell also includes a pattern matching system, but it is weaker than the other systems mentioned. The patterns are restricted to function arguments, and are not nearly as expressive as Mathematica's for trees nor Peter Norvig's system or Snobol for strings (Hudak et al., 1999).

The Tiburon tool is a comprehensive system for manipulating regular tree grammars, tree automata, and tree transducers, including weighted variants (May and Knight, 2008). It supports many useful algorithms, such as intersection, determinization, recog-

nition, top-k generation, and maximum likelihood training. However, variables that appear in both a rule's lhs and rhs must occur at a depth less than two on the left, and Tiburon cannot easily simulate our vertical path or sequence operators.

Timbuk is a system for deciding reachability with term rewriting systems and tree automata (Genet and Tong, 2003), and it also performs intersection, union, and determinization of tree automata. Though variables can appear at arbitrary locations in terms, they always match exactly one term from a fixed set, and therefore do not match sequences or vertical paths.

Xpath and XSLT are languages for manipulation of XML trees. As its name indicates, Xpath expressions describe paths in trees to the relevant nodes, rather than patterns representing the trees to be matched, as in the TTT approach. It is useful for extracting structured but unordered information from trees, and supports numerous functions and predicates over matched nodes, but does not match ordered sequences. XSLT is also more procedurally oriented than TTT, and is useful for constructing XML representations of transformations of data extracted by Xpath. The primary advantages of TTT over Xpath and XSLT are a more concise syntax, ordered sequence matching, compositional patterns and templates, and in-place modification of trees.

3.3 Formal Models of Tree Transduction

A tree transducer is a mapping from trees to trees. They are generalizations of string transducers, which are essentially string rewriting machines, and may be defined mathematically or practically (with code). Tree transducers have substantial application to the field of statistical machine translation, especially with respect to the syntax directed models, where they are used to transform a statistical parse of a source sentence

into either a string in a target language or even a complete parse of the corresponding sentence in the target language. In later sections, we will discuss the repair of malformed statistical parses with a tree transducer. The classic text on formal approaches to tree transducers (and tree automata - which only accept or reject trees) is Gécseg and Steinby (1984). A more modern treatment is given in Comon et al. (2007). A short but good introductory article is Knight (2007). Much has also been written by the machine translation community, especially the syntax directed MT sub-community, about tree transducers.

Preliminary Notation

A set of symbols Σ is called a *ranked alphabet* iff $\forall \sigma \in \Sigma, \exists n \in \mathbb{N}$ such that $rank(\sigma) = n$. For the most part, ranked alphabets can be assumed to be finite.

A *variable* is a member of a special set of symbols, such as $\mathcal{X}$, of rank zero. In this paper, variables will be denoted by small case italic letters from the end of the alphabet (e.g. *x*, *y*, *z*).

The representations of trees in the syntactic parsing and tree transduction communities are slightly different. A syntactic tree corresponding to an `S` token dominating an `NP` and `VP`, etc., is represented as `(S (NP (DET THE) (NN DOG)) (VP (VBZ RUNS)) (. .))`. The tree transduction community would represent such as a tree as `S(NP(DET(THE) NN(DOG)) VP(VBZ(RUNS)) .(.))`. The former (syntactic) notation is somewhat more general, in that it enables representation of trees where the annotation of internal nodes does not always consist of exactly one symbol, such as empty brackets. With this notation, strange trees (those without immediate linguistic interpretations) can be represented, such as `((A) (B))`.

For the purpose of discussing tree transducers, it is necessary to have a compact,

concise notation for sets of trees. We will use the notation $T(\Sigma)$ to denote the set of trees formed using only elements of the ranked alphabet Σ, where the leaves are restricted to be elements of rank zero. To denote trees where the leaves may optionally be from a set of variables $\mathcal{X}$, we will use the notation $T(\Sigma, \mathcal{X})$. For trees where the leaves are preceded by members of a state set Q, we will use the notations $T(\Sigma_Q)$ and $T(\Sigma_Q, \mathcal{X})$ respectively denoting trees without variables and trees which may include variables at the leaves. This notation is adapted from Comon et al. (2007).

Formal Models and Concepts

Broadly speaking, tree transducers come in two main varieties: top-down and bottom-up. The standard formal models of tree transducers all involve a monotonic propagation of state information, either from root to frontier (top-down) or from frontier to root (bottom-up). Conceivably, one could have a tree transducer which operates non-deterministically from any point within a tree, but this would only be considered a tree transducer from the "mapping from trees to trees" point of view, and might more properly be called a tree (or term) rewriting system. (Note that the automata style tree transducers always terminate, whereas a rewriting system may have infinitely long derivations.)

A top-down tree transducer is a five-tuple $T = (\Sigma, \Delta, Q_i, Q, P)$, where Σ is an input symbol alphabet, Δ is an output symbol alphabet, Q is a (finite) state set, $Q_i \subseteq Q$ is a start state set, and P is a set of productions. The alphabets Σ and Δ are ranked. That is, every symbol has a non-negative integer arity, symbols of positive arity correspond to internal nodes, and symbols with arity zero correspond to terminals or leaves. State symbols are typically assumed to have rank one (though not always, as in the case of MBOTs, which are discussed later).

Similarly, a bottom-up tree transducer is a five-tuple $B = (Q, \Sigma, \Delta, Q_f, P)$. The sets Q, Σ, and Δ are ranked alphabets as before, but instead of a start state set, bottom-up transducers have a set of final states, $Q_f \subseteq Q$. Bottom up transducers operate from frontier to root, propagating state information upward through the tree and transforming subtrees along the way.

Because a tree transducer defines mappings (possibly one-to-many) from trees to trees, it is reasonable to talk about the domain and range of any particular transducer, and the image(s) and inverse image(s) of particular trees. Transducers for which any tree has multiple images are said to be non-deterministic. For this to occur, there must exist multiple rules with identical left hand sides.

Tree transducers vary in their expressive power according to the various restrictions allowed on the rules. The primary dimensions are: vertical direction (e.g. bottom-up or top-down), whether rules are allowed to contain variables, whether rules are allowed to duplicate or delete variables, rule height, and non-determinism.

Varieties of Rules

Vertical Direction Going into more detail, top-down tree transducer rules have the general form $q(\sigma(x_1, \ldots, x_n)) \rightarrow \delta(q_i(x_i), \ldots, q_j(x_j))$, where each of the variables on the right side appear first on the left. Using the notation introduced earlier, top-down tree transducer rules are essentially elements of the set $\{(T(Q_\Sigma, \mathcal{X}), T(\Delta_Q, \mathcal{X}))\}$, subject to the above restriction that variables appearing in the second field also appear on the left. The class of top-down tree transducers will be referred to as **T**.

For example, $q(a(x_1, x_2)) \rightarrow b(q_1(x_1), q_2(x_2))$ is a top-down tree transducer rule. Both of the variables on the right hand side also occur on the left-hand side. A state appears immediately before any variable symbols on the right hand side. This rule

corresponds to the action that, if the transducer is in state q and encounters a subtree of rank two headed by the symbol a, then it will rewrite the subtree as a subtree headed by the symbol b, with the previous two children left unaltered, and proceed to process the left child in state q_1 and the right child in state q_2.

Similarly, bottom-up transducer rules have the general form $\sigma(q_1(x_1), \ldots, q_n(x_n)) \rightarrow q'(\delta_i(x_i), \ldots, \delta_j(x_j))$, where the variables appearing on the right hand side of the production are restricted to be among those which also appear on the left. These rules are elements of the set $\{(T(\Sigma_Q, \mathcal{X}), T(Q_\Delta, \mathcal{X}))\}$. The class of bottom-up tree transducers will be referred to as **B**.

Multi-bottom up transducers A novel type of bottom-up transducer is the Multi-bottom-up transducer (MBOT). MBOTs allow the state symbols to have rank greater than one. Maletti (2010) credits Arnold and Dauchet (1982) and Lilin (1981) with MBOTs. Maletti (2010) argues that multi-bottom-up tree transducers are an ideal formalism for machine translation, because they are as powerful as synchronous tree substitution grammars but have the computational advantage of being composable. Gildea (2012) however argues that parsing with MBOTs which result from compositions of STSGs remains computationally intractable, so MBOTs are not perfect replacement for STSGs.

Copying Rules A rule is said to be copying (or non-linear) if any variable symbol appears more than once on the right hand side. Left hand side variables are assumed to be distinct within each rule. Tree transducers which do not use any copying rules are referred to as linear. Their classes are **LT** and **LB**, for linear top-down transducers and linear bottom-up transducers respectively. Some linear transducers have convenient theoretical properties, which will be discussed in subsequent sections.

A top-down copying rule: $q(\sigma(x)) \to \delta(q'(x), q'(x))$

A bottom-up copying rule: $\sigma(q(x)) \to q'\delta(x, x)$

Deleting Rules Rules where a variable appearing in the left hand side does not appear in the right hand side are called deleting. Note that it is possible for fixed tree symbols to be removed by a rule without the rule being a deleting one. Only rules which can delete arbitrary (variable) tree fragments are deleting. Tree transducers which do not use any deleting rules are referred to (bluntly) as non-deleting. Their classes are **NT** and **NB**. Transducers which are both linear and nondeleting are referred to as **LNT** and **LNB**.

A top-down deleting rule: $q(\sigma(x, y)) \to \delta(q'(x))$

A bottom-up deleting rule: $\sigma(q_1(x), q_2(y)) \rightarrow q'(\delta(x))$

Extended rules As with the previous two properties, both bottom-up and top-down tree transducers can be augmented with extended rules. Typically, when discussing extended tree transducer rules, one means that the left hand side can involve arbitrarily tall tree fragments, specifying multiple levels of structure. When more precision is necessary, it is possible to discuss extended-lhs and extended-rhs rules independently. However, it seems to be a common assumption that rhs rules may be extended even when dealing with tree transducer classes that are not themselves referred to as "extended". Adding extended (lhs) rules strictly increases the power of **T**, **LT**, and **LNT** transducers (Knight, 2007; Maletti, 2008). Class names for extended transducers have an "x" prepended, as **xLNT**, **xLNB**, **xT**, **xB**, ...

A top-down extended lhs rule: $q(\sigma_1(x, \sigma_2(y))) \rightarrow \delta(q'(x, y))$

A bottom-up extended lhs rule: $\sigma_1(q_1(x), \sigma_2(q_2(y))) \rightarrow q'(\delta(x, y))$

Synchronous Grammars

The notion of synchronous grammars as tree transducers was investigated by Shieber (2004). Synchronous grammars simultaneously generate pairs of strings, often in separate languages. A synchronous grammar transforms one sentence into another if there exists a simultaneous derivation. Similarly, a synchronous grammar can be viewed as a transducer of one tree into another when there exist simultaneous derivations. The three main types of synchronous grammars are synchronous context-free (SCFG), synchronous tree-substitution (STSG), and synchronous tree adjoining grammars (STAG). SCFG is slightly less powerful than the class **T** (top-down tree transducers, non-linear and deleting rules allowed), because the derivations are not directed by a state which

is separate from the structure of the tree. STSG is more powerful than SCFG, but less powerful than **xT**. The parallel between substitution of tree fragments with non-terminal leaves for extended tree transducer rules is clear. STAG includes the "adjoin" operation, which injects a tree fragment. Note that the languages formed by vertical paths from root to frontier of SCFG and STSG derivations are always regular, whereas the languages formed by STAG derivations can be context-free, so that STAG is more powerful than STSG.

Training

Graehl and Knight (2004) derive an algorithm to infer rule weights for a **xT** transducer so that the joint probability of a set of transformations (set of input/output trees) is maximized. Their rule learning procedure takes as input a **xT** transducer with a fixed set of rules, and a fixed set of pairs of input/output trees. The rule weights are adjusted via expectation maximization, where the expected rule counts are computed for each tree pair by intersecting the pair with the transducer to produce a weighted regular tree grammar (RTG). The corresponding language generated consists of all derivation trees which transform the observed input into the observed output. The nodes of these derivation trees are annotated with the rules at each step of the derivations. For the details of constructing these derivation grammars, the reader is referred to the work by Graehl and Knight (2004).

Rule Extraction

The training algorithm of Graehl and Knight (2004) assumes that the relevant rules are known a priori; however, one may conceivably have a set of tree pairs without accompanying tree transducer rules for which one desires to learn an optimal transducer.

Several researchers have discussed extraction of synchronous context-free rules, and some extraction of synchronous tree-substitution grammar rules. See papers Galley et al. (2004); Zhang et al. (2008a); Gildea and Štefankovič (2007); Post and Gildea (2009). And additional refs: (Chiang, 2010; Eidelman et al.; Mylonakis and Sima'an, 2010; Čmejrek and Zhou, 2010; Sankaran et al., 2011; Lavie et al., 2008; Blunsom et al., 2009)

Rewriting Systems

Rewriting systems are more general than tree transducers, because they allow multiple passes. (That is, there is no strict root-to-frontier or frontier-to-root ordering of rule applications).

3.4 The TTT Language

3.4.1 Pattern Matching

Patterns in TTT are hierarchically composed of sub-patterns. The simplest kind of pattern is an arbitrary, explicit list structure (tree) containing no match operators, and this will match only an identical list structure. Slightly more flexible patterns are enabled by the "underscore operators" `_!`, `_+`, `_?`, `_*`. These match any single tree, any non-empty sequence of trees, the empty sequence or a sequence of one tree, and any (empty or non-empty) sequence of trees respectively. These operators (as well as all others) can also be thought of as match variables, as they pick up the tree or sequence of trees they match as their binding.

The bindings are "non-sticky", i.e., an operator such as `_!` will match any tree,

causing replacement of any prior binding (within the same pattern) by that tree. However, bindings can be preserved in two ways: by use of new variable names, or by use of sticky variables. New variable names are obtained by appending additional characters – conventionally, digits – to the basic ones, e.g., `_!1`, `_!2`, etc. Sticky variables are written with a dot, i.e., `_!.`, `_+.`, `_?.`, `_*.`, where again these symbols may be followed by additional digits or other characters. The important point concerning sticky variables is that multiple occurrences of such a variable in a pattern can only be bound by the same unique value. Transductions are specified by a special pattern operator `/` and will be described in the next section.

More flexible operators, allowing for alternatives, negation, and vertical patterns among other constructs, are written as a list headed by an operator without an underscore, followed by one or more arguments. For example, `(! A (B C))` will match either the symbol `A` or the list `(B C)`, i.e., the two arguments provide alternatives. As an example involving negation, `(+ A (B _!)` $\sim$ `(B B))` will match any nonempty sequence whose elements are `A`s or two-element lists headed by `B`, but disallowing elements of type `(B B)`. Successful matches cause the matched expression or sequence of expressions to become the value of the operator. Again, sticky versions of match operators use a dot, and the operators may be extended by appending digits or other characters.

The ten basic argument-taking pattern operators are:

! Match exactly one sub-pattern argument.

+ Match a sequence of one or more arguments.

? Match the empty sequence or one argument.

* Match the empty sequence or one or more arguments.

{} Match any permutation of the arguments.

<> Match the sequence of arguments directly (without the parens enclosing the <> operator)

^ Match a tree that has a child matching one of the arguments.

^* Match a tree that has a descendant matching one of the arguments.

^@ Match a vertical path.

/ Attempt a transduction. (Explained later.)

Various examples will be provided below. Any of the arguments to a pattern operator may be composed of arbitrary patterns.

Negation: The operators `!`, `+`, `?`, `*`, and `^` support negation (pattern exclusion); i.e., the arguments of these operators may include not only alternatives, but also a negation sign ∼ (after the alternatives) that is immediately followed by one or more precluded patterns. If no alternatives are provided, only precluded patterns, this is interpreted as "anything goes", except for the precluded patterns. For example, `(+ ∼ (A A) (B B))` will match any nonempty sequence of expressions that contains no elements of type `(A A)` or `(B B)`. Note that the negation operator does not appear by itself; one must instead specify it in conjunction with one of the other operators. The pattern `(! ∼ P)` matches any single tree which does not match pattern `P`.

Conjunction: We have so far found no compelling need for an explicit conjunction operator. Of course, any pattern calling for a structured tree is by its nature conjunctive – all the tree components called for must be present. If necessary, a way to say that a tree must match each of two or more patterns is to use double negation. For example, suppose we want to say that an expression must begin with an `A` or `B` but must contain

an `A` (at the top level); this could be expressed as

`(! ~ (! ~ ((! A B) _*) (_* A _*))).`

However, this would be more perspicuously expressed in terms of alternatives, i.e.,

`(! (A _*) (B _* A _*)).`

We also note that the allowance for computable predicates (discussed below) enables introduction of a simple construct like

`(! (and? patt1 patt2))`,

where `patt1` and `patt2` are arbitrary TTT patterns, and `and?` is an executable predicate that applies the TTT matcher to its arguments and returns a non-nil value if both succeed and nil otherwise. In the former case, the binding of the outer `!` will become the matched tree.

Bounded Iteration: The operators `!`, `+`, `?`, `*`, and `^` also support bounded iteration, using square brackets. This enables one to write patterns that match exactly n, at least n, at most n, or from n to m times, where n and m are integers. Eg. `(![3] A)` would match the sequence `A A A`. The vertical operator `^[n]` matches trees with a depth-`n` descendant that matches one of the operator's arguments.

Vertical Paths: The operators `^*` and `^@` enable matching of vertical paths of arbitrary depth. The first, as indicated, requires the existence of a descendant of the specified type, while the second, with arguments such as `(^@` P_1 P_2 `...` P_n`)` matches a tree whose root matches P_1, and has a child matching P_2, which in turn has a child matching P_3, and so on. Note that this basic form is indifferent to the point of attachment of each successive offspring to its parent; but we can also specify a point of attachment in any of the P_1, P_2, etc., by writing `@` for one of its children. Because this operator (`@`) does not appear outside the vertical path context, it was not listed with the other operators above. Note as well that the argument sequence P_1 P_2 `...` can

Pattern	Tree	Bindings
`_!`	`(A B C)`	`(_! (A B C)`
`(A _! C)`	`(A B C)`	`(_! B)`
`(_* F)`	`(A B (C D E) F)`	`(_* A B (C D E))`
`(A B _? F)`	`(A B (C D E) F)`	`(_? (C D E))`
`(A B _? (C D E) F)`	`(A B (C D E) F)`	`(_?)`
`(^@ _! (C _*) E)`	`(A B (C D E) F)`	`(^@ (A B (C D E) F)) (_* D E)`
`(A B (<> (C D E)) F)`	`(A B (C D E) F)`	`(<> (C D E))`
`(A B (<> C D E) F)`	`(A B (C D E) F)`	`nil`

Table 3.1: Example Bindings for TTT Variables

itself be specified as a pattern (e.g., via `(+ ...)`), and in this case there is no advance commitment to the depth of the tree being matched.

Computable Predicates: Arbitrary predicates can be used during the pattern matching process (and consequently the transduction process). Symbols with names ending in a question mark, and with associated function definitions, are interpreted as predicates. When a predicate is encountered during pattern matching, it is called with the current subtree as input. The result is nil or non-nil, and when nil is returned the current match fails, otherwise it succeeds (but the non-nil value is not used further). Additionally, supporting user-defined predicates enables the use of named patterns.

Some Example Patterns: Here are examples of particular patterns, with verbal explanations. Also see Table 1, at the top of the next page, for additional patterns with example bindings.

- `(! (+ A) (+ B))`

  Matches a non-empty sequence of `A`'s or a non-empty sequence of `B`'s, but not a sequence containing both.

- `(* (<> A A))`

  Matches an even number of `A`'s.

- `(B (* (<> B B)))`

  Matches an odd number of `B`'s.

- `(({} A B C))`

  Matches `(A B C)`, `(A C B)`, `(B A C)`, `(B C A)`, `(C A B)` and `(C B A)` and nothing else.

- `((<> A B C))`

  Matches `(A B C)` and nothing else.

- `(^* X)`

  Matches any tree that has descendant `X`.

- `(^@ (+ (@ _*)) X)`

  Matches any tree with leftmost leaf `X`.

3.4.2 Transductions

Transductions are specified with the transduction operator, `/`, which takes two arguments. The left argument may be any tree pattern and the right argument may be constructed of literals, variables from the lhs pattern, and function calls.

Transductions may be applied to the roots of trees or arbitrary subtrees, and they may be restricted to apply at most once, or until convergence. When applying transductions to arbitrary subtrees, trees are searched top-down, left to right. When a match to the transduction lhs pattern occurs, the resulting bindings and transduction rhs are used to create a new tree, which then replaces the tree (or subtree) that matched the lhs.

Here are a few examples of simple template to template transductions:

- `(/ X Y)`

  Replaces the symbol `X` with the symbol `Y`.

- `(/ (! X Y Z) (A))`

  Replaces any `X`, `Y`, or `Z` with `A`.

- `(/ (! X) (! !))`

  Duplicates an `X`.

- `(/ (X _* Y) (X Y))`

  Remove all subtrees between `X` and `Y`.

- `(/ (_! _* _!1) (_!1 _* _!))`

  Swaps the subtrees on the boundaries.

A transduction operator may appear nested within a composite pattern. The enclosing pattern effectively restricts the context in which the transduction will be applied, because only a match to the entire pattern will trigger a transduction. In this case, the transduction is applied at the location in the tree where it matches. The rhs of such a transduction is allowed to reference the bindings of variables that appear in the enclosing pattern. We call these local transductions, as distinct from replacement of entire trees. Local transductions are especially advantageous when performing vertical path operations, allowing for very concise specifications of local changes. For example, the transduction

```
(^@ (* ((! S SBAR) _+))
    (/ (WH _!)
       (REL-WH (WH _!))))
```

wraps `(REL-WH ...)` around a `(WH ...)` constituent occurring as a descendant of a vertical succession of clausal (`S` or `SBAR`) constituents. Applied to the tree

`(S (SBAR (WH X) B) A)`, this yields the new tree `(S (SBAR (REL-WH (WH X)) B) A)`. Additional examples appear later (especially in the parse tree refinement section).

TTT also supports constructive functions, with bound variables as arguments, in the rhs templates, such as `join-with-dash!`, which concatenates all the bound symbols with intervening dashes, and `subst-new!`, which will be discussed later. One can imagine additional functions, such as `reverse!`, `l-shift!`, `r-shift!`, or any other function of a list of nodes that may be useful to the application at hand. Symbols with names ending in the exclamation mark are assumed to be associated with function definitions, and when appearing as the first element of a list are executed during output template construction. To avoid writing many near-redundant functions, we use the simple function `apply!` to apply arbitrary Lisp functions during template construction.

3.4.3 Theoretical Properties

A thorough treatment of the formal properties of tree transducers is Comon et al. (2007). A good overview of the dimensions of variability among formal tree transducers is given in Knight (2007). The main properties are restrictions on the height of the tree fragments allowed in rules, linearity, and whether the rules can delete arbitrary subtrees. Among the more popular and recent ones, synchronous tree substitution grammars (STSG), synchronous tree sequence substitution grammars (STSSG), and multi bottom-up tree transducers (MBOT) constrain their rules to be linear and non-deleting, which is important for efficient rule learning and transduction execution (Chiang, 2004; Galley et al., 2004; Yamada and Knight, 2001; Zhang et al., 2008b; Maletti, 2010).

