Defeasible Bayesian Reasoning

A Review of Defeasible Reasoning and Bayesian Networks

Luke Neville
Computer Science Honours
University of Cape Town
Cape Town, South Africa
nvlluk001@myuct.ac.za

ABSTRACT

This literature review seeks to examine the basics and fundamentals of Bayesian Networks and Propositional Logic with defeasible reasoning. This is done with the specific intention of informing a project which aims to introduce a classical and defeasible knowledge base into a Bayesian Network. This is done with the objective of reducing the complexity of the Bayesian Network by allowing for additional relationship dependencies between variables in the network. This review focuses on introducing these two topics independently, with specific insight into the implementation of algorithms used in reasoning.

We start by discussing Bayesian Networks and the success found in modelling probabilistic relationships between groups of variables. This is in part due to the assumption of independence between variables not directly connected in the network, allowing for efficient calculations of probability distributions. We then take a look at the basics of propositional logic and its use in reasoning about common place scenarios, before introducing defeasibility as a means reasoning about exceptional situations. We then briefly examine how propositional logic statements may be added to a Bayesian Network to produce additional dependencies between variables.

Keywords

Bayesian Networks; Formal Logic; Knowledge Representation; Propositional Logic; Defeasible Reasoning

1. INTRODUCTION

Over the past couple of decades Bayesian Networks have become an extensively used tool with which to quickly and efficiently reason about probabilistic information. Bayesian Networks have been researched in many fields, most notably in Artificial Intelligence in order to draw inference about common sense scenarios [1]. The power of Bayesian Networks lies in the way probabilities are computed. Due to the existence of efficient algorithms the probability of a given set of network variables can be computed without needing to compute the probability distribution of the entire network. This is due to the key feature of Bayesian Networks which is that a variable becomes independent of its non-descendents once its parent variables are known [2].

For centuries, mathematicians have been seeking formal structures in which to represent common logic in a mathematical environment. Propositional, or Classical, Logic is one such example. More recently the field has seen interest in Computer Science for its applications to automating inference techniques in knowledge based systems [3]. These implementations of logical reasoners have been successful in automating inference from knowledge bases, and form the basis of many artificial intelligence methods. Propositional logic however, does not allow for any exceptionality to exist within a knowledge base. Due to this, the scope with which it can be used to reason is limited. Defeasible reasoning extends classic logic by introducing well defined and systematic approaches to exceptions to general rules. This allows a knowledge base to contain sentences such as "*x typically* does y". The introduction of the notion of typicality deals with knowledge bases where a set of given rules compete or are in conflict with one another. This branch of reasoning has, as in with Bayesian networks, been used in Artificial Intelligence research in order to analyse common sense reasoning [4].

There has been extensive research into the construction and application of Bayesian networks as well as defeasible reasoning however, there has been limited study into the union of these two fields. This project aims to introduce basic defeasible reasoning into a Bayesian Network and to evaluate how this effects reasoning under the knowledge base. The defeasible reasoning introduced to the Bayesian Network will include limited forms of dependence between variables in the form of varying levels of implication. The project also involves a practical component, in which a reasoner is implemented which can reason about these extended Bayesian Networks.

This literature review aims to investigate the intersection of Bayesian Networks and defeasible reasoning. The review begins by investigating the current state of research into Bayesian Networks, focusing initially on the theoretical background of how probabilistic information is reasoned about, and then on how Bayesian Networks are implemented. Defeasible reasoning is also explored beginning with an introduction to reasoning through classical logic and the notation used, followed by an explanation of how exceptionality is introduced and reasoned with. We then take a look at how these two fields converge, with a specific focus on implementing a practical Defeasible Bayesian Reasoner.