The language TTT does not have any such restrictions, as it is intended as a gen-

eral programming aid, with a concise syntax for potentially radical transformations, rather than a model of particular classes of linguistic operations. Thus, for example, the 5-element pattern `(! ((* A) B) ((* A) C) ((* A) D) ((* A) E) ((* A)))` applied to the expression `(A A A A A)` rescans the latter 5 times, implying quadratic complexity. (Our current implementation does not attempt regular expression reduction for efficient recognition.) With the addition of the permutation operator {}, we can force all permutations of certain patterns to be tried in an unsuccessful match (e.g., `(({} (! A B C) (! A B C) (! A B C)))` applied to `(C B E)`), leading to exponential complexity. (Again, our current implementation does not attempt to optimize.) Also, allowance for repeated application of a set of rules to a tree, until no further applications are possible, leads to Turing equivalence. This of course is true even if only the 4 underscore-operators are allowed: We can simulate the successive transformations of the configurations of a Turing machine with string rewriting rules, which are easily expressed in terms of those operators and `/`. Additionally, pattern predicates and function application in the right-hand sides of rules are features present in TTT that are not included in the above formal models. In themselves (even without iterative rule application), these unrestricted predicates and functions lead to Turing equivalence.

The pattern operator set was chosen so that a number of disparate pattern matching programs could all be replaced with concise TTT rules. It does subsume regular tree expressions and can therefore be used to match any regular tree language. Specifically, alternation can be expressed with `!` and (vertical) iteration with `^@` and `*`. The example expression from Comon et al. (2007) can be specified as `(^@ (* (cons 0 @)) nil)`, which matches Lisp expressions corresponding to lists of zero or more zeros. TTT also differs from standard tree automata by lack of an explicit state.

Nondeterminism and noncommutativity: In general, given a set of transductions (or even a single transduction) and an input tree there may be several ways to apply the transductions, resulting in different trees. This phenomenon comes from three sources:

- Rule application order - transductions are not in general commutative.
- Bindings - a pattern may have many sets of consistent bindings to a tree (e.g., pattern `(_* _*1)` can be bound to the tree `(X Y Z)` in four distinct ways).
- Subtree search order - a single transduction may be applicable to a tree in multiple locations (e.g., `(/ _! X)` could replace any node of a tree, including the root, with a single symbol).

Therefore some trees may have many reduced forms with respect to a set of transductions (where by reduced we mean a tree to which no transductions are applicable) and even more reachable forms.

Our current implementation does not attempt to enumerate possible transductions. Rather, for a given tree and a list of transductions, each transduction (in the order given) is applied in top-down fashion at each feasible location (matching the lhs), always using the first binding that results from this depth-first, left-to-right (i.e., pre-order) search. Our assumption is that the typical user has a clear sense of the order in which transformations are to be performed, and is working with rules that do not interact in unexpected ways. For example, consider the cases of PP misattachment mentioned earlier. In most cases, such misattachments are disjoint (e.g., consider a caption reading "John and Mary in front and David and Sue in the back", where both PPs may well have been attached to the proper noun immediately to the left, instead of to the appropriate conjunction). It is also possible for one rule application to change the context of another, but this is not necessarily problematic. For instance, suppose that in the sentence "John

drove the speaker to the airport in a hurry" the PP "to the airport" has been misattached to the NP for "the speaker" and that the PP "in a hurry" has been misattached to the NP for "the airport". Suppose further that we have a repair rule that carries a PP attached to an NP upward in the parse tree until it reaches a VP node, reattaching the PP as a child of that VP. (The repair rule might incorporate a computable predicate that detects a poor fit between an NP and a PP that modifies it.) Then the result will be the same regardless of the order in which the two repairs are carried out. The difference is just that with a preorder discipline, the second PP ("in a hurry") will move upward by one step less than if the order is reversed, because the first rule application will have shortened the path to the dominating VP by one step.

In future it may be worthwhile to implement exhaustive exploration of all possible matches and expression rewrites, as has been done in Mathematica. In general this would call for lazy computation, since the set of rewrites may be an infinite set.

3.5 Example Applications

3.5.1 Linguistic

Parse Tree Refinement: First, here is a simple transduction to delete nil constituents (i.e., empty brackets), which sometimes occur in the Brown corpus:

```
(/ (_* () _*1) (_* _*1))
```

To distinguish between past and passive participles, we want to search for the verb *have*, and change the participle token correspondingly, as discussed earlier. The following two transductions are equivalent – the first is global and the second is an example of a local or on-the-fly transduction. For simplicity we consider only the *has* form of *have*. Observe the more concise form, and simpler variable specifications of the second

transduction.

```
(/
(VP _* (VBZ HAS) _*1 (VBN _!)  _*2)
(VP _* (VBZ HAS) _*1 (VBEN _!)  _*2))

(VP _* (VBZ HAS) _*
((/ VBN VBEN) _!)  _*)
```

To distinguish temporal and non-temporal nominals, we use a computable predicate to detect temporal nouns, and then annotate the NP tag accordingly. (One more time, we show global and local variants.)

```
(/ (NP _* nn-temporal?)
   (NP-TIME _* nn-temporal?))

((/ NP NP-TIME) _* nn-temporal?)
```

Assimilation of verb particles into single constituents is useful to semantic interpretation, and is accomplished with the transduction:

```
(/ (VP (VB _!1)
       ({} (PRT (RP _!2)) (NP _*1))
  (VP (VB _!1 _!2) (NP _*1)))
```

We often particularize PPs to show the preposition involved, e.g., PP-OF, PP-FROM, etc. Note that this transduction uses the `join-with-dash!` function, which enables us to avoid writing a separate transduction for each preposition:

```
(/ (PP (IN _!)  _*1)
    ((join-with-dash! PP _!)
     (IN _!)  _*1))
```

Such a rule transforms subtrees such as `(PP (IN FROM))` by rewriting the PP tag as `(PP-FROM (IN FROM)`.

As a final syntactic processing example (transitioning to discourse phenomena and semantics), we illustrate the use of TTT in establishing potential coreferents licensed by C-command relations, for the sentence mentioned earlier. We assume that for reference purposes, NP nodes are decorated with a SEM-INDEX feature (with an integer value), and pronominal NPs are in addition decorated with a CANDIDATE-COREF feature, whose value is a list of indices (initially empty). Thus we have the following parse structure for the sentence at issue (where for understandabilty of the relatively complex parse tree we depart from Treebank conventions not only in the use of some explicit features but also in using linguistically more conventional phrasal and part-of-speech category names; R stands for relative clause):

```
(S ((NP SEM-INDEX 1) (NAME John))
   (VP (V shows)
       ((NP SEM-INDEX 2) (NAME Lillian))
       ((NP SEM-INDEX 3) (DET the)
        (N (N snowman)
           (R (RELPRON that)
              ((S GAP NP)
               ((NP SEM-INDEX 4
                 CANDIDATE-COREF ())
                (PRON he))
               ((VP GAP NP) (V built)
                ((NP SEM-INDEX 4)
                 (PRON *trace*)))))))))
```

Here is a TTT rule that adjoins the index of a C-commanding NP node to the

CANDIDATE-COREF list of a C-commanded pronominal NP:

```
(_*
 ((NP _* SEM-INDEX _!.  _*) _+)
 _*
 (^* ((NP _* CANDIDATE-COREF
            (/ _! (adjoin! _!.  _!))                _*)
      (PRON _!)))
 _*)
```

The NP on the second line is the C-commanding NP, and note that we are using a sticky variable '`_!.`' for its index, since we need to use it later. (None of the other match variables need to be sticky, and we reuse '`_*`' and '`_!`' multiple times.) The key to understanding the rule is the constituent headed by '`^*`', which triggers a search for a (right) sibling or descendant of a sibling of the NP node that reaches an NP consisting of a pronoun, and thus bearing the CANDIDATE-COREF feature. This feature is replaced "on the fly" by adjoining the index of the C-commanding node (the value of '`_!.`') to it. For the sample tree, the result is the following (note the value '`(1)`' of the CANDIDATE-COREF list):

```
(S ((NP SEM-INDEX 1) (NAME John))
   (VP (V shows)
       ((NP SEM-INDEX 2) (NAME Lillian))
       ((NP SEM-INDEX 3) (DET the)
        (N (N snowman)
           (R (RELPRON that)
              ((S GAP NP)
               ((NP SEM-INDEX 4
                 CANDIDATE-COREF (1))
                (PRON he))
               ((VP GAP NP) (V built)
                ((NP SEM-INDEX 4)
                 (PRON *trace*)))))))))
```

Of course, this does not yet incorporate number and gender checks, but while these

could be included, it is preferable to gather candidates and heuristically pare them down later. Thus repeated application of the rule would also add the index 2 (for *Lillian*) to CANDIDATE-COREF.

3.5.2 Logical

Skolemization: Skolemization of an existential formula of type `(some x R S)`, where `x` is a variable, `R` is a restrictor formula and `S` is the nuclear scope, is performed via the transduction

```
(/ (some _! _!1 _!2)
    (subst-new!
              _!
              (_!1 and.cc _!2))).
```

The function `subst-new!` replaces all occurrences of a free variable symbol in an expression with a new one. (We assume that no variable occurs both bound and free in the same expression.) It uses a TTT transduction to accomplish this. For example, `(some x (x politician.n) (x honest.a))` becomes `((C1.skol politician.n) and.cc (C1.skol honest.a)).`

Inference: We can use the following rule to accomplish simple default inferences such as that if most things with property `P` have property `Q`, and most things with property `Q` have property `R`, then (in the absence of knowledge to the contrary) many things with property `P` also have property `R`. (Our logical forms use infix syntax for predication, i.e., the predicate follows the "subject" argument. Predicates can be lambda abstracts, and the computable boolean function `pred?` checks for arbitrary predicative constructs.)

```
(/ (_*
```

```
    (most _!.1 (_!.1 (!.p pred?))
               (_!.1 (!.q pred?)))
    _*
    (most _!.2 (_!.2 !.q)
               (_!.2 (!.r pred?)))
    _*)
   (many _!.1 (_!.1 !.p)
              (_!.1 !.r)))
```

For example,

```
((most x (x dog.n) (x pet.n))
 (most y (y pet.n) (x friendly.a)))}
```

yields the default inference

```
(many (x dog.n) (x friendly.a)).
```

The assumption here is that the two *most*-formulas are embedded in a list of formulas (selected by the inference algorithm), and the three occurrences of `_*` allow for miscellaneous surrounding formulas. (To allow for arbitrary ordering of formulas in the working set, we also provide a variant with the two *most*-formulas in reverse order.) Inference with tree transduction rules has also been performed by Koller and Thater (2010).

Predicate Disambiguation: The following rules are applicable to patterns of predication such as `((det dog.n have.v (det tail.n))`, `((det bird.n have.v (det nest.n))`, and `((det man.n) have.v (det accident.n))`. (Think of `det` as an unspecified, unscoped quantifier.) The rules simultaneously introduce plausible patterns of quantification and plausible disambiguations of the various senses of `have.v` (e.g., have as part, possess, eat, experience):

```
(/ ((det (! animal?)) have.v
    (det (!1 animal-part?)))
   (all-or-most x (x !)
    (some e ((pair x e) enduring)
     (some y (y !1)
      ((x have-as-part.v y) ** e)))))

(/ ((det (! agent?)) have.v
    (det (!1 possession?)))
   (many x (x !)
    (some e
     (some y (y !1)
      (x possess.v y) ** e))))

(/ ((det (! animal?)) have.v
    (det (!1 food?)))
   (many x (x !)
    (occasional e
     (some y (y !1)
      (x eat.v y) ** e))))

(/ ((det (! person?)) have.v
    (det (!1 event?)))
   (many x (x !)
    (occasional e
     (some y (y !1)
      ((x experience.v y) ** e)))))
```

Computable predicates such as `animal?` and `event?` are evaluated with the help of WordNet and other resources. Details of the logical form need not concern us, but it should be noted that the '`**`' connects sentences to events they characterize much as

in various other theories of events and situations.

Thus, for example, `((det dog.n have.v (det tail.n))` is mapped to:

```
(all-or-most x (x dog.n
 (some e ((pair x e) enduring)
  (some y (y tail.n)
   ((x have-as-part.v y) ** e)))))
```

This expresses that for all or most dogs, the dog has an enduring attribute (formalized as an agent-event pair) of having a tail as a part.

Logical Interpretation: The following transductions directly map some simple parse trees to logical forms. The rules, applied as often as possible to a parse tree, replace all syntactic constructs, recognizable from (Treebank-style) phrase headers like `(NN ...)`, `(NNP ...)`, `(JJ ...)`, `(NP ...)`, `(VBD ...)`, `(VP ...)`, `(S ...)`, etc., by corresponding semantic constructs. For example, "The dog bit John Doe", parsed as

```
(S (NP (DT the) (NN dog))
   (VP (VBD bit)
       (NP (NNP John) (NNP Doe))))
```

yields

```
(the x (x dog.n)
       (x bit.v John_Doe.name)).
```

Type-extensions such as '`.a`', '`.n`', and '`.v`' indicate adjectival, nominal, and verbal predicates, and the extension '`.name`' indicates an individual constant (name);

these are added by the functions `make-adj!`, `make-noun!`, and so on. The fourth rule below combines two successive proper nouns (NNPs) into one. We omit event variables, tense and other refinements.

```
(/ (JJ _!)  (make-adj! _!))
(/ (NN _!)  (make-noun! _!))
(/ (VBD _!)  (make-verb! _!))
(/ (_*.a (NNP _!.1) (NNP _!.2) _*.b)
   (_*.a (NNP _!.1 _!.2) _*.b))
(/ (NNP _+) (make-name! (_+)))
(/ (NP _!)  _!)
(/ (S (NP (DT the) _!)  (VP _+)) (the x (x _!)  (x _+))
```

These rules are illustrative only, and are not fully compositional, as they interpret an NP with a determiner only in the context of a sentential subject, and a VP only in the context of a sentential predicate. Also, by scoping the variable of quantification, they do too much work at once. A more general approach would use compositional rules such as `(/ (S (!1 NP?) (!2 VP?)) ((sem! !1) (sem! !2)))`, where the `sem!` function again makes use of TTT, recursively unwinding the semantics, with rules like the first five above providing lexical-level `sem!-values`.

We have also experimented with rendering logical forms back into English, which is rather easier, mainly requiring dropping of variables and brackets and some reshuffling of constituents.

3.6 Chapter Conclusion

The TTT language is well-suited to the applications it was aimed at, and is already proving useful in current syntactic/semantic processing applications. It provides a very concise, transparent way of specifying transformations that previously required extensive symbolic processing. Some remaining issues are efficient access to, and deployment of, rules that are locally relevant to a transduction; and heuristics for executing matches and transductions more efficiently (e.g., recognizing various cases where a complex rule cannot possibly match a given tree, because the tree lacks some constituents called for by the rule; or use of efficient methods for matching regular-expression subpatterns).

The language also holds promise for rule-learning, thanks to its simple template-to-template basic syntax. The kinds of learning envisioned are learning parse-tree repair rules, and perhaps also LF repair rules and LF-to-English rules.

4 Logical Reasoning and Probability

4.1 Chapter Introduction

We now transition to our work on knowledge representation where we discuss the problem of evidence combination. As an under-constrained task, any concrete estimates or actions taken as a result of multiple kinds of evidence[1] must be made on the basis of some set of assumptions. Identifying the exact set of assumptions poses a significant challenge. In cases where our domain knowledge allows us to constrain the problem, we can find a natural, parametric way of representing symbolic logical knowledge together with commonsense rules for probabilistic reasoning.

We are particularly interested in the Bayesian perspective. We also consider entropy based methods and the interval-valued probabilities of the Dempster-Shafer framework. The primary contribution of this chapter – and this dissertation overall – is our set of rules for probabilistic reasoning. These rules are a marriage of modal quantifiers (Schubert and Hwang, 2000) with an algebraic probability framework. The quantifiers are developed in the context of Episodic Logic, and the probability framework is de-

[1] These multiple sources of evidence may be corroborative, contradictory, and ambiguous all at once – though they are almost never complete.

scribed with "noisy" boolean connectives. (Schubert, 2004). This approach is inspired by related techniques where Bayesian networks are generalized from propositional to (somewhat) first-order domains.[2] Advantages of our approach are that we retain a full proof theory, our inference algorithms are lifted, and our numerical parameters correspond to simple conditional probabilities.

Portions of this chapter were originally presented at the 2016 Argument Strength workshop in Bochum, Germany and the 2017 Proceedings of the Advances in Cognitive Systems conference, in Troy, NY.

4.2 Logic as Knowledge Representation

Symbolic logic as a stand-in representation for knowledge expressed in natural language has several advantages. It is unambiguous, concise, and perspicuous. It lends itself well to general problems, such as question answering, planning, and story understanding. It is a well-established field with significant literature going back as far as the 1880s with the work of Frege in the case of symbolic logic, and modern proof techniques of unification-based resolution of the 1960s. The study of logic is of course even much older than this, with some of the earliest scholarly work on representation and inference recorded by Aristotle over two thousand years ago.

Despite this long history, the expressiveness of symbolic logic falls far short of natural language. Some concepts and idioms which are simple in language are not easily expressed in first order logic, and the consequences of those which are, may be difficult to prove.

[2] Discussed in Section 4.5.

4.2.1 First-Order Logic

First order logic (FOL) is a formal system of predicates, terms, variables, quantifiers, functions, and connectives. Commonly accepted model theoretic semantics include specification of a domain of discourse and an interpretive mapping between subsets of the domain and the non-logical symbols of the language corresponding to particular predicates, terms, and functions.

Truth of predications coincides with simple set membership. Truth of compound logical formulas is decided by a set of semantic rules specified in tandem with the syntactic grammar of the language. Universally quantified formulas hold when any substitution of domain members for their variables yields a true formula. Existential quantifications are true when the post-substitution formula holds for any member of the domain, rather than all. Negation, conjunction, and disjunction are straightforward. Material implication is by decree only false when the antecedent is true and the consequent false, leading to strange logical "truths" such as "If the moon is made out of cheese, then dogs can fly." When an implication is tautological, then the consequent is said to be a logical consequent of the antecedent. When a given formula is always true given that another set of formulas is true, then it is said to be entailed by those formulas. Tautological material implications coincide with entailment of the consequent by the antecedent.

In small theories, a domain may be explicitly enumerated. In others, such as general commonsense reasoning systems, the domain of discourse would ideally be taken to align with anything which could be a subject of human discourse – i.e., everything in the universe. Even relatively concise first-order theories of arithmetic involve infinite domains. Note that infinite domains do not require infinite specification – functional terms such as *Successor(x)* may be used to denote other terms without explicit enumer-

ation.

Logical inference with first-order knowledge bases is only semi-decidable. In the cases where entailment does not hold, any sound and complete proof method may be subject to infinite, non-terminating loops or recursion. Parallel proof attempts may be used to find both a formula and its negation; however, in many cases neither a formula or its negation will be entailed by a knowledge base. Therefore, semi-decidability of full first order theories is unavoidable.

In cases where an entailment relationship does hold between a formula and a set of formulas, then Herbrand's theorem assures us that a finite proof exists, we only have to find it. Luckily, Robinson-style resolution together with any complete search method (e.g., breadth-first) are simple, sufficient tools for finding such proofs. Resolution proceeds by grinding formulas in a knowledge base against a negation of a query term, and eliminates matched positive and negated formulas from conjunctive sets. If at any point the empty set is derived, then it represents a proof by contradiction of the query given the knowledge base. Note that pollution with even a single contradiction in a knowledge base may then become a sort-of mirage for resolution, obtaining false results. A contradictory knowledge base leads to a naïve belief in everything. It is therefore imperative that first-order databases are logically consistent – one of several challenges to constructing a universally applicable commonsense knowledge base.

The limited expressiveness of first order logic presents yet another challenge. For some simple expressions in natural language, translation to logic can be accomplished by compositional rules applied in tandem with each step of a syntactic analysis of an utterance, as mentioned earlier in Section 2.1; however, for many other expressions, there exists no such translation.

Further compounding on this *knowledge acquisition problem*, even given a perfect

mechanism for translation of facts from language to logic, establishing a usefully broad commonsense knowledge base is likely to require many tens of thousands of formulas. While correspondingly enormous corpora are widely available (such as CommonCrawl, Gigaword, and Wikipedia), Gordon and Van Durme (2013) showed that people tend to write about uncommon events and states, with notable bias against explicitly mentions of much of the commonsense knowledge which is necessary for true commonsense reasoning. Therefore we face problems of translation, representation, and bias. This chapter focuses on the representation problem, but first let us discuss Episodic Logic – an established, more general resresentation than first-order logic.

4.2.2 Episodic Logic

Episodic Logic (Schubert and Hwang, 2000) was developed to represent some of the diverse kinds of knowledge which are commonly expressed in natural language. It generalizes first order logic by allowing predicate and sentence modifiers, reification, quantifiers beyond existence ($\exists$) and universality ($\forall$), as well as episodic connectives for reasoning about specific, named episodes. The episodic connectives provide critical support for reasoning about temporal relations, events, and causation as are necessary for such applications as story-understanding, question answering, and planning. The implementation of Episodic Logic is Epilog, and is currently at version 2. Epilog2 is augmented with several specialist reasoners, in order to provide very efficient reasoning when certain predicates, such as those involving time or hierarchies, are involved.

There are two primary mechanisms for inference in Episodic Logic: goal-chaining and rule-instantiation. Goal-chaining directly supports question answering and theorem proving. Rule-instantiation supports spontaneous, forward inferences. Rule-instantiation gets its name from the tendency in the AI community to refer to condi-

nearly	typically	usually	normally
rarely	sometimes	often	frequently
few	many	most	three, four, five, . . .
some	no	all	every

Table 4.1: Some generalized quantifiers in English. The first two rows are adverbials, the third row exemplifies proportions and counting, and first-order quantifiers forming the last row.

tionals and universally quantified formulas as rules. Similar in spirit to the familiar rule of *modus ponens* as well as to resolution-based theorem proving, rule instantiation proceeds by matching a major premise (typically a conditional) with a minor premise (typically a ground predicate), substituting a converted version of the minor premise into the major premise, and then re-normalizing. The major and minor premise both must contain unifiable subformulas where the major subformula stands in negative context and the minor subformula is in positive context.

Goal-chaining begins with a query, such as: *is 17 prime?* It then proceeds with a similar unification based search to rule-instantiation, but rather than generating new consequents of the knowledge base, new subgoals are generated. The matching process is slightly different from that of rule chaining, restricted only to positively occurring existentially quantified variables or negatively occurring universally quantified variables. Goal-chaining in Epilog also utilizes a small set of natural deduction style rules, and is logically sound. All formulas stored by Epilog are first normalized (negations pushed inward, Skolemization, conjuncts separated, etc.). For detailed explanations of rule-instantiation, goal-chaining, and Episodic Logic in general, see Schubert and Hwang (2000).

4.2.3 Probability, Logic, and Generalized Quantifiers

Our interest is in expanding the domain of symbolic logic to include the kinds of knowledge which pertain to rules of thumb, frequencies, and general commonsense notions of probability. Logical reasoning in propositional and quantified logics is based on rules such as *modus ponens* or natural deduction rules, which are easily understood intuitively, independently applicable, tolerant of knowledge elaboration, and semantically justifiable. However, commonsense reasoning involves uncertainty. Therefore, reasoning research has long been tantalized by the goal of generalizing traditional methods so as to allow for uncertainty. Terms such as “often”, “usually”, and ”sometimes” along with other *generalized quantifiers*[3], are used prolifically in natural language, and often express qualitative knowledge of conditional probabilities, which are a crucial part of one’s knowledge that is used to interact with the world. Here are some diverse examples of the sorts of unreliable, commonsense knowledge that people seem capable of using for inference:

1. Most dogs are friendly.
2. Dogs are usually well-treated by their owner.
3. New PhD graduates are usually under 30 years old.
4. Restaurants almost always serve alcoholic beverages.
5. Supermarkets rarely sell socks.
6. Smokers often have other smokers as friends.
7. Dogs are usually someone’s pet.
8. If someone has a pet, it’s quite likely to be a dog.
9. If someone has a pet, it’s fairly likely to be a cat.

[3]See Table 4.1 for more examples of generalized quantifiers.

10. If someone hasn't eaten for several hours, s/he is probably hungry.
11. If a graduate student is among the authors of a research paper, his or her advisor is likely to be among the authors as well.
12. About 9 out of 10 research proposals are declined.

The point of this lengthy list of examples is to underscore just how heterogeneous our commonsense "rules of thumb" are, in terms of topical variety, propositional complexity, and reliability. Using very large knowledge bases containing such rules, taking account of the degree of certainty of the conclusions, lies well beyond current methods of uncertain inference in AI.

We should make a distinction here between general facts and rules. The first seven and last example are statements of fact, termed *generic sentences* in linguistic semantics. For our purposes, they can be viewed as rough statistical claims. The remaining examples (stated as conditionals) are best viewed as rules that directly suggest the degree of certainty of the consequent for any instance of the antecedent, when nothing else is known about that instance. However, the two kinds of formulations are closely related. For example, we can convert assertion (1) to a rule stating that *given any dog, it is quite likely to be friendly*. Conversely, we can regard rule (8) to be justified if in fact, say, 60% of pets are dogs. We will limit ourselves to rules that reflect statistical facts in this way. Note that as soon as we consider an *instance* of a rule, it no longer corresponds to a statistical fact, but rather is a rule for assigning a degree of belief to the consequent, when (the instance of) the antecedent is the only relevant knowledge.

An intelligent commonsense reasoning system should be able to accommodate this kind of knowledge (whether supplied as general facts or as rules), be able to communicate it, and be able to draw reasonable conclusions from it. As an example of the simplest kind of uncertain inference that a general AI system should handle, suppose

the proportion of dogs which are friendly is known to be about .8 (cf. example 1), and all we know about Rover is that Rover is a dog. Then we can conclude that Rover is friendly, with certainty .8. The general form of this example is that we are given (only) that a certain x is of type A, and that a proportion p of As are Bs, and we conclude (nonmonotonically) that x is of type B with certainty p. (See examples 1-7, 12.) Such reasoning (sometimes called *direct inference*) seems natural for humans, and thus should be trivial for a commonsense reasoning system. Arguably human beings possess (and can approximately verbalize) many millions of knowledge items of this kind, where A and B may be logically complex, and p may vary by subtle degrees from *certainly* to *certainly not*. Qualitatively expressed certainties can be put into correspondence with numeric values only in rough ways, but it does matter that the degrees of certainty vary, and sometimes approximate numbers are known, as in the last example.

Of course, in general we have, or progressively acquire, diverse knowledge about an individual such as Rover, not just a single fact. As we take more and more of this knowledge into account, our confidence concerning a particular conclusion (such as Rover's friendliness) may shift up or down. Thus, it is crucial to have sensible methods for *combining* inferences from miscellaneous facts bearing on the same conclusion. Also, our inferences do not in general consist of single steps. Rather, we draw conclusions of variable certainty from given facts (depending on the reliability of the rules employed), and proceed further from those conclusions, using any applicable rules. Thus, it is also crucial to have sensible methods of *chaining* from uncertain propositions to further conclusions, appropriately assigning degrees of certainty to those further conclusions.

In this thesis (and prior publications), we propose simple methods for probabilistic reasoning based on what we call *a-rules, b-rules*, and *c-rules* – see Section 4.7 for details. All three rule types allow for direct inference, and we specify well-motivated

methods of combining and chaining inferences. Under certain conditions, certainties of inferences based on our rules can be obtained using just the numerical probabilities of the rule consequents, given the antecedents. This is not possible in general because of dependencies among premises and inference chains, but thanks to an important innovation in our probability calculations–the use of *algebraic probabilities*–we are able to take account of such dependencies in principle. We discuss algebraic probabilities in Section 4.6. We also propose a *sparse truth assumption* (STA) in Section 4.4.1 as a general assumption about "natural" predicates that is often tenable and can greatly simplify inference of numerical certainties for the conclusions reached. Before discussing the necessary background material for evidence combination, we outline a set of challenge problems and corresponding desiderata.

4.3 Challenges and Desiderata

The fundamental challenge of reasoning with uncertainty is the choice of what to believe or do when presented with incomplete, ambiguous, or perhaps even contradictory evidence. Let us explore these issues through a series of short examples, based on the well-known "Nixon Diamond" and "Tweety the Chicken" problems. Here, we introduce the challenges, and we revisit them in Sections 4.9 and 4.10.

4.3.1 Ambiguity and The Nixon Diamond

The Nixon Diamond is a classic example of an ambiguous knowledge base. Suppose that most Quakers are pacifists, most Republicans are not pacifists, and that Nixon is both a Quaker and a Republican. The question of whether or not Nixon is a pacifist is muddled by the fact that the knowledge base somewhat entails both conclusions –

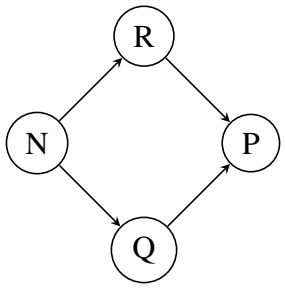


Figure 4.1: The Nixon Diamond

Nixon is and is not a pacifist. In the non-monotonic reasoning literature the terms *skeptical* reasoner and *credulous* reasoner are applied to those who do not believe that anything follows from the knowledge base, and those who are willing to believe either case. If we soften the problem to allow degrees of belief or likelihood, then it seems safe to conclude that Nixon is more likely to be pacifistic than the typical Republican, but less so than the typical Quaker. We will later show that our system is compatible with such a conclusion.

4.3.2 Specificity and The Birds

Tweety the chicken is a classic nonmonotonic reasoning challenge where there is a contradiction between rules of thumb. Consider the following variant, where we have liberally added a duck:

> Suppose that birds generally fly, that chickens and ducks are kinds of birds, all chickens and ducks are birds, and that chickens generally do not fly. Suppose further that Blue is an individual about which we only know that it is a bird, and all we know about the individual Tweety is that it is a chicken, and similarly we know that Daffy is a duck.

Of course the question is: *does Tweety fly?* A wide variety of approaches have

been developed to battle this problem of ambiguity, though most appeal to a notion of *specificity*, where conflicting inference paths are resolved by a kind of preemption mechanism that allows the more specific information (Chicken-ness) to preempt the more general information (Tweety's inferred Bird-ness). Of course, the expected answer is that Tweety probably does not fly, and that Blue probably does fly, but what about Daffy?

Daffy is a bird, and therefore seems likely to fly. But is he exactly as likely to fly as Blue, or more likely? If one thinks about birds and sets, then absent other information it seems likely that Daffy is more likely to fly than Blue based on his reduced likelihood of being a chicken; however, that is not necessarily true – what if were an ostrich, instead of a duck? We have assumed exactly the same information – namely, that an ostrich is a kind of bird.

This style of deduction – where a conclusion can change as more evidence is presented – is the basis of the field of *nonmonotonic reasoning*. I.e., "Larry is an Ostrich" leads to "Well, an Ostrich is a bird and birds fly so Larry probably flies". However, if one later learns that "Ostrich's are special birds which are too large to fly." then the previous entailment is no longer valid, as we would then conclude that Larry does not fly. Nonmonotonic logics have the property that including additional information in a knowledge base may nullify a previous entailment. Reasoning systems where $\Delta \models \alpha$ and $\Delta \subset \Delta'$ but $\Delta' \not\models \alpha$ are considered *nonmonotonic*. We discuss such systems and classic approaches in later in Section 4.5.2.

4.3.3 Stochastic Causal Behavior

We assume that there are two distinct forms of belief based knowledge, that which pertains to belief and that which pertains to disbelief. Belief knowledge is knowledge

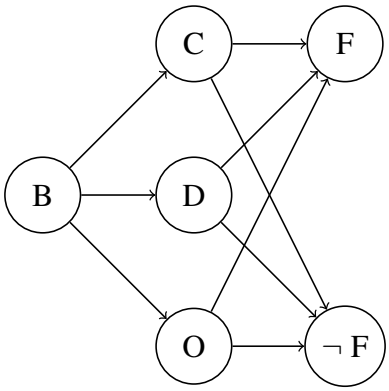


Figure 4.2: Specificity and the birds.

about things which cause other things to happen (or to be true). Disbelief knowledge is knowledge about things which cause things to NOT be true.

In the case of belief-supportive knowledge, consider things which may cause a person to have the sniffles, in particular allergies and a cold. Suppose that having allergies increase the likelihood of sniffles and that having a cold increases the likelihood of having the sniffles. What is the probability of having the sniffles if one has allergies and a cold? Qualitatively, it should be higher than given either cause alone.

Alternatively, instead of thinking of things causes, we may think of activities which make getting sick less likely. Suppose that getting a flu shot reduces likelihood of catching the flu and that frequently washing one's hands also reduces the likelihood of catching the flu. What's the probability of catching the flu when staying vaccinated and frequently washing your hands? This probability should be lower than either supplied conditional. Note that in both cases, the qualitative relationship should hold regardless of the actual value of the probabilities, be they near zero or near one. We have much more to say about inference from stochastic causal knowledge throughout the subsequent sections.

4.3.4 Relational Domains

Lastly, while the challenge problems discussed thus far have involved uncertainty in their conclusions, they are primarily ones of logical and class-based reasoning. An example application involving a different kind of probabilistic reasoning is the claim that for each common friend between a pair of persons, the likelihood that the pair are friends increases. One can see that as the number of common friends increases, so does the likelihood of transitive friendship, and that the strength of the boost in friendship likelihood decays with the distance, i.e., the number of intermediary friendships necessary to establish a transitive connection. We distinguish these complementary processes as *evidence combination* and *forward chaining*. We discuss evidence combination next.

4.4 Evidence Combination

In this section, we are concerned with the task of inference from incomplete, sparse sets of conditional probabilities which we refer to as *evidence combination problem*. We adopt a compact notation for discussing probabilities of the truth of logical statements, where we write $Pr(h|s)$ to represent $Pr(h \text{ is true } |s \text{ is true})$, where h and s are logical sentences over a common vocabulary. We use curly braces for readability. Suppose we have a knowledge base Δ which entails that $Pr(h|s) = a$ and that $Pr(h|t) = b$ for some $a, b \in \mathcal{R}$ where $0 \leq a, b \leq 1$. I.e., suppose that:

$$\Delta \models \{Pr(h|s) = a \wedge Pr(h|t) = b\} \tag{4.1}$$

What about $Pr(h|s \wedge t)$ – that is, what should we expect the likelihood of h to be, given that we learn both s and t are true? In general, there may not be any *specific*

probability entailed by the knowledge base. That is, there it may not be true that $\Delta \models Pr(h|s \wedge t) = c$ for *any* value $c \in \mathcal{R}$, because the evidence combination problem is *under-constrained*. The problem is also not completely free either, as any true values must obey standard laws of probability subject to the existence of some consistent joint-distribution. We can relate $Pr(h|s,t)$ to *marginal* probabilities $Pr(h)$ and $Pr(s,t)$ through Bayes' rule:

$$Pr(h|s,t) = \frac{Pr(s,t|h)Pr(h)}{Pr(s,t)} \tag{4.2}$$

However, as this introduces even more unknown parameters, it is clear that obtaining a reasonable estimate for $Pr(h|s,t)$ requires either more information, more assumptions, or both.

No order is guaranteed to hold among the probabilities we are given ($Pr(h|s)$, $Pr(h|t)$) and that which we wish to obtain – i.e., $Pr(h|s,t)$. It turns out that even if we constrain our parameters such that $Pr(h|s)$ and $Pr(h|t)$ are *equal*, then it is still unconstrained what effect the combined evidence should have on the likelihood of the effect. To show this, suppose that $Pr(h|s) = q = Pr(h|t)$ for some $q \in R$, and consider the following three examples which show that any relation may hold among q and $Pr(h|s,t)$.

As our first example, suppose that s and t correspond to eating spicy foods and drinking tequila, and that both actions may independently cause heartburn. In this case it seems that $Pr(h|s,t)$ – the likelihood of heartburn given that one has eaten spicy foods and drank tequila – should be *greater* than both s and greater than t – i.e., $Pr(h|s,t) > q$. If instead s and t are both facts indicating that a tossed coin is fair, such as "x is a quarter" and "x is a coin", and h represents "will land on heads when tossed", then of course the likelihood of a heads outcome when all evidence is accounted for should be *equal* to that of either piece of evidence independently. I.e.,

$Pr(h|s,t) = q$. Note also that if x is a quarter then it is also necessarily a coin – an instance of the specificity issue introduced in Section 4.3.2. Lastly, consider a causally inhibitive situation where s corresponds to "regularly gets adequate sleep", t represents "regularly washes hands", and h corresponds to "catches a cold". Further suppose that each of those actions reduces one's likelihood of catching a cold. In this case, it should follow that $Pr(h|s,t)$ is *less* than q.

Note that these examples do not depend on the prior likelihood of an event, or on the relative magnitudes of the possible causes. Instead, they are most dependent on *causal direction*. In this work, we are concerned with instances of commonsense reasoning where specific, distinct answers may be obtained based on one's causal assumptions (or lack thereof). We discuss causal models in more detail in Section 4.4.1. We discuss non-causal, correlative models subsequently in Section 4.4.2 and again, from the perspective of Markov Logic Networks, in Section 4.5.3.

4.4.1 Causally Motivated Approaches

We are especially interested in knowledge of *independent causes* and *independent effects*, which admit elegant probabilistic characterizations where *conditional independence* relations are woven together with symbolic logic. We do not consider the metaphysical properties of causality. Instead, we take concepts of *cause* and *effect* to be axiomatic, and our primary focus is on obtaining useful probabilistic models for commonsense reasoning. We next show that distinguishing causes and effects can sufficiently constrain the evidence combination problem that we may obtain concrete answers for many – but not all – situations.

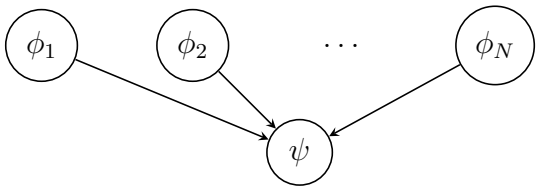


Figure 4.3: Each of the ϕ_n may independently cause event ψ.

Reasoning from Causes to Effects

Suppose we are interested in determining the likelihood of some event ψ, which we know a possible effect of one or more *independent causes*, denoted by the set $\phi_1, \cdots, \phi_N$. We have switched from h, s, and t to sub-scripted Greek letters to represent events with arbitrarily many possible causes or effects. By independent, we mean standard probabilistic independence:

$$Pr(\phi_1 \wedge \cdots \wedge \phi_n) = \prod_{i=1}^{n} Pr(\phi_i) \tag{4.3}$$

If we make a further assumption, called *accountability* by Judea Pearl (1988), that at least one of the possible causes must be true in order for ψ to occur, then we obtain a *noisy-or* model, where the likelihood of the effect corresponds to a negated conjunction of Bernoulli trials:

$$Pr(\psi|\phi_1^N) = 1 - \prod_{n=1}^{N}(1 - Pr(\psi|\neg\phi_1^{n-1} \wedge \phi_n \wedge \neg\phi_{n+1}^N)) \tag{4.4}$$

Equation 4.4 requires some explanation. Notationally, we write ϕ_i^j to represent the conjunction $\phi_i \wedge \cdots \wedge \phi_j$, and $\neg\phi_i^j$ for the conjunction of *negations* $\neg\phi_i \wedge \neg\phi_{i+1} \wedge \cdots \wedge \neg\phi_j$. Structurally, we refer to parameters of the form $Pr(\psi|\neg\phi_1^{n-1} \wedge \phi_n \wedge \phi_{n+1}^N)$ as *single cause* parameters, because they represent the probability of a given effect (ψ) when exactly one possible cause (ϕ_n) is active and all other possible causes are

inactive – thereby isolating the causal weight to a single cause, irrespective of all others. This kind of parameterization – and associated independent causal relations – yields a powerful, incremental evidence combination scheme, and describes the likelihood of an effect given 2^N possible combinations of causes while only requiring a linear number of parameters. See Figure 4.3.

The gradual accumulation of evidence, as might occur in commonsense reasoning domains, can be accounted for with a very straightforward, incremental updating procedure. Each successive update can be thought of as siphoning off some of the probability from the case where the effect does not occur, and giving it to the case where it does occur. If we let $q = Pr(\psi|\phi_1^{n-1})$ be the current probability of ψ, and let $f(\phi_n) = Pr(\psi|\neg\phi_1^{n-1} \wedge \phi_n \wedge \neg\phi_{n+1}^{N})$ be the single-cause conditional probability of ψ given ϕ_n, then the updated probability of the outcome also given ϕ_n is given by:

$$Pr(\psi|\phi_1, \cdots, \phi_n) = q + (1 - q)f(\phi_n). \tag{4.5}$$

Let us consider the concept of single cause parameters in more detail. The distinguishing feature of single cause parameters stems from the accountability assumption, and the associated constraint that the parameter corresponds to the likelihood of the event when all other possible causes are inactive. We will consider the difference from two perspectives: obtaining single cause parameters by sampling from the joint distribution, and obtaining single cause parameters by algebra from standard conditionals.

Single cause parameters can be difficult to evaluate by sampling because they represent a narrower phenomenological slice of parameter space. Although there are only linearly many such parameters, because of their assumed independence, as the number of parameters increases, the likelihood of all but one being false decreases exponentially. Therefore simple rejection sampling or counting procedures are unlikely to be

helpful for events with many possible causes, although they are adequate for events with fewer causes.

Single cause parameters are not directly available from standard conditional parameters either – unless additional parameters are also known. Moreover, while the two values are not completely free parameters, they may be very different for any particular cause and event. Suppose for an event with two independent causes that ϕ_1 is almost certainly true and is a very strong predictor of ψ – i.e., suppose both $Pr(\phi_1) \approx 1$ and $Pr(\psi|\phi_1 \wedge \neg\phi_2) \approx 1$. It follows that $Pr(\psi|\phi_2) \approx 1$ regardless of the value $Pr(\psi|\neg\phi_1 \wedge \phi_2)$, and thus single cause parameters can be vastly different from simple conditionals. To see this, recall that we have assumed $\phi_1 \perp\!\!\!\perp \phi_2$, so it follows that $Pr(\phi_1)Pr(\phi_2) = Pr(\phi_1 \wedge \phi_2)$, and therefore:

$$\begin{aligned}
Pr(\psi|\phi_2) &= Pr(\psi \wedge \phi_1|\phi_2) + Pr(\psi \wedge \neg\phi_1|\phi_2) \\
&= \frac{Pr(\psi \wedge \phi_1 \wedge \phi_2)}{Pr(\phi_2)} \frac{Pr(\phi_1)}{Pr(\phi_1)} + \cdots \\
&= \frac{Pr(\psi \wedge \phi_1 \wedge \phi_2)Pr(\phi_1)}{Pr(\phi_1 \wedge \phi_2)} + \cdots \\
&= Pr(\psi|\phi_1 \wedge \phi_2)Pr(\phi_1) + Pr(\psi|\neg\phi_1 \wedge \phi_2)Pr(\neg\phi_1) \\
&\approx Pr(\psi|\phi_1 \wedge \phi_2) \\
&= Pr(\psi|\phi_1 \wedge \neg\phi_2) + \cdots \quad \text{[Eqn 4.5]} \\
&\approx 1.
\end{aligned} \tag{4.6}$$

If we assume that a sufficient number of configurations are available, then it is easy to obtain the various priors $Pr(\phi_i)$ and $Pr(\psi)$, and by independence we can obtain the probability of any particular configuration of the causes, but the single cause parameter $Pr(\psi|\neg\phi_1^{i-1} \wedge \phi_i \wedge \neg\phi_{i+1}^n)$ remains intractable. Even Bayes' rule is little help, because

it would require us to obtain $Pr(\neg\phi_1^{i-1} \wedge \phi_i \wedge \neg\phi_{i+1}^n|\psi)$, which can *not* be factored into a product of n terms. Observing ψ is said to *explain away* the independence of the ϕ's.

In many situations we find that we can get away with marginal cause approximations of single cause parameters. Within commonsense reasoning domains, the vast majority of predicates are typically false when applied to arbitrary terms. E.g., it's evident that most people are not the current president of the United States, almost all animals are not dogs, etc. If we assume that $Pr(\neg\phi_i) \gg Pr(\phi_i)$ – for typical instances of ground predications or sentences – then we can make coarse approximations that $Pr(\neg\phi_1^{i-1} \wedge \phi_i \wedge \phi_{i+1}^n) \approx Pr(\phi_i)$ and that $Pr(\psi|\neg\phi_1^{i-1} \wedge \phi_i \wedge \phi_{i+1}^n) \approx Pr(\psi|\phi_i)$. We call this the *Sparse-Truth Assumption* (STA), which is similar in spirit to the well-known Closed World Assumption of default reasoning, Markov logic, and other first-order domains. Note that the assumption that $Pr(\phi_1) \approx 1$ from our preceding example violates the STA. We originally introduced this assumption in Purtee and Schubert (2016) and Purtee and Schubert (2017), and will be explicit about its use later in Section 4.7 where we use it in constructing commonsense probabilistic models.

Reasoning From Effects to Causes

We have described the noisy-or independent cause model, and how to infer the probability of an effect, given knowledge of possible causes. The complementary problem is to ascertain likely causes, given some observed effect. Both styles of inference can be thought of as a problem of evaluating conditional probabilities – $Pr(\textit{effect}|\textit{cause})$ and $Pr(\textit{cause}|\textit{effect})$. Judea Pearl refers to the former as *predictive* or *prospective* probabilities, and the latter as *diagnostic* or *retrospective* probabilities (Pearl, 1988). Ian Mackay refers to these as *forward* or *generative* probabilities, and the latter as *inverted* or *inverse* probabilities (MacKay, 2002). We will also sometimes use the phrase *back-*

ward reasoning. This task is related to those of abductive reasoning in symbolic logic and most plausible explanation inference from the fields of statistics and machine learning. Assuming we have available priors over the causes and the effect, we can perform backward reasoning with the noisy-or model by application of Bayes' theorem. If we let Φ represent the set of all ϕ_i, then:

$$Pr(\Phi|\psi) = \frac{Pr(\psi|\Phi)Pr(\Phi)}{Pr(\psi)} \tag{4.7}$$

The structure of a causal model depends on independence relationships, not the specific choice of known parameters. Equation 4.7 does not assume any particular parameterization or independence relations. It is simply true from basic probability – our causal model and parameterization do not have to match, as long as they are consistent. The likelihood of individual causes can be obtained with an application of the sum-rule:

$$Pr(\phi_i|\psi) = \sum_{\Phi/i} Pr(\Phi|\psi) \tag{4.8}$$

One can also obtain the highest probability configuration of Φ, which is known as the *most probable explanation* of the findings.

Consider the problem of evidence accumulation. In Equation 4.7, we have exactly one observation – whether or not ψ occurred – and we evaluate the conditional probability of some set of possible causes. Now let us generalize to the case of a possible cause having several possible effects, and therefore an opportunity for incremental evidence combination. See Figure 4.4 where we represent event ϕ as having M *conditionally independent* effects $\xi_1, \cdots, \xi_M$. We also consider the case where events may have multiple independent causes and multiple independent effects as in Figure 4.5, and a simplified version we call the "W-graph" as shown in Figure 4.7.