2. BAYESIAN NETWORKS FOR PROBABILISTIC REASONING

Bayesian Networks, also known as Belief Networks [15], are a class of probabilistic graphical models that have been used in many fields for use as causal modelling and probabilistic inference [14]. They allow for a coherent method

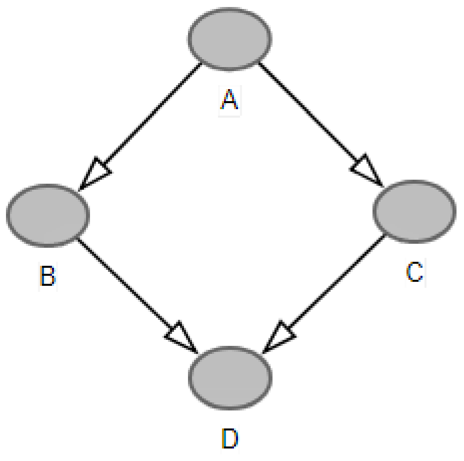

Figure 1: The DAG for a Simple Bayesian Network

of structuring probabilistic information about a system in a way that makes it is easy to logically reason about the system and it's potential states [2]. Bayesian Networks also include algorithms for coming to conclusions about the system.

A Bayesian Network is a double $BN = DAG, CPT$ where DAG is a Directed acyclic graph and CPT is a set of conditional probability tables [15]. A DAG is a directed graph (each edge has a direction associated with it) that contains no cycles - no path can be drawn from a node back to the same node by following the directed edges. In the DAG, the nodes correspond to the random variables that are being modelled, while the edges are interpreted as the probabilistic independence relationship between these variables [2] [15]. The absence of an edge between two variables denotes the independence between the variables [16].

Each variable in the Bayesian Network requires a conditional probability table which specifies the probabilistic relationship between the given variable and its parents in the DAG. Bayesian Networks are powerful due to the key notion that a variable is independent of its children once the values of its parents are know [16]. Another main feature of Bayesian Networks is guaranteed completeness and consistency. This is due to the fact that there can only ever be one probability distribution that completely satisfies the network [2]. To see this, we look at Bayesian statistics.

As described in Bayesian Statistics, The R Book [17] a probability distribution is a mathematical function that describes all the possible values that a random set of variables can take in a given system. For example, if the system contains four variables the probability function would be $P(A, B, C, D)$, where this is the probability of each variable taking a specific value. In a system where no independence is specified, the probability distribution must contain the all variables taken in relation to all over variables. If the variables in our example system were binary this would result in 2^n probability distributions where $n = 4$ in the example above - the number of variables in the system. As previously described, Bayesian Networks restrict this by introducing independence between certain variables.

A	*P(A)*
True	p_1
False	p_2

A	*B*	*P(B | A)*
True	True	p_3
True	False	p_4
False	True	p_5
Flase	False	p_6

A	*C*	*P(C | A)*
True	True	p_7
True	False	p_8
False	True	p_9
Flase	False	p_{10}

B	*C*	*D*	*P(D | B,C)*
True	True	True	p_{11}
True	True	False	p_{12}
True	False	True	p_{13}
True	False	False	p_{14}
False	True	True	p_{15}
False	True	False	p_{16}
False	False	True	p_{17}
False	False	False	p_{18}

Table 1: The CPT Tables for the Simple Bayesian Network

Bayes Rule is extremely powerful and forms the basis for the computations that are used to propagate probabilities through a Bayesian Network when new information is received. Bayes Rule allows us to ask, *"given this new information, how is our view of the world changed?"* [17]. For any two variables **A** and **B**, Bayes Rule states

$$P(A|B) = \frac{P(B|A) \times P(B)}{P(A)} \tag{1}$$

where $P(A|B)$ is read as *the probability of A given that B has taken place*. Bayes rule can be extended to multiple variables, for example $P(A|B, C, D)$ [16]. These extended Bayes are used to compute the values of a given variable in a Bayesian Network, where B, C, D would be the parent nodes of node A. This is further discussed in the following sections.

Due to these statistical properties and the independence between variables and their children, the computation required to compute the probability distribution of a Bayesian Network is greatly reduced. This gives Bayesian Networks their power - the probability of a given variable can be computed extremely efficiently [2]. For example, consider a system with n variables. To compute the joint Probability Distribution of this system, one would have to compute all possibilities of each combination of variables. This would require $2^n - 1$ probability values. When represented as a Bayesian Network however, this number is reduced as the probability value for each variable is only conditioned on its parents' probability value.