To derive an incremental inference method, let us first mention the well-known odds-likelihood perspective on probability. Instead of dealing with probabilities directly, it is sometimes useful to discuss the odds of an event ψ as:

$$O(\psi) = \frac{Pr(\psi)}{Pr(\neg\psi)} \tag{4.9}$$

When discussing events which are unlikely, it is common to refer to the odds *against* the event so that small fractions are not necessary. As our evidence combination discussion has been primarily concerned with conditional probabilities. The analog from the odds perspective is given by conditional odds – i.e.,

$$O(\psi|\xi) = \frac{Pr(\psi|\xi)}{Pr(\neg\psi|\xi)} \tag{4.10}$$

An interesting scenario happens when we simultaneously apply Bayes' rule to the numerator and denominator of Equation 4.10. Not only do the $Pr(\xi)$ terms cancel, but the expression factors into an interesting product:

$$O(\psi|\xi) = \frac{Pr(\xi|\psi)}{Pr(\xi|\neg\psi)} O(\psi) \tag{4.11}$$

The term on the left is known as the *Bayes' factor* or *Bayesian likelihood*, and the term on the right is the prior odds of ψ. What about an event with multiple effects, such as in Figures 4.4, 4.5, 4.6, and 4.7? In this case, the conditional independence assumed between the $\xi's$ given ψ allows the factorization:

$$O(\psi|\xi_1^N) = \text{Bf}(\xi_1^N|\psi) O(\psi) \tag{4.12}$$

$$O(\psi|\xi_1^N) = O(\psi)\prod_n \mathrm{Bf}(\xi_n|\psi) \tag{4.13}$$

Everything is valid from first principles in all cases until the second line of Equation 4.13, which achieves a dynamic-programming style factorization based on the assumed conditional independence of the effects given the cause. In addition to enabling incremental inference, this factorization reduces the necessary number of parameters known from 2^N to N. Now, we may soundly and incrementally reason about independent causes *and* effects, and we may perform probabilistic inference along the causal grain (i.e., obtaining generative or forward probabilities) or against it (i.e., with backward, diagnostic probabilities). However, not all models are causal, and though we may know some models are causal we may not know the direction. In such cases, we may turn to the principle of maximum entropy, as described in the next section.

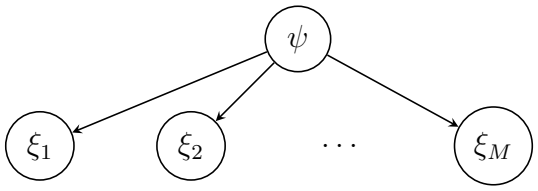


Figure 4.4: An event with several possible effects.

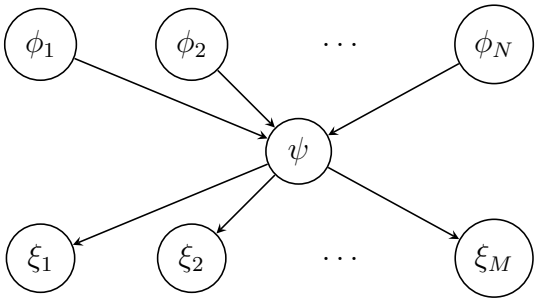


Figure 4.5: An event with several possible causes and effects.

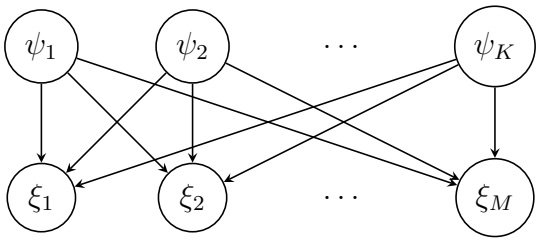


Figure 4.6: An acyclic, bipartite graph of causes and effects.

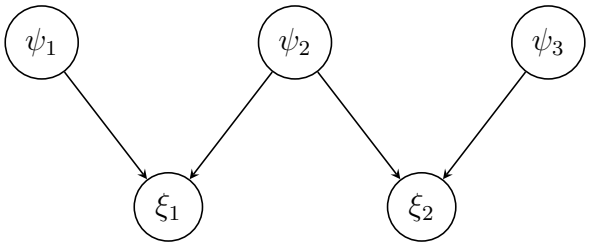


Figure 4.7: The "W" graph.

4.4.2 Entropy Maximization

Entropy maximization is an information-theoretic technique to obtain the distribution over a set of random variables which corresponds to making the fewest assumptions possible. Early work on on maximum entropy is due to Edward Jaynes (1957). This technique has been particularly successful when applied to problems of natural language processing, including part-of-speech tagging (Ratnaparkhi, 1996), sentence-boundary detection (Reynar and Ratnaparkhi, 1997), and syntactic parsing (Charniak, 1995). Markov Logic Networks, introduced by Richardson and Domingos (2006) and discussed in Section 4.5.3, provide an example of maximum entropy methods applied to logical inference. Grove et al. (1994) present an argument for a deep connection between maximum entropy and logical inference from the random worlds perspective. Their argument is restricted to supporting the connection for monadic logic (i.e., without relations or equality). They speculate, but do not prove, that the connection does *not* hold in the case of full predicate calculus. We discuss the possible worlds approach in more detail in Section 4.5.1. Here, we present a different, shallower application of maximum entropy directly to the problem of evidence combination in support of logical inference based on expectations over possible worlds. Epistemic knowledge – our constraints about the likelihood of events in the world – is accounted for through Lagrangian constraints on the optimization problem, which is of course based on the standard information theoretic definition of entropy. Let (Ω, Pr) be any discrete probability space, then the entropy is defined to be:

$$\mathrm{H}_{\Omega} = -\sum_{x \in \Omega} Pr_{\Omega}(x) \cdot \log(Pr_{\Omega}(x)) \tag{4.14}$$

Let us consider marginal and conditional constraints. That is, constraints such as

$Pr(\psi) = p$ and $Pr(\xi|\psi) = q$, respectively. But to begin with, we must answer the question of what exactly does $Pr(\psi) = p$ mean? By definition, it must be that there is some probability space of events Ω and that $\psi \subseteq \Omega$, but what would Ω look like when reasoning about common sense logical assertions? One perspective where the principle of maximum entropy may help is is that we may be in but one of many possible worlds, and we wish to obtain a distribution over those possible worlds as to which one is the "real" world. Note that his is in contrast to the premise of Grove et al. (1994) , who define their probabilities differently as limiting proportions of equally-likely worlds which satisfy a given formula as the size of the domain tends to infinity. We discuss the random worlds method in Section 4.5.1.

The number of possible worlds is typically quite large for a given theory; a fact which often causes problems for model-counting approaches. In propositional domains, the set of possible worlds is very straightforward – it is the space of all combinations of truth values for the propositions. For first-order theories, it is less clear what might be a natural form for Ω. For finite domains, this set is finite, but becomes intractably enormous very rapidly. E.g., a domain with N elements and a single 1-place function elicits $N!$ possible worlds. In infinite domains, there are uncountably many possible worlds.

Returning to the optimization problem, we introduce Lagrangian constraints to Equation 4.14, and solve for the function $Pr(x)$ at the maximum. This point must exist, because Equation 4.14 and our constraint functions are convex. Note that here we may solve analytically, but in practice often numerical optimization algorithms are used instead – such as gradient descent, Newton's method, or more recent variants such as L-BFGS (Boyd and Vandenberghe, 2004).

$$I_\psi(x) = \begin{cases} 1 & \Delta \cup x \models \psi \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

$$\mathrm{E}\left[I_\psi\right] = \sum_{x \in \Omega} Pr(x) I_\psi(x) \tag{4.16}$$

The general objective function for Boolean maximum entropy with conditional constraints $\{Pr(\phi_i) = p_i\}$ and marginal constraints $\{Pr(\psi|\phi_j) = q_j\}$ is:

$$H_\Omega - \sum_i \lambda_i \cdot (\mathrm{E}[\mathrm{I}_{\phi_i(x)}] - q_i) + \sum_j \lambda_j (\mathrm{E}[\mathrm{I}_{\psi \wedge \phi_j}(x)] - p_j \cdot E[\mathrm{I}_{\phi_j}(x)]) \tag{4.17}$$

Under this objective, the maximum entropy probability function is again log-linear in the Lagrangians. Let $x \in \Omega$ be any world, then:

$$Pr(x) = \frac{1}{Z} e^{\sum_i \lambda_i \cdot I_{\phi_i}(x) + \sum_j \lambda_j \cdot (I_{\psi \wedge \phi_j}(x) - q_j \cdot I_{\phi_j}(x))} \tag{4.18}$$

where

$$Z = \sum_{x \in \Omega} e^{\sum_i \lambda_i \cdot I_{\phi_i}(x) + \sum_j \lambda_j \cdot (I_{\psi \wedge \phi_j}(x) - q_j \cdot I_{\phi_j}(x))} \tag{4.19}$$

Dynamic programming is necessary to evaluate the objective function when more than a few Boolean variables are used. The complexity of the best dynamic programming algorithm still depends exponentially on the size of the largest set of variables which are related by the constraints. The probability of the formula ψ can be obtained as the sum of probabilities of all worlds wherein it is true:

$$Pr(\psi) = \sum_{x \in \psi} Pr(x) \tag{4.20}$$

Note that we have not specified and do not restrict these conditional probabilities to having come from either causal model (Figure 4.3 or Figures 4.4). This can be advantageous, as it is not always straightforward – or possible – to correctly choose causal orderings. The price for this advantage is that in cases where there is an underlying causal model, if we ignore it then our inferences may not be causally valid. As the saying goes, correlation is not causation.

4.5 Related Work

Various approaches to uncertain inference have been developed over the past decades, but they fall short of generalizing classical reasoning in a way that naturally extends such reasoning, retaining its attractive properties while allowing for degrees of certainty.

Production systems (also called rule-based systems or expert systems) such as Shortliffe and Buchanan's pioneering MYCIN system for diagnosing infections and recommending therapy (Shortliffe and Buchanan, 1975) use intuitive if-then rules with bounded numerical "certainty factors" (e.g., in the interval [-1.0,1.0]); these are combined in ways intended to boost or lower certainties in an intuitively natural direction, without going out of bounds. However, the rules operate on attribute-value lists, rather than logical formulas, and the certainty factors are not grounded in statistical knowledge or combined in ways that have any theoretical basis.

Bayesian networks (BNs) have the important advantage that known conditional frequencies relating rule antecedents to consequents can be used directly to set the network parameters; as well, they exploit independence assumptions that often seem justified, for example when the node-to-node connections can be interpreted as causal

influences. But BN inference is restricted to propositional variables, and involves complex marginalization processes (often approximated by sampling methods) quite unlike classical reasoning. While various Prolog-like quantified extensions exist (Milch et al., 2007; Raedt et al., 2007; Getoor and Grant, 2006), uncertain inference generally devolves into performing standard BN inference on *ad hoc* BNs built from ground instances of quantified relationships.

Undirected graphical models such as Markov networks share the advantage of BNs that their parameters are anchored in empirical frequencies. An advantage they have over BNs is that they do not presuppose any causal knowledge – "neighboring" variables are simply regarded as statistically related. However, the flip side is that the network parameters cannot be set directly using conditional frequency knowledge, but rather must be mathematically inferred from large data sets. This is a severe limitation, in view of the abundance of commonsense knowledge that we would like to impart to machines. Also, as in the case of BNs, quantified versions of undirected graphical models such as Markov Logic Networks (MLNs) rely on ad hoc grounding to perform uncertain inference, and there is no provision for the simple kind of direct inference about the likelihood of x being a B, given that it is an A, mentioned above.

Finally, formal extensions of first-order logic allowing for statistical knowledge and probabilistic qualification of sentences tend to be strong on semantics but weak on inference mechanisms. The probability of a conclusion is typically measured in terms of the proportion of possible worlds where it holds, but calculating this proportion (perhaps in the limit as the domain of individuals is expanded) can be extremely challenging, and this calculation must be performed each time the kb is expanded. Bacchus et al. (1996; 1993) proved that first-order probabilistic reasoning based on model counting has desirable theoretical properties, such as respecting specificity and entailment relations, and

supporting direct inference, but exact inference for knowledge bases with more than monadic predications is intractable. More recently, promising work has been done on lifted inference which aims to provide tractable algorithms for inference with model counting semantics (Gribkoff et al., 2014; Kazemi et al., 2016), but these methods still fall short of classical style reasoning with first-order information.

Our work is aimed at providing a probabilistic extension of quantified logic, similar in style to default logics (and to some extent production systems and BNs) but using rules that

- are framed in terms of logical antecedent and consequent formulas, allowing for matchable variables and for uncertainty of the consequent;
- are grounded in empirical frequencies (perhaps just estimates from linguistically stated generalizations); and
- are independently applicable (under certain provisos) in drawing uncertain conclusions from a given set of premises.

Further, we seek rules that jointly yield sensible results in cases where we know what the results should be, such as in Bayesian networks (BNs). We first discuss *a-rules* in some detail, motivating them and considering their range of applicability. We also provide characterizations of *b-rules* and *c-rules*, indicating where these are applicable, and also introduce algebraic probabilities as a general way of handling interdependencies. Our methods allow rule-based inferences in Bayesian networks that agree with those grounded in standard BN theory, and they provide intuitively reasonable, quantitative counterparts to nonmonotonic reasoning methods. As mentioned in the introduction, our main experimental test shows that a straightforward application of three a-rules can more than match the inferential accuracy of Markov Logic Networks, as originally tested by Richardson and Domingos (2006) in a "graduate students and their advisors"

domain. In addition, we also consider how our approach relates to classical nonmonotonic reasoning problems of Tweety the Chicken and the Nixon Diamond.

4.5.1 Random Worlds

The Random Worlds (RW) approach extends first order logic with *proportion terms* and *proportion expressions*. The proportion expressions have a well defined semantics in terms of a fixed domain and interpretation. To reason probabilistically, the RW method considers the set of all possible worlds (interpretative mappings) which are consistent with a given knowledge base, and assigns probabilities to formulas based on the proportion of those worlds in which the formula in question is also true. The clean semantics of random worlds comes at a price. Because the definition of probability is in terms of model counting, in general determination of probability is at least as complex as counting the number of solutions to exponentially many semi-decidable problems. There is a strong connection between maximum entropy methods for Boolean logic and random worlds probabilities when the logic is restricted to unary predicates – a still very expressive and important set of predicates (Grove et al., 1994). It is not clear what, if any, extensions are possible beyond unary predications.

Properties of Random Worlds

Bachhus et al. (1993; 1996) proved a variety of desirable properties hold of the RW method; we restate and comment on their results below. Note that the assumptions are that ψ and ϕ represent formulas, not merely predicates. They use the expression $\|\psi(x)\|_x$ to denote the proportion of domain elements which satisfy the formulas ψ, and $\|\psi(x)|\phi(x)\|_x$ to similarly denote the proportion of domain elements which satisfy ψ, when considering only those which satisfy ϕ. The function $Pr_\infty(\cdot)$ represents the

relative proportion of models of some formula as the size of the domain tends toward infinity and the allowable approximation error tends toward zero.[4]

1. **Logical Entailment:**

   The set $D(\Delta) = \{\phi : Pr_\infty(\phi|\Delta) = 1\}$ contains Δ and is closed under valid implication. That is, if $\models (\theta \Rightarrow \psi)$ and $\theta \in D(\Delta)$, then $\psi \in D(\Delta)$.

2. **Direct Inference:**

   Let $\phi(\vec{x})$, and $\psi(\vec{x})$ be formulas, where no constant in $\vec{c}$ appears in $\phi(\vec{x})$ or $\psi(\vec{x})$. Then

   $$Pr_\infty(\phi(\vec{c})|\{\psi(\vec{c}) \wedge \alpha \approx \|\phi(\vec{x})|\psi(\vec{x})\|_{\vec{x}}\}) = \alpha$$

   For example, if $\Delta = \{Bird(Tweety) \wedge \|Flies(x)|Bird(x)\|_x \approx 0.95\}$, then $Pr_\infty(Flies(Tweety)|\Delta) = 0.95$.

3. **Specificity:**

   Let $\vec{c}$ be a tuple of constants, and ϕ, ψ_1, and ψ_2 be formulas which do not contain any constant in $\vec{c}$. Let Δ be the conjunction

   $$\psi_1(\vec{c}) \wedge (\|\phi(\vec{x})|\psi_1(\vec{x})\|_{\vec{x}} \approx \alpha) \wedge$$

   $$(\|\phi(\vec{x})|\psi_2(\vec{x})\|_{\vec{x}} \approx_j \beta) \wedge \forall\vec{x}(\psi_1(\vec{x}) \Rightarrow \psi_2(\vec{x}))$$

   Then

   $$Pr_\infty(\phi(\vec{c})|\Delta) = \alpha.$$

   This is a desired property from the point of view of default reasoning. Suppose we have a simple knowledge base about birds: Birds typically fly. Penguins are

[4] More specifically, it is the common value of the limit superior and the limit inferior. This function is only defined when both such limits exist and have the same value. Note also that Bacchus et al. use a variety of tolerance values for approximations, written as $\approx_i$. Here, we drop the subscript and simply write $\approx$ for clarity.

birds. Penguins do not fly. Opus is a penguin. These statements can be translated into the language of random worlds as:

- $\forall x Penguin(x) \Rightarrow Bird(x)$.
- $\|Fly(x)|Bird(x)\|_x \approx 1$.
- $\|Fly(x)|Penguin(x)\|_x \approx 0$.
- $Penguin(Opus)$.

The specificity result shows that a random worlds interpretation of this Δ would entail that $Pr_\infty(Flies(Opus)|\Delta) = 0$.

4. **Relevance:**

   Suppose Δ has the form:

$$\psi(\vec{c}) \wedge (\|\phi(\vec{x})|\psi(\vec{x})\|_{\vec{x}} \approx \alpha) \wedge \Delta',$$

   where no constant in $\vec{c}$ appears in $\psi(\vec{x})$ or $\phi(\vec{x})$, and neither ψ nor Δ' mention any symbol in ψ. Then, $Pr_\infty(\phi(\vec{c})|\Delta) = \alpha)$.

   Note the subtle difference in assumptions of this theorem. No constant in the tuple of constants $\vec{c}$ is allowed to appear in $\psi(\vec{x})$ or $\phi(\vec{x})$. No symbol in the hypothesis formula $\psi(\vec{x})$ is allowed to be shared with $\phi(\vec{x})$ or Δ'.

   Suppose we augment our example bird knowledge base with additional information:

   - $Bird(Tweety)$.
   - $Yellow(Tweety)$.
   - $\|Beaked(x)|Bird(x)\|_x = 1$.

Then from this theorem we are still able to derive that Tweety flies and that Opus does not, while additionally able to show that Opus and Tweety are Beaked and that Tweety is yellow. This shows that the RW approach (with suitable assumptions) does not lose the specificity property in the presence of additional but irrelevant information, in either the typical (Tweety) or atypical (Opus) case. We discuss the behavior of our model on such problems in Sections 4.9 and 4.10.

4.5.2 Inheritance Networks

One of the earlier approaches toward solving the specificity problem is provided by non-monotonic inheritance networks (NINs), which are directed representations of defeasible class subsumption relations. Good references for nonmonotonic inheritance networks are Touretzky (1986), Stein (1990), Selman (1989), and Brachman & Levesque (2004).

Inheritance hierarchies originated in the 1970s/1980s as an attempt to solve the problem of nonmonotonic reasoning - that is, reasoning for which ones set of beliefs may change over time by revision. Inheritance hierarchies encode a set of default assumptions about an element of the domain, where the assumptions are encoded as a graph with monadic predicates as nodes and subset relations between the denotations of those predicates encoded as edges. The nonmonotonic/defeasible component enters in through the assumption that these edges are not exact, but instead only reflect what is typical. For instance, while all birds are animals, it can only be said that most birds fly. Both of these rules are encoded as directed edges, the former being from $Bird$ to $Animal$ and the latter from $Bird$ to $Flies$. Inheritance hierarchies also allow for negated edges, which encode knowledge such as "chickens do not fly" – note that this is stronger than it may seem, as it instead means that *most* chickens do not fly, rather

than simply that some of them do not. The compactness of the representation comes from the fact that additional rules are implicit in the path structure of the graph. We can reason that, because Tweety is a chicken, and all chickens are birds, and all birds are animals, it follows that Tweety is an animal.

We cannot allow all paths to be believed, because inheritance hierarchies are typically ambiguous ones. In our example, it is possible to reason both that Tweety flies (due to being a bird) and that Tweety does not fly (due to being a chicken). Ambiguity between classes is resolved according to various path-topological rules. While there are numerous distinct characterizations of such rules, most rely on a notion of preemption which we refer to as the *triangle rule*. Suppose from ϕ one may prove both ψ and $\neg\psi$, but the path to ψ involves multiple rules and the path to $\neg\psi$ is a single step. I.e., there is explicitly a rule "all $\phi's$ are not $\psi's$" in the KB. In the case of the Tweety example, we prefer to conclude that Tweety cannot fly because he is a chicken, and Tweety's chickenness has been axiomatically asserted. This is in contrast to his birdness, which is obtained by inference. A disambiguating *extension* of an inheritance hierarchy is non-contradictory subset of possible inferences, obtained from the transitive closure of the corresponding graph. In general, there are exponentially many possible extension, but some are considered *preferred extensions*, which are those that obey the principle of specificity. There is disagreement in the literature on exactly how to apply the principle, though in most approaches specificity is determined by allowing some arguments to preempt other arguments.

Inheritance hierarchies admit reasoning to varying levels of soundness. The simplest and most permissive form is termed *credulous reasoning*. A credulous reasoner chooses any preferred, *credulous extension*, and commits to believing all of its consequences. This is not the same as simply choosing any belief which is supported by a

path in the hierarchy, as it first prunes the set of beliefs by removing arguments which are preempted by other arguments, and chooses either alternative when presented with ambiguous. In contrast, *skeptical reasoning* is the principle that only those predications which are true in *all* credulous extensions are valid. Both credulous and skeptical reasoners rely on calculation of specificity, which is possible polynomial time (Stein, 1990); however, skeptical reasoning cannot be performed in a purely path based way (i.e., without enumeration) and is believed to be computationally intractable (Selman and Levesque, 1989).

Finally, we should mention that inheritance networks lack the ability to distinguish genuine entailment relations from those which are merely "normally true". In particular, all rules are assumed to be defeasible, and they are not distinguished by degree or certainty. Moreover, because of the reliance on directed paths, it is not clear how to generalize nonmonotonic inheritance methods beyond the monadic case. Here, we briefly summarize nonmonotonic inheritance networks because of our shared problem domain, and their impact on the nascent field of reasoning in the presence of uncertainty. While we considered taking inheritance networks as a starting point, it is for these reasons that we deemed them inadequate for our purposes and sought to develop our simple rules for probabilistic reasoning.

4.5.3 Markov Logic Networks

Markov Logic is an attempt to unify probabilistic reasoning with first order logic by associating each formula in a first order knowledge base with a non-negative weight (Richardson and Domingos, 2006). A Markov Logic Network can be thought of as a template for constructing Markov Random Fields (i.e., undirected graphical models).

Markov Logic assigns probabilities to first order formulas (with respect to a weighted

knowledge base) by considering the set of all possible groundings of all formulas and penalizing any resulting groundings which violate a formula in the knowledge base. The penalty is derived from the weight of the violated formula.

The key assumptions necessary for Markov Logic are that the domain is finite (in particular, the domain consists exactly of a set of known constants) and that the value of all functions are known. (The function constraint can technically be eliminated by considering the set of all possible functions, but that is computationally expensive.)

Given a Markov logic network ($M_{L,C}$) comprised of a set of weighted logical formulas (L) and a set of constants (C), the conditional probability of any first order formula (F_1) given any other first order formula F_2 is:

$$P(F_1|F_2, M_{L,C}) = \frac{P(F_1 \wedge F_1|M_{L,C})}{P(F_2|M_{L,C})} = \frac{\sum_{x \in X_{F_1} \cap X_{F_2}} P(X = x|M_{L,C})}{\sum_{x \in X_{F_2}} P(X = x|M_{L,C})}$$

where X_{F_i} is the set of worlds where F_i holds, and $P(x|M_{L,C})$ is given by:

$$P(X = z) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)},$$

where $n_i(x)$ is the number of true groundings of F_i in x, $x_{\{i\}}$ is the truth values of the atoms appearing in F_i, and $\phi_i(x_{\{i\}}) = e^{w_i}$.

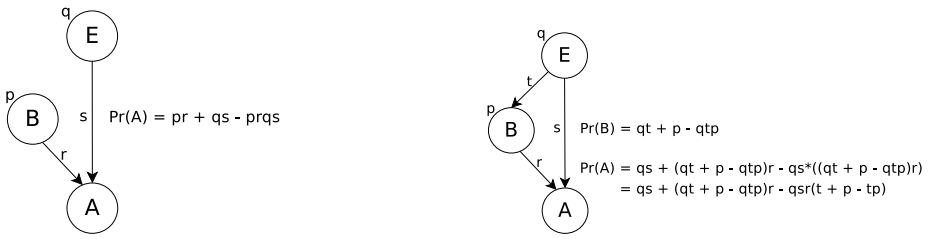
The probability space of Markov Logics are finite, ground universe of predicates. The probability space of Random Worlds is similarly derived from the set of all possible logical interpretations which satisfy a given model. In the case of Markov Logic $Pr(\psi|\phi)$ is defined for any pair of wffs ψ and ϕ. For Random Worlds, the probability's existence is dependent on the existence of limits. In both cases, the underlying model provides a way to assign a probability to any well-formed formula of first order logic, they differ with respect to how the probability enters: Markov Logic assigns weights

to formulas whereas Random Worlds explicitly quantifies the proportion of domain elements for which certain formulas are true. Both approaches retain proper entailment. Both are computationally intractable in general. Inference in Markov Logic is #P complete. Inference in Markov Logic is fully decidable as a consequence of the assumption that the domain size is known (and finite). Inference in RW is at least as hard as entailment in FOL and is therefore not strictly decidable.

4.6 Algebraic Probabilities

Algebraic probabilities were introduced by Schubert (2004) (under the name "quasi-probabilities") as a general means of manipulating and evaluating probabilities in Bayesian networks (BNs) built from noisy-AND/OR/NOT nodes – which can model all boolean-valued BNs. The combinatory algebra of these probabilities is closely analogous to ordinary algebra using +, -, and product, except that product, written '*', is *idempotent*, i.e., $\alpha * \alpha = \alpha$ for any algebraic probability expression α. (The idempotency of '*' derives from that of logical $\wedge$, while $(1 - \alpha)$ represents the probability of a negation). Expressions are formed by combining *elementary* probabilities, which are independent of one another. For a product $\alpha * \beta$ where α and β share no elementary probabilities, '*' reduces to ordinary product, i.e., $\alpha * \beta = \alpha\beta$.

A fundamental problem in uncertain inference is that inference chains leading to the same (or opposite) conclusions may rely on some of the same uncertain knowledge items along the way. In such a case the two inference chains do not provide independent evidence for the conclusion, and so the probability of the conclusion cannot be obtained correctly by combining the numerical results of the inference chains as if the chains were independent. Rather, we need to take account of the "provenance" of the results



(a) Standard (b) Modified

Figure 4.8: The Burglary-Alarm-Earthquake Example.

being combined. Algebraic probabilities provide a general way around this difficulty, at least in principle.

The simplest demonstration of this point is provided by considering applying the same rule twice to the same premise, with the same conclusion. In such a case, the two rule applications are certainly not providing independent evidence, but rather the *same* evidence, twice. Yet by using algebraic probabilities and idempotent products we obtain the correct result. Suppose that when the rule is first applied, the probability of the conclusion is still 0. Then if the certainty variable of the rule is p, the result will be p. If the rule is applied a second time, the result will be $p+p*(1-p) = p + p - p = p$, i.e., unchanged. It is easy to show that this still holds if the initial probability of the conclusion is some arbitrary algebraic value α.

A more subtle example is provided by a variant of "Mr. Holmes' burglar alarm". The standard version is shown in Figure 4.8(a), where a burglary B and an earthquake E are independent causes of an alarm A (and they are the *only* possible causes). In variant (b), it is assumed that burglars may take advantage of earthquakes to commit burglaries. Thus both influences on A have a common ancestor in their possible causation, E. As such, they are no longer independent. Yet, as is shown in the figure, when we combine the influences "as if" they were independent – but using idempotent product – we obtain the exact result for the probability of A (as the reader can verify).

Details of algebraic probability manipulation in BNs can be found in Schubert (2004), but we should mention the general conditioning rule: For any prior algebraic probability *Pr**(ϕ), when evidence ψ is brought to bear, the resultant probability is

$$Pr^*(\phi)\frac{*Pr^*(\psi)}{*Pr^*(\psi)} \tag{4.21}$$

where *Pr**(ϕ) denotes the algebraic probability of ϕ (similarly, *Pr**(ψ)) and the two occurrences of $*Pr^*(\psi)$ indicate idempotent multiplication of the numerator and denominator of *Pr**(ϕ) by *Pr**(ψ). For example, suppose that A is given as evidence in Figure 4.8(b). Then the probability of E is updated to

$$\begin{gathered}\frac{q * [(p+qt-pqt)r+qs-qsr(p+t-pt)]}{(p+qt-pqt)r+qs-qsr(p+t-pt)} = \\ \frac{q[(p+t-pt)r+s-sr(p+t-pt)]}{(p+qt-pqt)r+qs-qsr(p+t-pt)} = \\ \frac{q[s+r(1-s)(p+t-pt)]}{(p+qt-pqt)r+qs-qsr(p+t-pt)}s.\end{gathered} \tag{4.22}$$

A convenient generalization of the algebra, for example in handling c-rules or mutually exclusive rule antecedents, is to allow "spectra" $p_1, p_2, ..., p_k$ of elementary probabilities, where $p_i * p_j = 0$ for distinct i, j. Note that this generalizes the fact that $p*(1-p) = p-p = 0$. Members of different elementary spectra are considered independent of one another.

4.7 Simple Rules for Probabilistic Reasoning

We write rules as implications with a *certainty variable* modifying the consequent (or in c-rules, multiple certainty variables, each modifying one of a set of alternative

consequents). Certainty variables are treated as boolean random variables (Bernoulli variables) whose probability is the probability of truth of the consequent, when the antecedent is true. Therefore, a certainty variable can be thought of as added conjunctively to the rule antecedent, so that the consequent is true whenever both the antecedent and the certainty variable are true. This probability should at the same time reflect the proportion of true instances of the antecedent where the consequent is true.

We should note here that a certainty variable is a unique Bernoulli variable only in a rule that contains no matchable ("universal") variables as predicate arguments. When there are such matchable variables, these are also considered arguments of the certainty variable, i.e., the latter is in general a *function* of the matchable rule arguments. The value of this certainty function is regarded as a distinct Bernoulli variable for each combination of values of its arguments. For example, given a rule stating that an arbitrary dog x is friendly with probability $(p\ x)$ (see below), $(p$ ***Rover***$)$ and $(p$ ***Fido***$)$ are distinct random variables, assuming that the names refer to distinct individuals.

In general, rules used for uncertain inference should be *stable* in the sense that their certainty variables should have fixed probabilities, rather than ones that shift in the course of a reasoning process, as a result of shifts in the certainties of various propositions affected by the reasoning process. Further, the overall effect of applying a set of rules should be independent of their order of application. (However, we found that we needed to relax that constraint somewhat in considering the interaction among rules leading to contrary conclusions.)

The intuitions behind our three types of rules are quite different; accordingly, they differ in the kinds of knowledge they can be used to encode, and in the way they update consequent probabilities (given the truth or at least the probability of their antecedents), i.e., their combinatory and chaining behavior.

4.7.1 a-Rules

Amplification rules capture a simple, intuitive notion of some event or predication justifying an increased belief in some other event or predication. For example, consider the following simple a-rule:

$(\forall x[[x\ dog] \Rightarrow (:a\ (p\ x)[x\ friendly]])$,

i.e., for any dog x, conclude with certainty $(p\ x)$ that x is friendly (in the absence of other information). Here $\forall$ is being loosely used not as a universal quantifier, but as an indication that x is a matchable rule variable.[5] Note also that we are using square brackets and predicate infixing for sentential formulas, and round brackets and functor prefixing for functional expressions. As noted above, $(p\ x)$ is a distinct Bernoulli variable for each value of x. Moreover, these Bernoulli variables are considered independent of one another and of the certainty variables associated with other a-rules. (As such they are *choice variables* in the sense of Poole (2008).)

We noted above that the Bernoulli random variables $(p\ Rover)$ and $(p\ Fido)$ are treated as independent of one another. However, we normally take the instantiated certainty variables of any given rule to have the same probability of truth. For example, using vertical bars to indicate numerical probabilities of random variables, we may write

$|(p\ x)| = .8$ for all x,

so that the rule in effect affirms that any arbitrarily selected dog is 80% likely to be friendly. Note that we would assume this rule to be grounded in an empirical claim, namely, that 80% of dogs are friendly. As explained earlier, once the rule has been instantiated, it no longer reflects any statistical fact, but is just a rule for assigning or

[5] We misuse '`all`' analogously for rules in the EPILOG inference engine, for convenience in the implementation.

updating the certainty of the consequent, given the antecedent. This kind of rule instantiation immediately provides a means of direct inference of probabilistically qualified conclusions – something unavailable in, for example, nonmonotonic reasoning methods (NMR) and MLN methods.

Direct inference applies only if no other knowledge has (yet) been applied to the conclusion. We started by saying that a-rules serve to increase belief in a conclusion; so suppose that the certainty of a conclusion has already reached some level p in a reasoning process, leaving remaining uncertainty $(1 - p)$. Then an a-rule with certainty variable q will add a fraction q of the remaining uncertainty $(1 - p)$ to the certainty p, with result $p + q.(1 - p)$. (To take account of a possible dependency of p on q, we replace multiplication of numbers by idempotent multiplication of algebraic probabilities below.) Note that a-rules can be arbitrarily weak or strong (with $0 < |q| < 1$), but a sufficient number of weak rules may push the certainty of a conclusion arbitrarily close to 1.

Also note that direct inference corresponds to applying an a-rule in a case where the prior probability of the conclusion is vanishingly small (i.e., with $p = 0$, the update to $p + q.(1 - p)$ is precisely q). We believe that ascribing near-0 probability to ground predications, in the absence of any relevant knowledge about the arguments of the predicate, is often very reasonable. We call this the *sparse truth assumption* (STA), in recognition of its similarity to the *closed world assumption* (CWA) often used in nonmonotonic reasoning. The plausibility of the STA can be appreciated, for example, by considering what proportion of entities in any general world ontology are *dogs*, or are *friendly*, or *serve alcoholic beverages* (see our previous examples)—surely, essentially 0, in view of real-world entities such as stars or grains of sand, let alone abstract ones like numbers.

The combinatory behavior of a-rules, apart from being intuitively natural, can be further motivated in the following way. If certain distinct facts provide *independent support* for a conclusion, then the certainty of the conclusion should behave like the probability that independent Bernoulli trials will yield "success", i.e., a positive outcome on at least one of the trials. This intuition leads to the *noisy-OR* rule of combination,

Pr(success) = $1 - (1-p)(1-q)(1-r)...$,

where $p, q, r, ...$ are the (magnitudes of) the choice variables of the rules whose antecedents independently lend support to the shared conclusion. (For 2 variables this is $p + q - p.q$.) A classical example is "Mr. Holmes' burglar alarm", which may be triggered by an intruder or an earthquake; certainly, if either (or both) of these events occurred, they independently suggest that the alarm may have been triggered (Pearl 1988:49-50). A key observation for our purposes is that the noisy-OR result can be obtained with separate rule applications, in any order; viz., as explained above, for a-rules with certainty variables $p, q, r, ...$, each rule simply adds an increment to the current certainty of the conclusion, where that increment is the choice-variable probability times the amount by which the current probability falls short of 1.0. We saw that starting with probability (nearly) 0 and applying two rules with certainty variables p, q, the resulting certainty is $p+q(1-p)$, and this is of course the same as the noisy-OR result $p+q-pq$.

As may already be clear from our examples and discussion, the general form of an a-rule is

$(\forall x_1...x_k\ [\phi \Rightarrow (:a\ (p\ x_1...x_k)\ \psi)])$,

presumed to involve (match) variables $x_1, ..., x_k$. The distinct values of the match variables correspond to distinct, independent Bernoulli random variables $(p\ x_1...x_k)$. In implementing a-rules, we represent a hypothesis H obtained by uncertain inference as

a certainty-modified statement such as $(:a\ p\ H)\}$. Treating this as an initial knowledge base Δ, we can represent the operation of adding another such statement about H based on another a-rule as $\Delta' = \Delta \cup \{(:a\ q\ H)\}$, where $Pr(H|\Delta') = p + q * (1 - p)$, and the '$*$' operator represents idempotent algebraic product, discussed below.

Noisy-OR belief combination has been widely used in applications where all the influences on the certainties of the propositions (domain variables) of interest are believed to be causal influences. For example, large 2-level BNs have been constructed in which diseases are roots (at level 1) and the "findings" at level 2 can be caused by one or more diseases (or by an independent unknown cause). It is hardly surprising that causal models have played such a prominent role in uncertain inference, since in some sense the quest for understanding the phenomena in the world around us, including illness, is a quest for discovering underlying causal mechanisms. As well, much evidence has been accumulated that human perception and cognition involve causal Bayesian inference (Gibson et al., 2013; Jacobs and Kruschke, 2011).

Now, some of the sample rules we have listed are arguably causal, even if the causal mechanisms are obscure. For instance, we could say concerning claim (1) that it is the intrinsic nature of dogs that generally causes them to be friendly; or, concerning (2), that the intrinsic nature of dogs and dog owners generally causes the owners of the dogs to treat them well. But we could hardly claim, concerning (3), that the intrinsic nature of new Ph.D. graduates causes them to be under 30 years old. Various other rules in our list resist a causal interpretation.

Nonetheless, we hypothesize that we can cast many generalizations as a-rules, even if they resist a causal interpretation. In a supplementary document (Schubert, 2017) we cast (3) as an a-rule and apply this in combination with another a-rule to the effect that a person with a young sibling is apt to be young as well. We find that the certainty

that a new Ph.D. with a 24-year-old sibling is under 30 aligns well with the results of a detailed generative model for age differences between siblings, and ages for attaining a Ph.D. The main evidence we bring in this paper for the efficacy of a-rules is the set of results for the "graduate students and their advisors" domain.

4.7.2 b-Rules

The best example of *b-rules* (Bayes rules) are rules that duplicate the effect of naïve Bayesian inference from evidence items to a hypothesis, where the evidence items are conditionally independent of one another, given the hypothesis. Naïve Bayesian inference is often used as an approximate method of uncertain inference in reasoning from effects to their causes, despite its weaknesses. For example, suppose that an effect has two possible alternative causes, and we learn that the effect is true, and update the probabilities of the causes accordingly. If we then learn that one of the causes is true, the probability of the other cause should drop – the effect has been "explained away", so that a second explanation is unnecessary or at least less likely. Naïve Bayesian inference fails to account for this phenomenon.

Nonetheless, naïve Bayesian inference is sometimes correct, so we would like to cast Bayesian rules in a form such as

$(\forall x[[x$ *has-runny-nose*$] \Rightarrow (:b\ (p\ x)[x$ *has-cold*$]])$.

Now, it is well-known that naïve Bayesian inference can be performed one evidence item at a time by using likelihood-ratio updating of the odds in favor of the conclusion (hypothesis) in question. For example, we could update the odds in favor of x having a cold by multiplying the prior odds of that eventuality by the likelihood ratio

$$\frac{Pr([x\ has-runny-nose]|[x\ has-cold])}{Pr([x\ has-runny-nose]|\neg[x\ has-cold]).} \tag{4.23}$$

Thus we could supply such likelihood ratios as b-rule parameters. However, for uniformity we wish to use probability parameters in all rules; so $(p\ x)$ should lie between 0 and 1. Therefore in general, instead of supplying $Pr(E|H)/Pr(E|\neg H)$ as b-rule parameter for a rule $(E \Rightarrow (:b\ p\ H))$, we instead use

$$p = Pr(E|H)/[Pr(E|H) + Pr(E|\neg H)].$$

It is easily verified that with this choice of parameter, $p/(1-p)$ is the likelihood ratio needed for updating the odds in favor of H. Note that applying a b-rule is thus quite different from applying an a-rule: Instead of using the certainty variable to *increment* the probability of the consequent, we are using it to *scale* the odds, and thus in effect the probability, of the consequent. This scaling may either raise or lower the probability of the consequent, depending on whether the likelihood ratio is greater or less than 1.

We have already pointed out that naïve Bayesian inference has problems with "explaining away". The way this shows up in formulating b-rules is that the p-parameter of a rule $(E \Rightarrow (:b\ p\ H))$, when there is also an alternative rule $(E \Rightarrow (:b\ p'\ H'))$, will involve the prior probability of H'. But since this probability is itself subject to change as a result of inference, the b-rule will no longer be stable – its parameter may well change as other inferences are made.

Because of this defect, b-rules seem to have limited applicability in general reasoning. Interestingly, by reliance on algebraic probabilities we could theoretically do *all* reasoning using b-rules. (This is a reflection of the fact that Bayes' rule is universally valid.) But such an approach would be equivalent to the general conditioning method for algebraic probabilities mentioned above it – and like that method, would require computing algebraic marginals for all propositions of interest.

4.7.3 c-Rules

Categorial rules are intended to encode conditional knowledge where truth of the antecedent redistributes likelihoods among a competing spectrum of alternative (and exhaustive) possibilities. For example, learning that a person that we know little about is a grandparent may shift our beliefs about the person's likely age category, and knowing that the person has a 30-year-old sibling may shift them in another direction. Other clear examples of this arise when reasoning with taxonomic information, e.g., in guessing whether the animal that chewed open some plastic garbage bags overnight is a dog, raccoon, skunk, bear, or something else. We write c-rules in the form (neglecting matchable predicate arguments for simplicity here),

$E \Rightarrow (:c\ q_1\ H_1\ q_2\ H_2\ ...q_k\ H_k)$.

Like b-rules, c-rules can be understood in terms of likelihood ratio updating. If disjoint hypotheses $H_1, H_2, ..., H_k$ each suggest the truth of E with some likelihood $Pr(E|H_i)$, $i = 1, ..., k$, then the prior probabilities of the H_i can be updated by multiplying them by those likelihoods, in effect forming a Hadamard product, which is then normalized by dividing by the sum of the individual product terms. Since we normalize after applying a c-rule, we could simply use $q_i = Pr(E|H_i)$ as the certainty variables, but we wish to view the q_i as a distribution over the H_i when we know only E. Therefore we normalize the q_i as $q_i = Pr(E|H_i)/\sum_j Pr(E|H_j)$. Algebraically, we treat the q_i as a spectrum of mutually exclusive Bernoulli random variables, independent of other such spectra and of certainty variables in a-rules. Thus, if the prior probabilities of $H_1, H_2, ..., H_k$ are $p_1, p_2, ..., p_k$ respectively, then with q_i as just defined, the updated probabilities, given E, are

$p'_1, p'_2, ..., p'_k$,

where $p'_i = p_i q_i / Z$, for $i = 1, 2, ..., k$, $Z = \sum_{i=1}^{k} p_i q_i$.

Notably, this update rule coincides with Dempster-Shafer updating when the D-S "frame of discernment" consists of disjoint sets and the prior distribution over those sets is uniform.

As in the case of a-rules, we find it generally appropriate to apply the STA, i.e., the H_i all have vanishingly small probabilities prior to application of known facts and rules (and a vanishingly small sum of these priors); but once E is affirmed, the posterior probabilities of the H_i add up to 1. In other words, the H_i categories are exhaustive, given E.

It is also worth noting the connection to b-rules. Any b-rule $E \Rightarrow (:b\ q\ H)$, where q = $Pr(E|H)/[Pr(E|H) + Pr(E|\neg H)]$ (see previous subsection) could be reformulated as a c-rule $E \Rightarrow (:c\ q\ H\ q'\ \neg H)$, where q' = $Pr(E|\neg H)/[Pr(E|H) + Pr(E|\neg H)]$. However, in this case the specified alternatives do not have vanishingly small probabilities before knowledge is applied – rather, $Pr(H)$ is near 0, and $Pr(\neg H)$ is near 1.

Because c-rules simultaneously take account of interactions among alternative conclusions, given the antecedent, they are much more broadly applicable than b-rules. In fact, they can be used in principle to reason "anti-causally" from effects to possible causes in BNs. For example, suppose that in a BN for, say, disease diagnosis, a certain symptom E has three possible causes A, B, C (and only these). Assume that they influence E in noisy-OR fashion. Then we can formulate a c-rule whose disjoint conclusions correspond to the possible combinations of truth values of A, B, C, so that there will be 7 categories (with at least one of A, B, C true):

$E \Rightarrow (:c\ q_a\ \{A\}\ q_b\ \{B\}\ q_c\ \{C\}\ q_{ab}\ \{AB\}\ ...\ q_{abc}\ \{ABC\})$.

where $\{A\}, ..., \{AB\}, ..., \{ABC\}$ respectively denote the hypotheses that *only* A is true, ..., the hypothesis that *only* A and B are true, ..., and finally that all three of A, B, C

are true. Writing the single-cause probabilities as $Pr(E|\{A\}) = u, Pr(E|\{B\}) = v, Pr(E|\{C\}) = w$, we can rewrite the c-rule in unnormalized form as

$$E \Rightarrow (:c\, u\, \{A\}\, v\, \{B\}\, w\, \{C\}\, u{+}v{-}uw\, \{AB\}\, ...\, 1{-}(1{-}u)(1{-}v)(1{-}w)\, \{ABC\}).$$

Having applied such rules, we can find individual posterior probabilities of causes A, B, C by marginalizing, e.g., adding the probabilities of the combinations $\{A\}$, $\{AB\}$, $\{AC\}$, $\{ABC\}$ in which A is true in order to obtain the posterior probability of A.

For example, such rules can completely model inference of root causes in any 2-level noisy-OR BN, given the truth of some of the symptoms. (Symptoms known to be false require separate rules.) Of course, the sizes of the c-rules grow exponentially with the number of causes a symptom can have. On the other hand, if all of the causes C have very low marginal probabilities compared to their single-cause probabilities $Pr(E|C)$ (i.e., the STA holds), then the logical combinations of causes where more than 1 or 2 are true may well have negligibly small probabilities, i.e., the symptom E is very unlikely to have more than 1 or 2 causes. In the above example, if we can neglect all but the first-order terms, the (unnormalized) rule just becomes

$$E \Rightarrow (:c\, u\, A\, v\, B\, w\, C),$$

i.e., it is as if A, B, C were mutually exclusive alternatives. We have not yet experimented with exact or approximate c-rules, but conjecture that together with a-rules they will provide a very powerful basis for general rule-based reasoning.

4.8 Inference with Our Proposed Rules

We have implemented inference with a-rules through extensions to a first-order theorem prover based on the language of Episodic Logic (Schubert and Hwang, 2000). Our inference algorithms are built upon extensions of logical deduction – though we emphasize that they are not strictly deductive rules, but instead use deductive reasoning machinery to combine evidence.