In summary, Bayesian Networks work as well as they do at computing probability distributions due to the assumption that variables that are not directly connected in the network are independent - that is, their probabilities are not directly dependent on one another. Bayes theorem is central to Bayesian Networks - the construction and propagation of probabilities relies on it. Bayes theorem could, however be applied to variables that are not directly connected in the network. This would still produce probability distributions that could be reasoned with, but the algorithms that make Bayesian Networks so fast and efficient could not be applied.

2.1 Reasoning with a Bayesian Network

The following section discusses how Bayesian Network is used to come to conclusions about variables. For a more in depth look at this topic see Modelling and Reasoning with Bayesian Networks (Darwiche, A) [24], from where these algorithms are taken. We start with discussion on the types of queries that can be made on a Bayesian Network, followed with brief explanation of some algorithms that can provide answers to these queries.

2.1.1 Types of Queries

The simplest and most intuitive query to ask is what is the Probability of some variable, e, in the network, i.e. $P(e)$. This may extend to asking the Probability of multiple variables, i.e. $P(e, q)$. The variables in question (E, Q) are known as Evidence variables, and the query $P(e, q)$ is known as the probability-of-evidence query.

A marginal distribution is a projection of the joint probability distribution on a subset of the network variables. In other words, it is the sum of the probabilities of the subset of network variables. When the marginal is computed with no evidence, this is known as the prior marginal. This is opposed the posterior marginal, which is computed with some evidence.

The Most Probable Explanation (MPE) is the most probable instantiation of the network given some evidence. If $x_1..x_n$ are the network variables and e is some prior known evidence, the MPE is the values of $x_1..x_n$ where $P(x_1..x_n|e)$ is at a Maximum. This instantiation is called the most probable Explanation given evidence e.

The Maximum a posteriori query is a more general case of the MPE discussed above. Given a network with variables X, and M is a subset of those variables, the query seeks to ask what are the values m of M where given some evidence e, $P(m|e)$ is maximal. This is the same as the most probable Explanation except where the network variables in question are some subset of the entire network variables.

2.1.2 Algorithms for Answering Queries

There are a variety of methods of answering these queries on a network. Each method has it's strength at answering a specific type of query, and may range in time and algorithmic complexity.

One of the simplest methods of inference in Bayesian Networks is that of Variable Elimination. This method is restricted to answering probability of evidence and prior and posterior marginal queries. Variable Elimination is the process of successively removing variables from a Bayesian network, while still maintaining the ability to answer queries on the remaining variables, which in this case would be the variables of interest.

Consider a Bayesian Network with 5 variables, N = A,B,C, D,E. Variable Elimination will be able to compute a marginal $P(D, E)$ by eliminating the variables A, B and C from the network. This process of summing out variables is based off reducing the variables in the join probability distribution. Given the network N, there will be 2^5 possible worlds, though some of them may not exist due to properties of a Bayesian network as previously discussed. In some of these worlds, all variables in the network will hold the same truth value bar one variable. For example, there are two worlds where B, C, D and E = true - one where A = true, and one where A = false. In order to sum out the variable A, for instance, the probabilities of these worlds occurring are added, and the variable A is simply dropped from the table. This process can be repeated for all variables in the the table not under question, until the probability distribution only contains the variables that form part of the query. This can be continued until there is only one variable left. If this is done, the prior marginal can be computed for each individual variable. If some evidence is given, the appropriate worlds are simply removed from the probability distribution and the marginals are recalculated.

Factor Elimination improves on the complexity of variable elimination when answering multiple queries. Considering a Bayesian network, the marginals for each variable can be computed using variable elimination, in $O(n^2exp(w))$ time where n is the number of variable and w is the width of the network. This will clearly become an issue when dealing the large networks. Using an algorithm known as the Jointree Algorithm, this complexity is reduced to $O(nexp(w))$ time. This algorithm is best described as Factor Elimination, where factors are eliminated instead of variables.