4.8.1 Rule Combination

Two fundamental challenges of reasoning with a-rules pertain to combining the effects of convergent rules (i.e., with a shared conclusion) and reasoning with the resultant uncertain conclusions. In general, we combine convergent a-rules with an algebraic implementation of noisy-OR. This means that the effect of an a-rule will always be to amplify the likelihood of its consequent, regardless of the magnitude of the corresponding algebraic probability. It is straightforward to do this mechanically for an arbitrary first-order formula ϕ:

From $(:a\ p\ \phi), (:a\ q\ \phi)$ infer $(:a\ (p\ +\ q\ -\ p*q)\ \phi)$

Since the product $p*q$ is algebraic, identical terms are multiplied idempotently, and mutually exclusive terms yield zero. The algebraic approach robustly allows for arbitrary a-rule combinations (even with themselves, as already noted).

The complementary problem of reasoning with uncertain conclusions is similarly tackled with a combination of deduction-style rules and algebraic probabilities. Suppose that the antecedent ϕ of a rule $\phi \Rightarrow (:a\ q\ \psi)$ becomes established with certainty p (algebraically expressed). Then, since the choice variable q is in effect conjoined with ϕ, the probability of the conclusion is $p*q$. In other words, rule chaining is a

matter of (idempotently) multiplying the antecedent probability by the choice-variable probability. Technically, we obtain this result with a chaining rule

From $\phi \Rightarrow \psi, (:a\ p\ \phi)$ infer $(:a\ p\ \psi)$,

where we allow ψ to be replaced by $(:a\ q\ \psi)$.

As an example, if most dogs are friendly animals, and most friendly animals make good pets, then a given dog is rather likely to make a good pet. Further, if the dog is known to be a labrador, and we have no more specific knowledge about the friendliness of labradors than that of dogs, we would still make the same inference.

4.8.2 Negative, Disjunctive, and Existential Rule Consequents

a-Rules with negative conclusions need to be distinguished from those with positive conclusions, since under the STA, the knowledge-free probability of a negative conclusion is near-1 rather than near-0. So we track a-rule applications leading to a positive predication separately from ones leading to its negation. We combine their effects by treating the negative conclusion as a "spoiler" for the positive conclusion; for example, if the positive conclusion receives probability p and the negative conclusion receives probability q, then the resulting probability is $p * (1 - q)$. As an intuitive example, sunshine and warm air might both suggest pleasant weather, but their combined effect will be spoiled if strong winds are present also, and tend to imply unpleasant weather.[6]

Reasonable handling of disjunctive rule-consequents depends on the presumed relationship among disjuncts. If they are disjoint, the appropriate formalization is in terms of c-rules. If they can be considered conditionally independent, given the antecedent, then it makes sense to split the rule, also splitting the choice variable p of the rule

[6]Here is a sanity check on our rule of combination: Suppose we regard the probability $(1 - p)$ of $\neg\phi$ suggested by the positive evidence for ϕ as evidence against ϕ. Substituting this for q in $p * (1 - q)$ gives back p – as it should!

into independent elementary probabilities $p_1, ..., p_k$, by default of equal magnitude, say x, satisfying $|p| = 1 - (1 - x)^k$ (so that the numerical probability of the disjunction with come out to $|p|$). If possible, however, one would use empirical data to assign approximate values to the $|p_i|$.

We have not yet adequately explored interactions between existentials and a-rules, either analytically or empirically. One possibility is Skolemization, with due attention to the fact that a Skolem constant may well share its denotation with a proper name ("*An AI professor* advises Bill, namely Mary"). There are also issues of the relative scopes of probabilistic qualification and existentials ("Probably an AI professor advises Bill", vs. "There is an AI professor who probably advises Bill".)

4.8.3 Obtaining Final Probabilities

Our algorithm for obtaining final probabilities relies on three stages: construction of the inference graph, deriving algebraic probability expressions from the graph, and then numerically evaluating the probabilities.

We construct the inference graph by forward reasoning. Since our empirical comparisons herein are concerned with finite Markov Logic Networks, we were able to use exhaustive forward inference.[7] This method allows us to distinguish evidence-based uncertainty (viz., when an a-embedding can be retrieved) and simple ignorance (retrieval failure). Also in contrast with Markov Logic, we can compute probabilities incrementally through local updates to the graph as new information is discovered.

Our algorithm does not descend recursively into quantified contexts, but because

[7] In future we may attempt using simplification of algebraic probabilities to detect convergence (as fixed points), and use "interestingness" and low-probability criteria to limit forward inference, as was done in a previous version of the EPILOG inference engine – which, however, computed probabilities in a very ad hoc way (Schubert and Hwang, 2000).

our inference algorithms are first-order, derivable relationships between quantified wffs will in principle be correctly accounted for. We leave fuller investigation of this topic to future work.

After construction of the inference graph (or a subset), we obtain algebraic probability expressions for arbitrary formulas based on a recursive querying procedure. If a formula (or its negation) is known to be true without probabilistic qualification, then the probability is simply 1 (or 0). If a formula is a literal, then we separately obtain the embedding a-statements for it and its negation, and combine them as outlined above. For conjunctions about which we have no direct knowledge, we form the algebraic product of the algebraic probability of each conjunct. Finally, for disjunctions, we form the algebraic noisy-OR of the algebraic probabilities of the disjuncts.

Ultimately, we numerically evaluate the determined algebraic probability of the formula in question. Suppose that in a complex algebraic probability, the α_i terms in every subexpression of form $\alpha_1 * \alpha_2 * ... * \alpha_n$ are built up from distinct elementary probability variables. Then the α_i terms are independent of one another, the idempotent products become ordinary products, and numerical evaluation reduces to simple numerical substitution and arithmetic evaluation. When the same elementary (or complex) probabilities occur in different operands of '$*$', then we must first eliminate these occurrences (e.g., by expanding the products), which can be computationally complex – in general, this process is NP-hard, as is proved by a reduction from 3SAT (Schubert, 2004). However, this can be improved for common cases through resolution-like simplifications and common factor extraction.

4.9 Looping back: Tweety the Chicken

Tweety the Chicken illustrates two important kinds of reasoning about taxonomic hierarchies / natural kinds. The classic nonmonotonic challenge posed by the Tweety problem is to derive that Tweety flies more like a chicken, than a bird, because the most *specific* knowledge we have about how Tweety flies is based on his chicken-ness.

Suppose that we there is also a Bird named Blue, and a Duck named Daffy. Then, lacking anything more specific about Ducks, we conclude that Daffy probably flies based on his being a bird, and of course Blue probably flies also based on being a bird. If we also believe there are no other kinds of birds beyond ducks and chickens, then because we know that Daffy is a not a chicken, it follows that Daffy is even *more likely* to fly than Blue.[8] However, in the general case there may also be other kinds of birds (i.e., Ostriches, Parrots, etc.), and it is no longer true that Ducks are more likely to fly than arbitrary birds – depending instead on specific values of the parameters.

Consider the following algebraic approach, based on a taxonomy of bird-kinds (bird, chicken, duck), and bird-features (flying or flightless).

1. $Pr(Flying|Chicken) = r$
2. $Pr(Chicken|Bird) = q$
3. $Pr(Flying|\neg Chicken) = s$
4. $Pr(Flying|Bird) = q * r + (1 - q) * s$
5. $Pr(Bird|Chicken) = 1$

[8] Assuming a finite but arbitrary total number of birds.

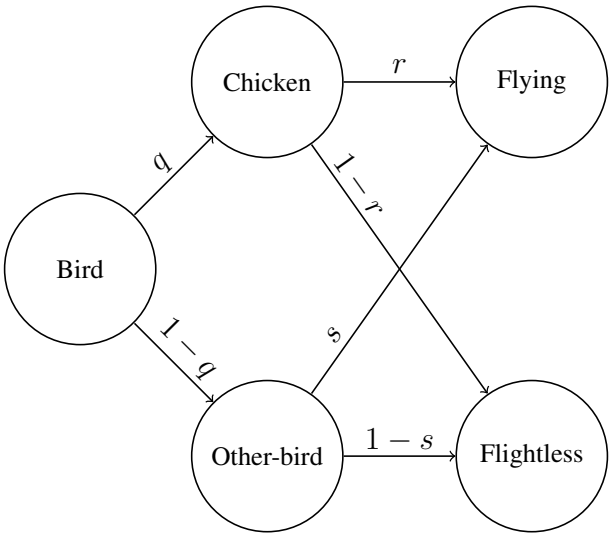


Figure 4.9: Chicken network.

Note that statements (1-3) are assumptions, and that statement (4) follows from those assumptions. Schubert (2004) proved the validity of the algebraic *conditioning formula*:

$$\frac{Pr^*(A_1) * \cdots * Pr^*(A_m) * Pr^*(C_1) * \cdots * Pr(C_n)}{Pr^*(C_1) * \cdots * Pr(C_n)} \tag{4.24}$$

Applying the conditioning formula yields:

$$Pr^*(Fly|Chicken \wedge Bird) = \frac{Pr^*(Fly) * Pr^*(Chicken) * Pr^*(Bird)}{Pr^*(Chicken) * Pr^*(Bird)} \tag{4.25}$$

From basic probability we have:

$$Pr^*(Fly) = qr + (1 - q) * s \tag{4.26}$$

And from assumption (V), and basic probability again, it follows:

$$Pr^*(Chicken) = q \tag{4.27}$$

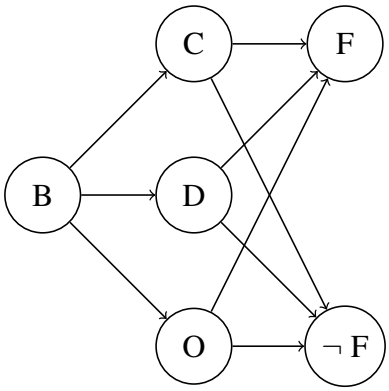


Figure 4.10: Duck network.

And because we are only discussing birds,

$$Pr^*(Bird) = 1 \tag{4.28}$$

Therefore,

$$\begin{aligned} Pr^*(Fly|Chicken \wedge Bird) &= \frac{[qr + (1-q) * s] * 1 * q}{1 * q} \\ &= qr \\ &= q \\ &= Pr^*(Fly|Chicken). \end{aligned} \tag{4.29}$$

Note that $s = Pr^*(Fly|\neg Chicken)$ factors out.

Let us also consider an expansion of the problem, as in Figure 4.10 where we have included kinds for Ducks and Other-birds. Then it also follows that $Pr(Fly|Duck \wedge Bird) = Pr(Fly|Duck)$, but no comparative relation is required to hold between $Pr(Fly|Duck) = h$ and $Pr(Fly|Bird) = qr + uh + vk$ – it depends on the specific proportions involved. This lack of conclusion is reasonable from a skeptical perspective. Instead of Ducks, which in the real world are quite likely to be fliers relative to

other kinds of birds, we could have substituted Penguins, which are very unlikely to fly. The structure of the model would not have changed, so if we *were* allowed the inference that Daffy is more likely to fly than Blue, then we could have (wrongfully) concluded the same thing about his less-famous friend – Opus the penguin.

All this discussion leaves the question, which of our a,b, c-rules (if any) are appropriate here? Let us consider each in turn, according to the principles outlined in Section 4.10, namely how each of the rules change the probability of an event, relative to the probability prior to applying the rule.

1. a-rules always increase or decrease the probability relative to the prior.
2. c-rules always boost one (or some) event(s) at the expense of another (or many) event(s), relative to the prior.
3. b-rules pull the probability "closer", by boosting when the prior is lower, and reducing when the prior is higher.

What about the necessary assumptions?

1. a-rules – the antecedent should be independent of other possible causes (if any) of the consequent.
2. c-rules – the antecedent should be independent of other possible effects of the consequences, the consequences are mutually exclusive and exhaustive (in cases where the antecedent holds), and a uniform prior assumed over the causes when no other prior knowledge is available.
3. b-rules – the antecedent should be conditionally independent (given the consequent) of other common causes (or b-rule antecedents) of the consequent.

We can easily rule out a-rules in this situation. Concerning the keystone assertion, that "Chickens don't normally fly", we certainly don't want to boost the probability of flying given that something is a chicken. While it does make some sense to attenuate the probability of flying, with a rule such as

$$[Chicken \implies (:a\ (1-r)\ \neg Fly],$$

this only captures a portion of the problem and neglects the broader picture. We immediately run into trouble with the "Birds normally fly" rule. It is clear that bird-ness and chicken-ness are not independent, therefore we must use some expression for the probability of birds flying. I.e., instead of

$$[Bird \implies (:a\ p\ Fly)],$$

we might write

$$[Bird \implies (:a\ (q * r\ +\ (1-q) * s)\ Fly)],$$

but this admittedly cumbersome. Worse, if we consider the case where ducks and other-birds are involved as well, we see that we immediately run *afoul*[9] of the mutual exclusion between natural kinds. In particular, we would have a non-zero probability of being both a duck and a chicken! Amplification rules are useful in causal situations, but for taxonomies, they take a backseat to category rules.

Let us therefore instead consider representing Figure 4.9 with category rules, where the knowledge base Δ is given by:

1. $(\forall\ x\ ((x\ Bird) \implies (:c\ (q\ x)\ (x\ Chicken)\ (-\ 1\ (q\ x))\ (xOther\text{-}bird))))$

[9] Sorry.

2. $(\forall\ x\ ((x\ Chicken)\ \implies\ (:c\ (r\ x)\ (r\ Flying)\ (-\ 1\ (r\ x))\ (xFlightless))))$

3. $(\forall\ x\ ((x$ *other-bird*$)\ \implies\ (:c\ (s\ x)\ (xFlying)\ (-\ 1\ (s\ x)(xFlightless)))))$

4. $(\forall\ x\ ((x\ Chicken)\ \implies\ (x\ \ Bird)))$

5. $(Tweety\ chicken)$

6. $(Blue\ bird)$

By the algebraic, pointwise, normalized product semantics of c-rules as described in Section 4.7, it follows that

$$Pr^*(Fly(Tweety)|\Delta) = r \tag{4.30}$$

and we may make the inference, rather than the assertion, that

$$Pr^*(Fly(Blue)|\Delta) = q * r + (1 - q) * s \tag{4.31}$$

Therefore, c-rules seem the most appropriate for dealing with the Tweety the chicken problem from a logical point of view, but they're based on an underlying algebraic probability framework which does not necessarily require c-rules at the bottom layer. From the nice specificity result protrudes one thorn: we must be able to express $Pr(Fly|Bird)$ using an independent parameterization, which means $Pr(Fly|Bird)$ and $Pr(Fly|Chicken)$ cannot both be (directly) variable parameters, and instead we need intermediate parameters expressing how likely a non-chicken-bird is to fly – although they do factor out from the final expression, they do *not* factor out from the $Pr(Fly|Bird)$ term, which means an assertion that $Pr(Fly|Bird)$ has some value (say, f), must instead be interpreted that an expression has that value – something like $q * r + (1 - q) * s$.

4.10 Looping back: The Nixon Diamond

Returning to the Nixon Diamond, as introduced in Section 4.3.1, let us consider how we can approach the problem using our proposed rules. One of the main things to consider is how one thinks the rule should impact prior knowledge – taking for granted that the kind of rule we're considering has a probability interpretable as the conditional probability of the target proposition.[10] Each of our a-, b-, and c-rules behave differently with respect to the change of probability relative to the prior:

1. a-rules are appropriate only if we think they should always boost the target probability, no matter how small the rule probability might be, and regardless of what knowledge has already been applied (modulo algebraic interactions – e.g,. if we've already applied the rule, it shouldn't boost the target if applied again.)

2. b-rules are appropriate only if we think that they should boost the target probability just in case the rule probability is greater than the knowledge-free prior (i.e., expected truth-frequency of the target), and lower it just in case the rule probability is less than the knowledge free prior (again, modulo algebraic interactions).

3. c-rules are appropriate only if we think that they should always boost the same alternative(s) in favor of the other(s), regardless of the prior probability of the target, and regardless of what rules have already been applied (again, modulo algebraic interactions). They give the correct target probabilities, when all we know is the antecedent truth, if we can reasonably assume a uniform knowledge-free prior distribution over the alternatives (so, two equal probabilities in the binary case).

[10] Given that all we know is the truth of the rule antecedent. In the case of a c-rule, the rule provides the conditional probability of each alternative, given –only – the truth of the antecedent.

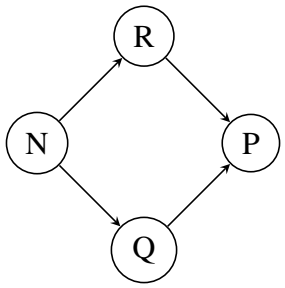


Figure 4.11: The Nixon diamond.

For the Nixon Diamond, we should first point out that a "skeptical" reasoner can't draw any conclusions, even if we do have a knowledge-free prior for the target (i.e., the proportion of Pacifists in the entire population under consideration). Echoing the first words of Section 4.4, any value of $Pr(Pacifist|Quaker \wedge Republican)$ is possible – the parameter is mostly free. See Figure 4.11.

But we're interested in more "credulous" approaches to reasoning (i.e., making additional assumptions). Now, NMR-style credulity leads to confident assertion that Nixon is or is-not a pacifist – we don't want *that* kind of unthinking credulity. Another option is entropy maximization, but that method considers the joint distribution over all variables involved – including those for Quaker and Republican, about no inferences are warranted. Instead, let us consider a Bayesian approach and for instance that pacifism is not that uncommon among Republicans, say 10%; and suppose that pacifism is virtually guaranteed for Quakers, say with probability 99.9%. Then it seems reasonable to expect $Pr(P|Q \wedge R) > 50\%$.

So, let's consider a, b, c-rules. We can eliminate a-rules from consideration, because a rule that, say 1% of Republicans are pacifists should not *boost* the probability of Nixon being a Republican – unless perhaps 1% is actually higher than the population average (i.e.,that Republicans aren't often pacifists, but nonetheless are more often so than the population average).

Next, we can eliminate c-rules, because by observation (3) applied to a binary c-rule, such a rule will always lower the probability of the alternative with probability $< 50\%$. But suppose, for example, that our rules are that 20% of Buddhists are pacifists and 30% of vegans are pacifists (where both of these percentages are above the norm). Then surely a vegan Buddhist is more than 30% likely to be a pacifist; but a pair of c-rules will say that the probability is $.2(.3)/[.2(.3) + .8(.7)]$, or 9.68%!

That leaves b-rules. Such rules need a knowledge-free prior, to be applicable at all. So, let's assume that the knowledge-free prior of pacifism is known, and let $Pr(Pacifist) = p$, $Pr(Pacifist|Quaker) = q$, and $Pr(Pacifist|Republican) = r$. According to the usual presumption of the Nixon diamond, suppose q is high and r is low.

Another significant assumption necessary for b-rules to give exact results is the conditional independence assumption between Republicanism and Quakerism, given that pacifism is known. I.e., that $Q \perp\!\!\!\perp R|P$. This is akin to the assumptions made by Naïve Bayes classifiers, and is non-causal. Note that the relative frequencies may be very different, as long as they are not highly correlated for individual samples of the population. Then we can show that:

$$Pr(P|Q,R) = \frac{qr}{qr + (1-q)(1-r)\frac{p}{1-p}} \tag{4.32}$$

Then the rule based on $Pr(P|Q) = q$ boosts this combined probability if $q > Pr(Pacifist)$, and the rule based on $Pr(P|R) = r$ lowers the combined probability if $r < Pr(Pacifist)$. We feel that this behavior is intuitively reasonable, and that b-rules can provide an answer to the question Nixon's pacifism.

As an interesting aside, if one takes account of the same prior for $Pr(P)$ in a c-rule approach, the result is $qr/[qr + (1-q)(1-r)(1-p)/p]$, which is nearly the same as

the result for b-rules but the term $p/(1-p)$, which is the prior *odds* of pacifism[11] has been inverted. Unless the prior happens to have value 1/2, this leads to bad results as already noted. The results are also bad for the "vegan Buddhist" variant.

Returning to our broader point, the fundamental differences between the update effects of a, b, c-rules presumably arise from different implicit assumptions about (conditional) independence. It seems that conditions (1-3) above can provide guidance (to a "credulous" reasoner) even when it's hard to intuit independence assumptions that would justify one type of rule or another. For example, in the Nixon diamond, we don't have clear intuitions whether or not it's reasonable to regard Quakers and Republicans to be independently distributed,given we're looking at pacifists, or similarly given that we're looking at non-pacifists. I.e., suppose one has a certain (very low) degree of belief that a known pacifist is a Republican. Will that probability drop further if one learns that s/he is a Quaker? Not necessarily – Republicans may not be proportionately rarer among pacifists in general than they are among pacifist Quakers – but we lack intuitions on that.

What of our other assumption, that we know the prior $Pr(Pacifist)$? If we don't know the prior, then we have to ask, where it is likely to lie compared to the rule probabilities at hand? If we think of $Pr(P|Q) = q$ as a rule that should boost the probability of being a pacifist, and of $Pr(P|R) = r$ as a rule that lowers the probability of being a pacifist, then we're in effect (under a b-rule approach) assuming that $q > p > r$. If the Quaker and Republican rules are all we have, we might assume that $Pr(Pacifist)$ lies halfway between q and r; this would at least assure the right direction of change if either premise happens to apply to some individual. But we might have further rules bearing on pacifism, and if there are any known rules that we regard as providing posi-

[11] We discuss the odds-likelihood approach in Section 4.4.1.

tive evidence for pacifism have rule probabilities lower than r, then we should shift our estimate downward, to be less than that lower value.

None of this addresses causal prevention (attenuation) but that's because, as noted, causal rules are inappropriate for the Nixon diamond, as long as we regard $Pr(P|Q) = q$ and $Pr(P|R) = r$ as rule probabilities. Essentially, a-rules are based on assuming a "causally complete" model of the propositions in question, where each rule gives a "single-cause" probability, and the probability of the conclusion is 0 if all rules impinging on that conclusion have false antecedents. Under a sparse truth assumption, the rule probabilities are also in effect single-cause conditional probabilities. See Section 4.4.1 for more details.

If we also have causal interference (prevention), things get difficult. In Section 4.7, there is an asymmetry in the order of rule application when presented with both amplification and attenuation of a common formula. We choose to apply prevention rules last (to avoid reducing an already infinitesimal sparse truth), but that isn't necessarily always right. For instance, coming back to the "flu prevention" example, suppose the flu is caused either by exposure by touching infected surfaces or by virus transmission by breathing infected air (i.e., someone sneezing or coughing). Then hand-washing might combat the former cause, while keeping your distance from visibly infected people might combat the latter. A flu shot might, to some extent, combat infection from both possible causes. See Figure 4.12. So one can't just make an assumption of "across-the-board prevention", for any preventative measure that may do some good. Rather, some measures combat some specific positive causes, while others combat other causes. An across-the-board prevention assumption will in general give too lower resultant probabilities – instead, we need nuanced models that say what preventative measures combat what possible causes.

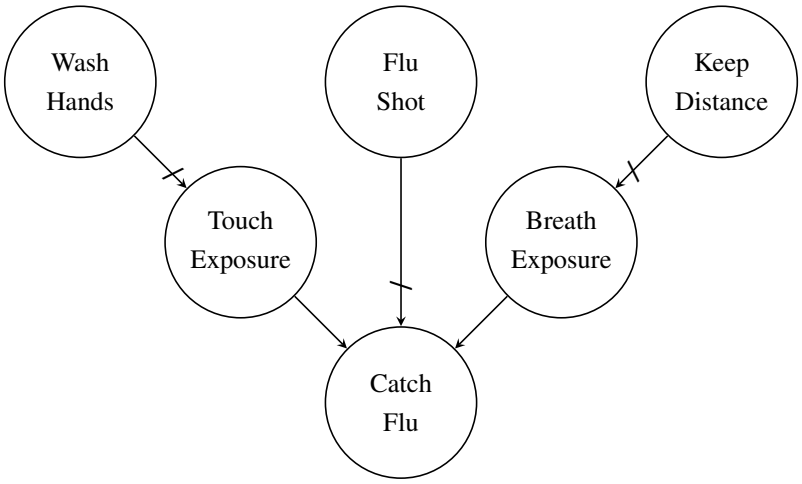


Figure 4.12: Nuanced causal prevention

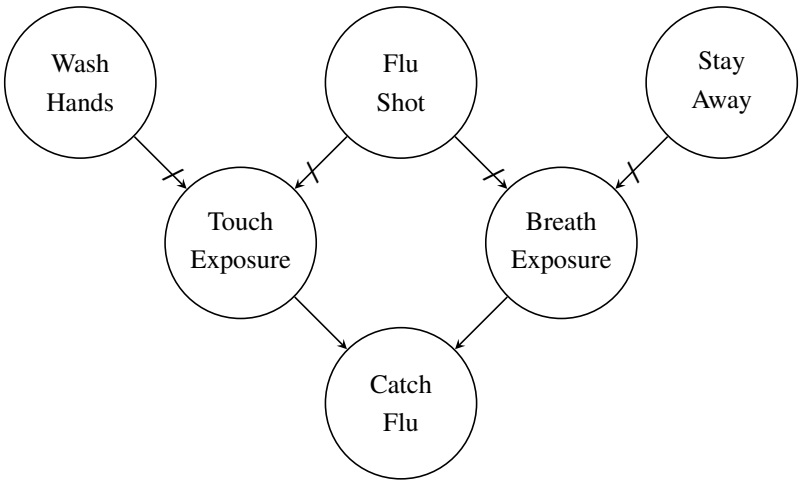


Figure 4.13: Nuanced causal prevention, alternate figure.

4.11 Experiments

In this section, we present empirically obtained evidence in support of the validity of our method by predicting advisor-advisee relationships over a moderately large database of facts about a computer science department. This dataset was introduced to the community by Richardson and Domingos (2006), who provide a strong baseline using Markov Logic; hereafter, we refer to it as the UW-CSE dataset.

A Markov logic network (MLN) is a set of weighted first-order statements. Inference in MLNs consists of constructing a large Markov random field with nodes for each ground predication. The probability of a model (assignment of truth to all ground formulas) is given by a normalized exponential weighting function. Inference of the probability of formula is performed by marginalizing over all worlds wherein the formula is true. Markov logic is a widely-studied field with many applications. Many algorithms exist for inference and learning, and most rely on some form of statistical sampling. Exact inference in Markov logic is #P complete in the size of the domain.

In this section, we compare accuracy and computational demands of our model that obtained with the original MLN. In the following years, structure learning and improved inference algorithms boosted the performance of Markov logic networks on this dataset; however, we assess our method with respect to the original work.

4.11.1 Setup

The UW-CSE dataset consists of two main components:

- A database of ground facts describing academic relationships among professors, students, courses, and publications within a computer science department. The set of terms ranges over persons, publications, and courses. The set of predicates

include type information (person vs publication), publication authorship, course instructors, teaching assistant roles, and advisor-advisee relationships. The complete dataset includes 1320 constants and 34 predicates and relations. The constants are partitioned into non-overlapping subsets by sub-area of computer science (ai, systems, theory, languages, and graphics.)

- The UW-CSE knowledge base is provided in two formats. At its core is a first-order knowledge base constructed by surveying members of the department for simple natural language statements describing the department which was then semi-manually edited for first-order syntax. The dataset also includes an MLN representation of the same first-order knowledge base. The baseline Markov logic network includes 82 rules.

To obtain baseline results, we use the MLN supplied by the authors of the dataset as well as the publicly available Alchemy implementation.

To apply our method, we converted all first-order implications found in the supplied knowledge base into amplification rules. Because MLNs make a unique names assumption, we add an additional $O(n^2)$ predications to each subset enforcing that assumption (using negated SameName, SameCourse, SamePublication, etc., predicates). This expansion of the database is largely responsible for computational complexity in our method as applied to the dataset. As a way of minimizing the number of rules used in evaluating our method, we omitted all first-order statements that do not immediately involve the target relation. This leaves us with three rules regarding co-teaching and co-publishing.

- $(\forall s, c, p, q$
  $(((s\ Phase\ postQuals)\ \wedge\ (c\ TaughtBy\ p\ q)\ (c\ TA\ s\ q)\ \neg(c\ CourseLevelLevel 100)$
  $\Rightarrow (:a\ (P1\ s\ c\ p\ q)\ (s\ AdvisedBy\ p)))))$

- $(\forall s,\ c,\ p,\ q$
  $(((s\ Phase\ postGenerals) \wedge (c\ TaughtBy\ p\ q)\ (c\ TA\ s\ q)\ \neg(c\ CourseLevelLevel 100))$
  $\Rightarrow (:a\ (P2\ s\ c\ p\ q)\ (s\ AdvisedBy\ p))))$

- $(\forall t,\ p,\ s$
  $(((t\ Publication\ p) \wedge (t\ Publication\ s)\ \neg(p\ SamePerson\ s)\ (p\ Professor)\ (s\ Student))$
  $\Rightarrow (:a\ (P3\ t\ p\ s)\ (s\ AdvisedBy\ p))))$

We obtain numerical parameters for our a-rules by simply counting conditional frequencies in the dataset. We perform inference using our model as outlined previously in this paper. Following Richardson and Domingos, we report area under the precision-recall curve (AUC) results obtained by cross-validation across the subareas of the dataset in Table 4.2.

4.11.2 The Denormalization Problem

While we are able to convert from some implicative form rules to our probabilistic rules, the conversion from CNF rules to our rules is more difficult, because exponentially many implications may share the same CNF representation.

For a Horn KB, we can easily side-step this issue by assigning positive literals to the conclusion, and negative literals to the antecedent. However, this approach is inadequate for non-Horn KBs such as the UW-CSE KB.

Consider a rule such as "If X and Y copublish, and they are not the same person, then X may advise Y or Y may advise X." In sentential form, this becomes $\neg(SamePerson(X,Y)) \wedge CoPublish(X,Y) \implies AdvisedBy(X,Y) \vee AdvisedBy(Y,X)$. The clause form is $S(X,Y) \vee \neg CP(X,Y) \vee AB(X,Y) \vee AB(Y,X)$, and so the simple polarity based transformation would build the rule: $CP(X,Y) \implies SP(X,Y) \vee$

Setup	NRules	Learn (m)	Infer (m)	AUC
Full MLN	82	9.1	6.5	0.320
Small MLN	3	0.5	0.1	0.227
a-rules	3	45	122	0.370

Table 4.2: AUC Results predicting Advisor-Advisee Relations
Computation time and mean area under the precision-recall curve when predicting the AdvisedBy relation between pairs of people. Generative learning was used for the MLN setups. All experiements were performed using the same system, and with minimal external load. The system has a core i7-2600K Intel processor and 16GB available RAM. The MLN uses 14GB ram while our method uses less than 2GB.

$AB(X,Y) \vee AB(Y,X)$. It is clear that the causal information has not been preserved through the denormalization process.

For standard first-order reasoners this is not an issue. However, because we make decisions based on an implied causal arrow inside our material implication arrow, this poses a significant challenge.

4.11.3 Results

Using our method, we obtain more accurate recovery of advisor-advisee relationships than the baseline MLN method; however, this comes at a significant computational expense. As noted this expense stems from the explicit world closure predicates (e.g., SamePerson, SameCourse, SamePublication). On average, this causes our stored database to grow from approximately 250 type predications to 8200 per subset. In principle we could obtain a dramatic speedup by using a sparse representation for some predicates. We also emphasize that unlike MLNs, our method is well-suited for open domains. Similarly, our parameter learning method is based on a simple recursive descent computation, which can in principle be performed much more efficiently through careful use of hashtables and conjunct ordering.

5 Conclusion

5.1 Concluding Remarks

Throughout this dissertation, we explored a variety of topics related to reasoning from natural language. The timeline of the work roughly parallels a pipeline from language to logic.

Our early work focused on improving the quality of logic extracted from text. As our translation step was (and remains) accomplished through compositional rules applied to treebank-style parses, obtaining an accurate statistical parser was imperative. While phrase-level accuracies for statistical parsers were already quite high, the high number of phrases per typical sentence led to frequent errors, which in turn led to malformed and undesirable logical forms.

A great variety of methods for parsing were hotly competing, including both discriminative and generative models, and parsers based on latent refinements of the underlying grammar. Contemporaneous with our work, connectionist parsers were beginning to perform competitively as well. We tackled the problem from two different perspectives: as a learning problem and as a programming program.

As an attempt from the machine learning perspective, we proceeded with a simple idea that short sentences should be easier to parse than long sentences. (They have fewer phrases, after all.) We wondered whether they could be used to bootstrap a parser to more reliable accuracies at the sentence level. Because of our interest in bootstrapping, the self-training technique seemed the most promising. Self-training is straightforward to describe and implement: train a parser on its own output. This technique was originally found to be unsatisfactory (Charniak, 1997), but recent papers (published two years before we began the project) indicated that self-training actually *worked* with a particular combination of a generative parser and a discriminative reranker. Therefore, we chose to narrow our work to focus on the same parser. We were looking at the process of self-training itself, hoping for something like a simple greedy boosting algorithm which would vastly improve the quality of parses, and therefore, our knowledge base. Of course, the problem was more subtle and challenging than expected. We experimented for approximately two years with various setups, looking to produce a regression model to pick "the best" sentences to use for self-training, which we considered a generalization of the short-sentences method. In retrospect, we spent too much time looking for a regression model when we should have been working from a broader perspective. After failure to produce significant results, the work was abandoned for some time to focus on the remaining tasks of pattern transduction and knowledge representation. However, after returning nearly a year later, we noticed a subtle pattern in the data.

We observed that the semi-supervised training data should have a similar distribution to the real data. Training on exclusively short sentence had caused severe overfitting to those short sentences, and performance was therefore degraded on the vast majority of normal length sentences. We had therefore found the opposite of the rule

we expected: *rejecting* short (and very long) sentences tended to improve performance on real data. We observed a similar effect when considering sentences by bigram novelty. By filtering out outlier sentences, we were able to realize gains from larger sets of semi-supervised data, and effectively pushed the asymptote just a bit higher – beyond the state of the art. After more distance and more consideration, we also have another very simple recommendation: random restart. One of the challenges in this work was the surprisingly high variance in parser accuracy when different samples of semi-supervised data were used for training. Instead of fighting against it, we suggest using the variance to one's advantage, and incorporating a random restart step during self-training to optimize performance on held-out data. In the future, we wish to explore a combination of these two rules of thumb, and expect it will push the asymptote slightly higher, perhaps even with reduced computational effort (and data) during training.

We worked in parallel from a programmatic, utilitarian point of view as well. We developed the TTT language in order to solve numerous small but burdensome problems. We were able to successfully repair some of the more systematic parse errors, but only when doing so using hand-authored patterns. (Part of the motivation in developing TTT were that there already were many such patterns, implemented in numerous scraps of Lisp.) We originally proposed to investigate learning transductions for parse repairs, but after exploring the tree-transduction literature more deeply we decided to move on from TTT to the bigger picture problems of commonsense reasoning. TTT has subsequently been used by other researchers as well for knowledge extraction and sharpening.

Finally, we began working exclusively on the knowledge representation problem in the winter of 2015. Our objective was to provide an implementation for probabilistic reasoning within Episodic Logic, in a way that would support such diverse tasks as

story-understanding, planning, and general feature extraction for jointly reasoning with computer vision (e.g., interpreting captioned photos). We discovered a vast and detailed literature on various attempts to unite probability and first-order logic, but we found each method to be lacking in some degree or another. We explored work ranging from classical path-based methods of nonmonotonic reasoning to recent work on Markov Logic Networks and various approaches to object-oriented Bayesian networks. We tried to keep reasoning from natural language in mind, due to our interest in general commonsense reasoning.

The first tangible progress was our idea to represent probabilistic information with a modal "certainy" quantifier, fulfilling a role much like an adverb. Sentences such as, "John is tall, with certainty 90%" became within reach, but still we struggled to find a uniform method of evidence combination that could cope with the wide variety of such sentences. After many lengthy discussions, we conceptualized our approach as involving three simple kinds of rules for probabilistic reasoning. We dissected the problem of evidence combination, keeping in mind classic challenges (e.g., Tweety and Nixon) and more recent ones (photo captioning, commonsense reasoning), and settled on the use of independent algebraic parameterizations of the world. We founded our rule types upon conditional probabilities along with specific (but implicit) independence relations. Because of the noisy and-or-not nature of algebraic probabilities, algorithms for algebraic simplification and numerical calculation bear a strong resemblance to resolution based theorem proving. We were therefore able to construct new modal contexts for episodic logic which contained algebraic probability expressions, but were manipulated with the same resolution-based rule instantiation mechanism for inference. This immediately gave us lifted inference, and a full proof theory (provenance). So far we have only been able to empirically validate one of our three kinds of rules but we are strongly

interested in continuing the work to include an efficient unified implementation and bigger-picture applications.

Bibliography

Bacchus, Fahiem, Adam J Grove, Joseph Y Halpern, and Daphne Koller. 1993. Statistical foundations for default reasoning. *IJCAI*.

Bacchus, Fahiem, Adam J Grove, Joseph Y Halpern, and Daphne Koller. 1996. From statistical knowledge bases to degrees of belief. *Artificial intelligence*, 87(1):75–143.

Black, E., S. P. Abney, D. Flickinger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. P. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings 4th DARPA Speech and Natural Lang. Workshop*, pages 306–311. Morgan Kaufmann, Pacific Grove, CA.

Blunsom, Phil, Trevor Cohn, Chris Dyer, and Miles Osborne. 2009. A gibbs sampler for phrasal synchronous grammar induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 782–790. Association for Computational Linguistics, Stroudsburg, PA, USA.

Boyd, Stephen and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

Brachman, Ronald and Hector Levesque. 2004. *Knowledge representation and reasoning*. Elsevier.

Charniak, Eugene. 1995. Parsing with context-free grammars and word statistics. Technical Report CS-95-28, Brown University.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180. Association for Computational Linguistics, Ann Arbor, Michigan.

Chiang, David. 2004. *Evaluation of Grammar Formalisms for Applications to Natural Language Processing and Biological Sequence Analysis*. Ph.D. thesis, University of Pennsylvania.

Chiang, David. 2010. Learning to translate with source and target syntax. In *Association for Computational Linguistics*.

Čmejrek, Martin and Bowen Zhou. 2010. Two methods for extending hierarchical rules from the bilingual chart parsing. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 180–188. Association for Computational Linguistics, Stroudsburg, PA, USA.

Collins, Michael. 1997. Three generative, lexicalised models for statistical parsing. In *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23. Morgan Kaufmann, San Francisco.

Comon, H., M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. 2007. Tree automata techniques and applications. Available on: *http://www.grappa.univ-lille3.fr/tata*. Release October, 12th 2007.

Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 6(8):451–455.

Eidelman, Vladimir, Chris Dyer, and Philip Resnik. ???? Language model and grammar extraction variation in machine translation.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*, pages 273–280. Boston.

Gécseg, F. and M. Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

Genet, Thomas and Valrie Viet Triem Tong. 2003. Timbuk – a tree automata library.

Getoor, Lise and John Grant. 2006. Prl: A probabilistic relational language. *Machine Learning*, 62(1-2):7–31.

Gibson, Edward, Leon Bergen, and Steven T. Piantadosi. 2013. Rational integration of noisy evidence and prior semantic expectations in sentence interpretation. *Proceedings of the National Academy of Sciences*, 110(20):8051–8056.

Gildea, Daniel. 2001. Corpus variation and parser performance. In *2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 167–202. Pittsburgh, PA.

Gildea, Daniel. 2012. On the string translations produced by multi bottom-up tree transducers. *Computational Linguistics*.

Gildea, Daniel and Daniel Štefankovič. 2007. Worst-case synchronous grammar rules. In *Proceedings of the 2007 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-07)*, pages 147–154.

Gillick, Dan. 2009. Sentence boundary detection and the problem with the u.s. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, NAACL-Short '09, pages 241–244. Association for Computational Linguistics, Stroudsburg, PA, USA.

Gordon, Jonathan and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 25–30. ACM, New York, NY, USA.

Graehl, Jonathan and Kevin Knight. 2004. Training tree transducers. In *Proceedings of the 2004 Meeting of the North American chapter of the Association for Computational Linguistics (NAACL-04)*.

Graff, David. 1995. North american news text corpus. *Linguistic Data Consortium*.

Gribkoff, Eric, Guy Van den Broeck, and Dan Suciu. 2014. Understanding the complexity of lifted inference and asymmetric weighted model counting. *CoRR*, abs/1405.3250.

Griswold, Ralph E. 1978. A history of the snobol programming languages. *ACM Sigplan Notices*, 13(8):275–308.

Grove, Adam J, Joseph Y Halpern, and Daphne Koller. 1994. Random worlds and maximum entropy. *Journal of Artificial Intelligence Research*, 2:33–88.

Huang, Zhongqiang, Mary Harper, and Slav Petrov. 2010. Self-training with products of latent variable grammars. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 12–22. Association for Computational Linguistics, Stroudsburg, PA, USA.

Hudak, Paul, John Peterson, and Joseph Fasel. 1999. A gentle introduction to haskell 98.

Jacobs, Robert A and John K Kruschke. 2011. Bayesian learning theory applied to human cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(1):8–21.

Jaynes, Edwin T. 1957. Information theory and statistical mechanics. *Physical review*, 106(4):620.

Kasami, T. 1965. An efficient recognition and syntax analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, MA†.

Kazemi, Seyed Mehran, Angelika Kimmig, Guy Van den Broeck, and David Poole. 2016. New liftable classes for first-order probabilistic inference. *CoRR*, abs/1610.08445.

Knight, Kevin. 2007. Capturing practical natural language transformations. *Machine Translation*, 21:121–133.

Koller, Alexander and Stefan Thater. 2010. Computing weakest readings. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 30–39. Association for Computational Linguistics, Stroudsburg, PA, USA.

Lavie, Alon, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*, SSST '08, pages 87–95. Association for Computational Linguistics, Stroudsburg, PA, USA.

Levy, Roger and Galen Andrew. 2006. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *LREC 2006*.

MacCartney, Bill and Christopher D Manning. 2014. Natural logic and natural language inference. In *Computing Meaning*, pages 129–147. Springer.

MacKay, David J. C. 2002. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA.

Maletti, Andreas. 2008. The power of extended top-down tree transducers. *SIAM J. COMPUT.*

Maletti, Andreas. 2010. Why synchronous tree substitution grammars? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 876–884. Association for Computational Linguistics, Stroudsburg, PA, USA.

Marcus, G. 2018. Deep Learning: A Critical Appraisal. *ArXiv e-prints*.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

May, Jonathan and Kevin Knight. 2008. A primer on tree automata software for natural language processing.

McClosky, David, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 152–159. Association for Computational Linguistics, Stroudsburg, PA, USA.

McClosky, David, Eugene Charniak, and Mark Johnson. 2008. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 561–568. Association for Computational Linguistics, Stroudsburg, PA, USA.

McClosky, David, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 28–36. Association for Computational Linguistics, Stroudsburg, PA, USA.

Milch, Brian, Bhaskara Marthi, Stuart Russell, David Sontag, Daniel L Ong, and Andrey Kolobov. 2007. Blog: Probabilistic models with unknown objects. In *Statistical relational learning*.

Mylonakis, Markos and Khalil Sima'an. 2010. Learning probabilistic synchronous cfgs for phrase-based translation. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 117–125. Association for Computational Linguistics, Stroudsburg, PA, USA.

Norvig, Peter. 1992. *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Pearl, Judea. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Petrov, Slav, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 433–440. Association for Computational Linguistics, Stroudsburg, PA, USA.

Poole, David. 2008. The independent choice logic and beyond. In *Probabilistic inductive logic programming*, pages 222–243. Springer.

Post, Matt and Daniel Gildea. 2009. Bayesian learning of a tree substitution grammar. In *Proc. Association for Computational Linguistics (short paper)*. Singapore.

Purtee, Adam and Lenhart Schubert. 2012. Ttt: a tree transduction language for syntactic and semantic processing. In *Proceedings of the Workshop on Applications of Tree Automata Techniques in Natural Language Processing*, ATANLP '12, pages 21–30. Association for Computational Linguistics, Stroudsburg, PA, USA.

Purtee, Adam and Lenhart Schubert. 2016. (abstract) simple rules for probabilistic common sense reasoning. In *Proceedings of the 2016 Workshop on Argument Strength*, pages 26–38.

Purtee, Adam and Lenhart Schubert. 2017. Simple rules for probabilistic commonsense reasoning. *Proceedings of the 2017 meeting of Advances in Cognitive Systems*.

Raedt, Luc De, Angelika Kimmig, and Hannu Toivonen. 2007. Problog: A probabilistic prolog and its application in link discovery. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2462–2467.

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Conference on Empirical Methods in Natural Language Processing*.

Reynar, Jeffrey C and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the fifth conference on Applied natural language processing*, pages 16–19. Association for Computational Linguistics.

Richardson, Matthew and Pedro Domingos. 2006. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

Riezler, Stefan, Tracy H King, Ronald M Kaplan, Richard Crouch, John T Maxwell III, and Mark Johnson. 2002. Parsing the wall street journal using a

lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 271–278. Association for Computational Linguistics.

Sankaran, Baskaran, Gholamreza Haffari, and Anoop Sarkar. 2011. Bayesian extraction of minimal scfg rules for hierarchical phrase-based translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 533–541. Association for Computational Linguistics, Stroudsburg, PA, USA.

Schubert, Lenhart. 2002. Can we derive general world knowledge from texts? In *Proceedings of the second international conference on Human Language Technology Research*, pages 94–97. Morgan Kaufmann Publishers Inc.

Schubert, Lenhart. 2017. Supplement for rules for probabilistic reasoning. `http:www.cs.rochester.edu/~apurtee/cogsys-supplement.pdf`.

Schubert, Lenhart K. 2004. A new characterization of probabilities in bayesian networks. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 495–503. AUAI Press.

Schubert, Lenhart K. and Chung Hee Hwang. 2000. Episodic logic meets little red riding hood: a comprehensive natural representation for language understanding. In Lucja M. Iwańska and Stuart C. Shapiro, editors, *Natural language processing and knowledge representation*, pages 111–174. MIT Press, Cambridge, MA, USA.

Selman, Bart and Hector J Levesque. 1989. The tractability of path-based inheritance. In *Proceedings of IJCAI*.

Shieber, Stuart M. 2004. Synchronous grammars as tree transducers. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+ 7)*, pages 88–95. Vancouver, BC.

Shortliffe, Edward H and Bruce G Buchanan. 1975. A model of inexact reasoning in medicine. *Mathematical biosciences*, 23(3-4):351–379.

Stein, Lynn Andrea. 1990. *A preference-based approach to inheritance*. Brown University, Department of Computer Science.

Touretzky, David S. 1986. *The mathematics of inheritance systems*, volume 8. Morgan Kaufmann.

Van Durme, Benjamin, Phillip Michalak, and Lenhart K Schubert. 2009. Deriving generalized knowledge from corpora using wordnet abstraction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 808–816. Association for Computational Linguistics.

Van Durme, Benjamin and Lenhart Schubert. 2008. Open knowledge extraction through compositional language processing. In *Proceedings of the 2008 Conference on Semantics in Text Processing*, pages 239–254. Association for Computational Linguistics.

Wolfram Research, Inc. 2017. Mathematica, Version 11.1. Champaign, IL, 2017.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of ACL-01*. Toulouse, France.

Younger, Daniel H. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10:189–208.

Zhang, Hao, Daniel Gildea, and David Chiang. 2008a. Extracting synchronous grammar rules from word-level alignments in linear time. In *COLING-08*, pages 1081–1088. Manchester, UK.