The algorithm will compute the marginal of some variable Q in a network by eliminating all factors except for the one that contains the variable Q. A factor is a grouping of variables, $P(X, Y, W)$, as found in a CPT. The elimination of a factor f from a set of factors S firstly comprises of eliminating all variables V that appear only in the factor f, and then multiplying the result of the sum of those variables by some other factor is the set.

Conditioning, also known as case analysis, is a form of inference in which the probability of a variable is computed by considering all cases which correspond to a particular assumption. Then, each case is solved under its corresponding assumption, and summed together to obtain the probability of the variable. This is illustrated as

$$P(x) = \sum_c P(x, c) \tag{2}$$

Here, the probability of x is computed by considering a number of cases c, and summing the probability of x with each case c. There are two Conditioning algorithms, Cutset and Recursive Condition, both of which are discussed in depth in Darwiche, A. [24].

Approximate inference is a method of appropriating the results of a query, allowing for trade-offs between quality of approximation and computational resources. Belief Propagation is one such inference method. In this method, messages are passed from node to node, effected by the probability and relationship between the nodes. The messages are passed inwards, towards the node that is being queried. This method of message passing is based off of the jointree algorithm used in factor elimination Another method of appropriate inference is Stochastic Sampling. In Stochastic Sampling, situations are repeatedly simulated according to their probability and then the probability of events is estimated due to their occurrence in the simulated situations. For example, consider an event or variable “is rainy”. If we simulate 100 situations and find out that is is rainy in 25 of them, we say that the probability of being rainy is 1/4.

2.2 Applications and Implementation of Bayesian Networks

Bayesian Networks have found success in a wide variety of fields. In recent years, applications in Machine Learning and Artificial Intelligence have been fruitful, especially in fields such as speech recognition [2] [18] and document and text classification [19]. There have also been widespread uses in biology, especially bioinformatics and gene expression analysis [20].

Due to their widespread use, there are many software and computer language packages that can efficiently learn and evaluate Bayesian Networks. One such example is BNFinder [21], a python package that can construct Bayesian Networks from experimental data, aimed for use in biological fields. BNFinder is a free tool available to researchers with the specific aim of being simple and efficient to use. Since BNFinders release, a second version has been released, BNFinder2 [22], which is faster and contains a number of other improvements. BNFinder2 is freely available under the GNU public licence.
JAGS (Just Another Gibbs Sampler) is an open source program for modelling and simulating Bayesian models [23]. It is licensed under the GNU general public licence and is written in C++.

3. FORMAL LOGIC FOR REASONING

Logic, as a philosophical and mathematical tool, is hard to define. It provides a basis for evaluating and reasoning about the world around us. Mathematically, it is necessary for this powerful tool to be formalised into well defined languages which can be further used for knowledge representation. This allows us to draw , where a knowledge base is some collection of statements about a given world.

This mathematical need for knowledge representation has led to a wide variety of formal logic representations. These representations vary from languages with few operators and thus limited expressivity to languages many operators and high levels of expressivity. These languages with high expressive power may be more useful in modelling the world accurately, however this comes at a cost of high algorithmic complexity [5].

Today, the most common formal mathematical logic systems are propositional and first-order logic due to their applicability to modelling philosophical theory and nature [6]. We will focus on propositional logic and its relevance to drawing inference about a given knowledge base.

3.1 Classical Reasoning

Propositional Logic, also known propositional calculus, is a branch of logic that deals with singular truth-bearing variables [7] and how these variables can be connected to create logical arguments. These truth-bearing variables are known as 'atoms' and are the smallest unit of knowledge in a propositional knowledge base [5]. They can only take on one of two states - true or false. Logical connectives can be used to extend an atom or join multiple atoms together in order to create composite atoms. These composite atoms can in turn only be true or false, though they are composed of one to many true or false atoms. Logical connectives can be seen as operators as they perform some function on the atoms. Atoms and composite atoms are known as sentences, and a collection of sentences makes up a knowledge base [5][7]. Something that is important to note, is that all sentences in a knowledge base can either hold a value of true or false - these are known as a declarative statement or sentence. This implies that a sentence can never be both true and false, or neither of the two.