Zhang, Min, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008b. A tree sequence alignment-based tree-to-tree translation model. In *In Proceedings of ACL*.

A Probabilistic Parameter Independence

Definitions

A *well-formulated probabilistic knowledge base* (WFPKB) must be both logically consistent and probabilistically consistent. A KB is *logically consistent* if there exists at least one satisfying interpretation, where an interpretation assigns truth and falsity to Boolean literals, enumerates functional mappings, and assigns truth and falsity to all propositions.[1] A KB is *probabilistically consistent* if a joint distribution exists over all elements of the sample space, which is the set of all possible interpretations, such that the joint distribution is in turn consistent with the set of probabilistic constraints induced by the KB.

A *probabilistic parameter term* (PPT) over Boolean variables is either the marginal probability of a conjunction of (possibly negated) variables, the probability of a disjunction of (possibly negated) variables, or the conditional probability of one set of (possibly negated, conjoined, or disjoined) variables given a different set (also possibly negated, conjoined, or disjoined) variables.[2]

A set of PPTs is *independent* if for every assignment of values to the terms from the open interval $(0, 1)$ there exists a joint probability distribution over the interpretations of the Boolean variables involved in the set for which the value of each PPT assigned by the distribution matches the chosen assignment from the open intervals. A set of independent PPTs is called an *independent parameterization* (IP) of the corresponding Boolean interpretations.[3]

---

[1] Boolean satisfiability is NP-Complete. First-order SAT and Horn SAT are only semi-decidable.(Brachman and Levesque, 2004, p .67)

[2] It would be interesting to relate the definition of PPTs to conjunctive (or disjunctive) normal form. Framing the definition with a context-free grammar would be a concise way to generalize to the case of N variables, but it would also be problematic because of the significant semantic repetition. E.g., while $Pr(A \wedge A \wedge A)$ is syntactically valid, it seems undesirable to consider because it is exactly equivalent (logically and probabilistically) to a more "normal" representation as $Pr(A)$. The issue is not trivial though, because the latter is also equivalent to $1 - Pr(\neg A)$.

[3] Values chosen from interior of unit hypercube.

An IP is *maximal* if no other PPTs involving the relevant variables may be added to it without breaking the independence. An IP is *complete* if exactly one consistent joint distribution over the relevant variables exists for each assignment of values to the PPTs from the open interval $(0, 1)$. An incomplete IP may be completed through entropy maximization or by making additional assumptions about the direction and independence of causal relationships in the KB.

Observations

Every joint distribution assigns unique values to every PPT, and therefore for all sets of IPs.

A complete joint distribution determines the values of every PPT. For any distribution and any PPT, the value assigned to the PPT by the distribution can be obtained by algebraic manipulation of the elements of the joint distribution or (equivalently) evaluation of expectation of indicator functions over Boolean interpretations. Conditional parameters require two indicator functions, one for the conjunction of the antecedent and the consequent, and another for only the antecedent. For this reason, every joint distribution over Boolean variables also uniquely determines the values of every PPT and also therefore every element of every IP.

It is an open problem to obtain an expression for the number of distinct sets of IPs as a function of the total number of involved variables.

A complete IP over N Boolean variables must include exactly $2^N - 1$ PPTs. Although there are 2^N elements in the joint distribution, one is determined by the unitary sum constraint of the definition of probability distributions.

Hypothesis 1 *Any joint distribution over a given set of variables can be obtained by choosing suitable values for any complete set of IPs over those variables. (So, in a sense all IPs are isomorphic.)*

Proof. The joint distribution uniquely determines the values of the IPs, so choose those values.

Hypothesis 2 *The definitions of maximal and complete are equivalent.*

This is unproved.

Hypothesis 3 *Every complete IP has exactly* $2^N - 1$ *elements and every IP with* $2^N - 1$ *elements is complete.*

This is unproved.

Hypothesis 4 *The largest possible IP over* N *variables contains* $2^N - 1$ *terms.*

Hypothesis 5 *Any (open) distribution over Boolean variables can be exactly specified as a set of constraints on any complete independent parameterization.*

IPs over Two Variables

We examine the simple case of obtaining IPs of knowledge bases over two Boolean variables, A and B. First, we enumerate the set of probabilistic terms.

Marginals: $Pr(A)$, $Pr(B)$, $Pr(\neg A)$, $Pr(\neg B)$

Conjunctions: $Pr(A \wedge B)$, $Pr(A \wedge \neg B)$, $Pr(\neg A \wedge B)$, $Pr(\neg A \wedge \neg B)$

Disjunctions: $Pr(A \vee B)$, $Pr(A \vee \neg B)$, $Pr(\neg A \vee B)$, $Pr(\neg A \vee \neg B)$

Conditionals: $Pr(A|B)$, $Pr(A|\neg B)$, $Pr(B|A)$, $Pr(B|\neg A)$,
$Pr(\neg A|B)$, $Pr(\neg A|\neg B)$, $Pr(\neg B|A)$, $Pr(\neg B|\neg A)$

We can immediately prune the set of candidate independent parameters by observing complement symmetry. E.g., that constraining $Pr(A) = p$ is the same as constraining that $Pr(\neg A) = 1 - p'$, and that $Pr(x|y) = q$ exactly constrains $Pr(\neg x|y)$ to be $1 - q$.[4] Note that this is not the case when considering the antecedent of a conditional, as we show soon.

By exploiting complement relationships, we reduce our set of candidate independent parameters to:

Marginals: $Pr(A)$, $Pr(B)$

Conjunctions: $Pr(A \wedge B)$, $Pr(A \wedge \neg B)$, $Pr(\neg A \wedge B)$, $Pr(\neg A, \wedge B)$

Disjunctions: $Pr(A \vee B)$, $Pr(A \vee \neg B)$, $Pr(\neg A \vee B)$, $Pr(\neg A, \vee B)$

Conditionals: $Pr(A|B)$, $Pr(A|\neg B)$, $Pr(B|A)$, $Pr(B|\neg A)$

Of these remaining terms, there are 14 choose 3 which is 364 combinations. Substitution of complements in any IP involving these terms trivially yields yet another IP. We limit duplication of effort by growing the sets of IPs additively. Any single parameter is trivially independent. Therefore, we first seek pairs of independent parameters, and then to each of these consider adding a third term to obtain the full set of complete sets of independent parameters for any distribution over two Boolean variables.

We systematically enumerate parameters, by choosing pairs of categories. When combining a marginal with a non-marginal, we only consider $Pr(A)$. It's clear that symmetric arguments follow when instead considering $Pr(B)$.

1. Marginal, Marginal

[4] This does not hold for the case of belief functions. In that case, one equality constraint induces a corresponding inequality constraint due to the bound on the sum of masses.

The only pair of marginals, $Pr(A)$ and $Pr(B)$, is independent.

2. Marginal, Conjunction

   No pair is independent.

3. Marginal, Disjunction

   No pair is independent.

4. Marginal, Conditional

   All pairs are independent.

5. Conjunction, Conjunction

   No pair is independent.

6. Conjunction, Disjunction

   No pair is independent.

7. Conjunction, Conditional

   Exactly the pairs for which the antecedent is not conjoined are independent. They are:

   $Pr(A \wedge B)$, $Pr(A|\neg B)$
   $Pr(A \wedge B)$, $Pr(B|\neg A)$
   $Pr(A \wedge \neg B)$, $Pr(A|B)$
   $Pr(A \wedge \neg B)$, $Pr(B|\neg A)$
   $Pr(\neg A \wedge B)$, $Pr(A|\neg B)$
   $Pr(\neg A \wedge B)$, $Pr(B|A)$
   $Pr(\neg A \wedge \neg B)$, $Pr(A|B)$
   $Pr(\neg A \wedge \neg B)$, $Pr(B|A)$

8. Disjunction, Disjunction

   No pair is independent.

9. Disjunction, Conditional

   Exactly the pairs for which the antecedent of the conditional is not disjoined with its complement in the disjunction are independent. They are:

   $Pr(A \vee B)$, $Pr(A|B)$
   $Pr(A \vee B)$, $Pr(B|A)$

$Pr(A \vee \neg B), Pr(A|\neg B)$
$Pr(A \vee \neg B), Pr(B|A)$
$Pr(\neg A \vee B), Pr(A|B)$
$Pr(\neg A \vee B), Pr(B|\neg A)$
$Pr(\neg A \vee \neg B), Pr(A|\neg B)$
$Pr(\neg A \vee \neg B), Pr(B|\neg A)$

10. Conditional, Conditional

    All pairs are independent.

Proofs

Lemma 1 *No pair consisting of a marginal and a conjunction are independent.*

Proof:

$Pr(A)$ $Pr(A \wedge B)$ - not independent, because $Pr(A) \geq Pr(A \wedge B)$

$Pr(A)$ $Pr(A \wedge \neg B)$ - not independent, because $Pr(A) \geq Pr(A \wedge \neg B)$

$Pr(A)$ $Pr(\neg A \wedge B)$ - not independent, because $Pr(A) = 1 - Pr(\neg A) \leq 1 - Pr(\neg A \wedge B)$

$Pr(A)$ $Pr(\neg A \wedge \neg B)$ - not independent, because $Pr(A) = 1 - Pr(\neg A) \leq 1 - Pr(\neg A \wedge \neg B)$

Lemma 2 *No pair consisting of a marginal and a disjunction are independent.*

Proof:

$Pr(A)$ $Pr(A \vee B)$ - not independent, because $Pr(A) \leq Pr(A \vee B)$.

$Pr(A)$ $Pr(A \vee \neg B)$ - not independent, because $Pr(A) \leq Pr(A \vee \neg B)$

$Pr(A)$ $Pr(\neg A \vee B)$ - not independent, because $Pr(A) = 1 - Pr(\neg A) \geq 1 - Pr(\neg A \vee B)$

$Pr(A)$ $Pr(\neg A \vee \neg B)$ - not independent, because $Pr(A) = 1 - Pr(\neg A) \geq 1 - Pr(\neg A \vee \neg B)$

Lemma 3 *Every pair consisting of a marginal and a conditional is independent.*

Proof:

$Pr(A)$ $Pr(A|B)$ - independent, because as long as Pr(A) is not 1 or 0, then one may draw B such that an arbitrary proportion of it overlaps with A.

$Pr(A)$ $Pr(A|\neg B)$ - independent, because as long as Pr(A) is not 1 or 0, then one may draw B such that an arbitrary proportion of its complement overlaps with A.

$Pr(A)$ $Pr(B|A)$ - independent, because regardless of the total area of A, one may make draw B to overlap an arbitrary proportion of A, as long as Pr(A) is not 0.

$Pr(A)$ $Pr(B|\neg A)$ - independent, because regardless of the total area of A, one may make draw B to fill an arbitrary proportion of the complement of A, as long as Pr(A) is not 1.

Lemma 4 *No pair of conjunctions is independent.*

Proof: As the probability of any one conjunction approaches one, the probability of all other conjunctions approach zero due to the constraint that the probabilities of all conjunctions sum to one.

Lemma 5 *No pair consisting of a conjunction and a disjunction is independent.*[5]

Proof: Consider combining $Pr(A \wedge B)$ with the disjunctions.

$Pr(A \wedge B)$ $Pr(A \vee B)$ - not independent, because $Pr(A \wedge B) \leq Pr(A \vee B)$.

$Pr(A \wedge B)$ $Pr(A \vee \neg B)$ - not independent, because $Pr(A \wedge B) \leq Pr(A) \leq Pr(A \vee \neg B)$

$Pr(A \wedge B)$ $Pr(\neg A \vee B)$ - not independent, because $Pr(A \wedge B) \leq Pr(B) \leq Pr(\neg A \vee B)$

$Pr(A \wedge B)$ $Pr(\neg A \vee \neg B)$ - not independent, because $Pr(A \wedge B)$ = 1 - $Pr(\neg A \vee \neg B)$

By symmetry, all other pairs of a conjunction with a disjunction are dependent.

Lemma 6 *No pair of disjunctions are independent.*

Proof:

[5] Another proof method applicable to any set of terms involving disjucntions would be to use DeMorgan's laws, and reference the corresponding results for conjunctions.

$Pr(A \vee B)$ $Pr(A \vee \neg B)$ - not independent, because $Pr(A \vee B) + Pr(A \vee \neg B) \geq Pr(B) + Pr(\neg B) = 1$.

$Pr(A \vee B)$ $Pr(\neg A \vee B)$ - not independent, because $Pr(A \vee B) + Pr(\neg A \vee B) \geq Pr(A) + Pr(\neg A) = 1$.

$Pr(A \vee B)$ $Pr(\neg A \vee \neg B)$ - not independent, because $Pr(A \vee B) + Pr(\neg A \vee \neg B) \geq Pr(A) + Pr(\neg A) = 1$.

1. By symmetry, no other pairs of dijunctions is independent.

Lemma 7 *Exactly the pairs of a disjunction and a conditional for which the antecedent is not disjoined with its complement are independent.*

$Pr(A \vee B) = \neg(\neg(A \vee B)) = \neg(\neg A \wedge \neg B) = 1 - Pr(\neg A \wedge \neg B)$

$Pr(A \vee \neg B) = \neg(\neg(A \vee \neg B)) = \neg(\neg A \wedge B) = 1 - Pr(\neg A \wedge B)$

$Pr(\neg A \vee B) = \neg(\neg(\neg A \vee B)) = \neg(A \wedge \neg B) = 1 - Pr(A \wedge \neg B)$

$Pr(\neg A, \vee \neg B) = \neg(\neg(\neg A \vee \neg B)) = \neg(A \wedge B) = 1 - Pr(A \wedge B)$ Therefore the following independence relationships hold when combining a disjunction with a conditional,

$Pr(A \vee B)$, $Pr(A|B)$

$Pr(A \vee B)$, $Pr(B|A)$

$Pr(A \vee \neg B) Pr(A|\neg B)$

$Pr(A \vee \neg B) Pr(B|A)$

$Pr(\neg A \vee B) Pr(A|B)$

$Pr(\neg A \vee B) Pr(B|\neg A)$

$Pr(\neg A \vee \neg B) Pr(A|\neg B)$

$Pr(\neg A \vee \neg B) Pr(B|\neg A)$

Lemma 8 *All pairs of conditionals are independent.*

$Pr(A|B) Pr(A|\neg B)$ - independent, because we may freely change the proportion of B's which are A and the proportion of not B's which are A.

$Pr(A|B)Pr(B|A)$ - independent. Let $P(A|B)$ be fixed to any value in $(0, 1)$. Then if we wish $P(B|A)$ to be close to zero, we may make introduce arbitrarily many A's which are not in B. If we wish to make $P(B|A)$ close to one, we can do so by introducing no new additional As. Think of this as freely shrinking the marginal probability of either A or B.

$Pr(A|B)Pr(B|\neg A)$ - independent. Choosing A—B is equivalent to choosing A—B, therefore the independence decision for this case is equivalent to the case of $Pr(\neg A|B)$ and $Pr(B|\neg A)$, which is in turn equivalent by symmetry to $Pr(A|B)$ and $Pr(B|A)$, which w

$Pr(A|\neg B)Pr(B|A)$ - independent, by symmetry with the previous case.

$Pr(A|\neg B)Pr(B|\neg A)$ - independent, because of equivalence to the case of $Pr(\neg A|\neg B)$ and $Pr(\neg B|\neg A)$, which is symmetric to the case of $Pr(A|B)$ and $Pr(B|A)$, which are independent.

$Pr(B|A)Pr(B|\neg A)$ - independent by symmetry with the case $Pr(A|B)$ and $Pr(A|\neg B)$.

Lemma 9 *Exactly the pairs of a conjunction and a conditional for which the antecedent is not conjoined are independent.*

Proof:

$Pr(A \wedge B)$ $Pr(A|B)$ - not independent, because $Pr(A \wedge B) \leq Pr(A \wedge B)$ / $Pr(B)$ = $Pr(A|B)$

$Pr(A \wedge B)$ $Pr(A|\neg B)$ - independent, because we may set any proportion of the complement of B to also be included in A without affecting the value of $Pr(A \wedge B)$, assuming that $Pr(A \wedge B) < 1$.

$Pr(A \wedge B)$ $Pr(B|A)$ - not independent, because $Pr(A \wedge B) \leq Pr(A \wedge B)$ / $Pr(A)$ = $Pr(B|A)$

$Pr(A \wedge B)$ $Pr(B|\neg A)$ - independent, because we may set any proportion of the complement of A to also be included in B without affecting the relative size of $Pr(A \wedge B)$, assuming that $Pr(A \wedge B) < 1$. Consider the next conjunction paired with conditionals.

$Pr(A \wedge \neg B)$ $Pr(A|B)$ - independent

$Pr(A \wedge \neg B)$ $Pr(A|\neg B)$ - not independent, $Pr(A \wedge \neg B) \leq Pr(A \wedge \neg B)$ / $Pr(\neg B) = Pr(A|\neg B)$

$Pr(A \wedge \neg B)$ $Pr(B|A)$ - not independent, $Pr(A \wedge \not B) \leq Pr(A \wedge \neg B)/Pr(A) = Pr(\neg B|A) = 1 - Pr(B|A)$.

$Pr(A \wedge \neg B)$ $Pr(B|\neg A)$ - independent, because we may set any proportion of elements which are not included in A to be included in B without affecting the relative size of $Pr(A \wedge \neg B)$ It seems to be the case that we may assume a conjunction and a conditional, as long as the conjunction does not conjoin the conditional's antecedent. Therefore, to finish the pairs of conjunctions and conditionals, we assume that only the following are independent.

$Pr(\neg A \wedge B)$ $Pr(A|\neg B)$

$Pr(\neg A \wedge B)$ $Pr(B|A)$

$Pr(\neg A \wedge \neg B)$ $Pr(A|B)$

$Pr(\neg A \wedge \neg B)$ $Pr(B|A)$

B C-Rule Convergence Example

1. $(\forall\ x\ ((x\ Bird)\ \Rightarrow\ (:c\ (q\ x)\ (x\ Chicken)\ (-\ 1\ (q\ x))\ (x Other\text{-}bird))))$
2. $(\forall\ x\ ((x\ Chicken)\ \Rightarrow\ (:c\ (r\ x)\ (r\ Flying)\ (-\ 1\ (r\ x))\ (x Flightless))))$
3. $(\forall\ x\ ((x$ **other-bird**$)\ \Rightarrow\ (:c\ (s\ x)\ (xFlying)\ (-\ 1\ (s\ x)(xFlightless)))))$
4. $(Tweety\ chicken)$
5. $(:c\ q\ Chicken\ (1-q)\ Other\text{-}bird)$ [1,4 MP]
6. $(:c$
   $q\ (:c\ r\ Chicken \wedge Flying\ (1-r)\ Chicken \wedge Flightless)$
   $(1-q)\ Other\text{-}bird)$
7. $(:c$
   $q.r\ Chicken \wedge Flying$
   $q.(1-r)\ Chicken \wedge Flightless$
   $(1-q)\ Other\text{-}bird))$ [6, c-simplify]
8. $(:c$
   $q.r\ Chicken \wedge Flying\ \ q.(1-r)\ Chicken \wedge Flightless$
   $(1-q)\ (:c\ s\ Other\text{-}bird \wedge Flying\ (1-s)\ Other\text{-}bird \wedge Flightless))$ [7,3, chain-c]
9. $(:c$
   $q.r\ Chicken \wedge Flying\ q.(1-r)\ Chicken \wedge Flightless$
   $(1-q).s\ Other\text{-}bird \wedge Flying\ (1-q).(1-s)\ Other\text{-}bird \wedge Flightless)$ [8, c-simplify]

So that $Pr(Flying|Bird)$ is the sum over those disjoint possibilities (which is equivalent to using the a-rule combination rule due to the algebra, but maybe that's just a coincidence) which entail flying-bird, namely: $q.r + (1-q).s$. Suppose we have a new axiom (where I'm supposing that if it had an antecedent, that it's already satisfied)

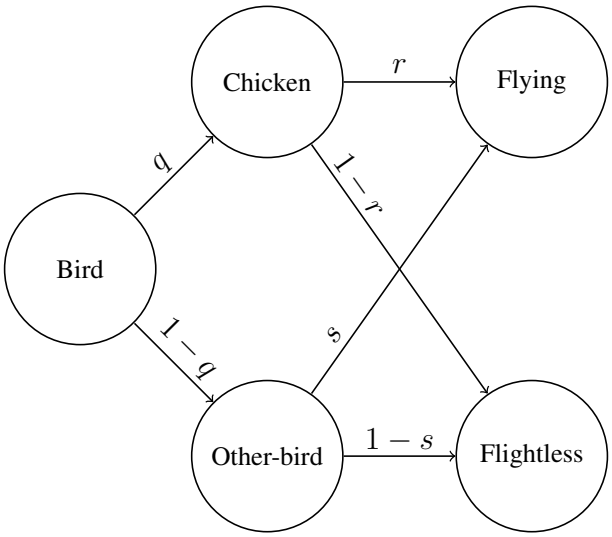


Figure B.1: An taxonomic spectrum example.

10. $(:c\ p\ Chicken\ (1-p)\ Other\text{-}bird)$

We could take exactly the same line of reasoning to yield:

11. $(:c$
    $p.r\ Chicken \wedge Flying\ p.(1-r)\ Chicken \wedge Flightless$
    $(1-p).s\ Other\text{-}bird \wedge Flying\ (1-p).(1-s)\ Other\text{-}bird \wedge Flightless)$

And then, because we have aligned spectra, we could combine 9 and 11:

12. $(:c$
    $(p.r).(q.r)/Z\ Chicken \wedge Flying$
    $(p.(1-r)).(q.(1-r))/Z\ Chicken \wedge Flightless$
    $((1-p).s).((1-q).s))/Z\ Other\text{-}bird \wedge Flying$
    $((1-p).(1-s)).((1-q).(1-s))/Z\ Other\text{-}bird \wedge Flightless))$

where $Z = p.q.r + p.q.(1-r) + (1-p).(1-q).s + (1-p).(1-q).(1-s) = p.q + (1-p).(1-q)$. Rewriting for clarity:

12. $(:c$
    $p.q.r/Z\ C \wedge F$
    $p.q.(1-r)/Z\ C \wedge \neg F$
    $(1-p).(1-q).s/Z\ \neg C \wedge F$
    $(1-p).(1-q).(1-s)/Z \neg C \wedge (q\ F)$

Note that $Z = p.q + (1-p).(1-q)$. Keep this point in mind, and consider, what if instead of chaining at the end, we had combined "greedily", meaning instead for step 11 we combined all our chicken evidence and then propagated that, i.e.,:

11* $(:c\ p.q/Z'\ Chicken\ (1-p).(1-q)/Z'\ Other\text{-}bird)$,
... where $Z' = p.q + (1-p).(1-q)$.

12* $(:c$
$(p.q/Z').r\ Chicken \wedge Flying$
$(p.q/Z').(1-r)\ Chicken \wedge Flightless$
$(1-p).(1-q)/Z'\ Other\text{-}bird)$ [11*, 2, c-chain, c-simplify]

13* $(:c$
$(p.q/Z').r\ Chicken \wedge Flying$
$(p.q/Z').(1-r)\ Chicken \wedge Flightless$
$((1-p).(1-q)/Z').s\ Other\text{-}bird \wedge Flying$
$((1-p).(1-q)/Z').(1-s)\ Other\text{-}bird \wedge Flightless))$

Rewriting for clarity:

13* $(:c$
$p.q.r/Z'\ C \wedge F$
$p.q.(1-r)/Z'\ C \wedge \neg F$
$(1-p).(1-q).s/Z'\ \neg C \wedge F$
$(1-p).(1-q).(1-s)/Z'\ \neg C \wedge \neg F)$

And since we already showed that $Z = Z'$, we can see that we get the same result regardless of path.