As sentences are precise in their truth value, we can use them to reason about the world, or the state of a given system. In natural language, we use declarative sentences all the time to describe what is going on around us. For example, we can state that *"it is raining"*. This is a propositional atom - it holds a singular value of either true or false and tells us something about the current state of the world.
In classical logic, upper case letters (**P**,**Q**,**R**,**S**...) are used to represent atoms [5]. As such, our sentence *"it is raining"* can be labelled as **P**. Another sentence, *"the ground is wet"* may be labelled as **Q**.

3.1.1 Logical connectives

Propositional logic is limited to five logical connectives. This is enough to ensure an expressive language, while still maintaining simplicity in reasoning. Many of these connectives find their roots in natural language, namely *and*, *or*, *not* and *if P then Q*. These logical connectives are discussed below. For a more in-depth explanation of each operator, see *A Concise Introduction to Logic (DeLancey, C)* [5]

Sentences can be constructed by applying these logical connectives to atoms. The *not* operator is applied to one sentence, whilst the remaining connectives are applied on pairs of sentences. The *not* operator, described as negation, is represented by the symbol ¬. This inverts the value associated with the atom to which it is applied. In natural language, we can read negation as simply adding *"it is not the case that..."* to the beginning of our sentences. For example, if our sentence **P** is *"it is sunny"*, ¬**P** can be interpreted as *"it is not the case that is is sunny"*, or simply, *"it is not sunny"*.

Truth tables are a useful method of depicting the values of logical expressions for each combination of the input variables [8]. Thus, we can use truth tables to show the logical validity of propositional statements. Below, the truth table for negation shows the possible inputs for **P** and the resulting values for ¬**P**.

P	$\neg P$
1	0
0	1

Table 2: Truth table for ¬P

The *and* connective, known as conjunction, uses the symbol ∧. This is applied to two sentences, resulting in one compound sentence. This compound sentence takes the value true if both conjuncts (that is, both individual sentences) are true; otherwise the compound sentence is false. This follows from our natural language usage of the word *"and"*. From out example above of rain and went ground, **P** ∧ **Q** is *"it is raining and the ground is wet"*. This would be true only if it was raining and the ground is wet, otherwise the sentence would be false. It is important to note that ordering of the conjuncts does not matter with conjunction - **P** ∧ **Q** is equivalent to **Q** ∧ **P**.

Disjunction - the *or* connective - works in much a similar way to conjunction. The symbol ∨ is used to denote dis-

P	Q	$P \wedge Q$
1	1	1
1	0	0
0	1	0
0	0	0

Table 3: Truth table for P $\wedge$ Q

junction. Or requires two conjuncts and for the compound sentence to be true, only one of the two sentences is required to be true. Using our example from above P $\vee$ Q would be interpreted as *"it is raining or the ground is wet"*. As with conjunction, ordering to this sentence is not important.

P	Q	$P \vee Q$
1	1	1
1	0	1
0	1	1
0	0	0

Table 4: Truth table for P $\vee$ Q

It is useful to come to conclusions about some information, based off of the results of some other, prior fact. In propositional logic, this is known as a conditional sentence, and is used in the same way we use *"if" (some bit of information) "then" (some other bit of information)* in natural language. The symbol used to represent this *if ... then ...* relationship is $\rightarrow$. For example, $\mathbf{P} \rightarrow \mathbf{Q}$ would be read as *"if it is raining then the ground is wet"*. The first sentence, **P**, is known as the antecedent, and the second, **Q**, is known as the consequent. Note that unlike conjunction and disjunction the order of the sentence matters - $\mathbf{Q} \rightarrow \mathbf{P}$ holds an entirely different meaning.

Conclusions are drawn from conditional sentences by first analysing the antecedent. If the antecedent is true, then the truth value for the sentence simply takes on the value of the consequent. This follows from how we would interpret these scenarios in natural language. If the antecedent is true, then the statement's truth relies on the outcome of the consequent. If the antecedent is false however, then the value for the sentence is always true. This is due the fact that no real conclusion can be drawn from the statement if if the antecedent is false, and thus we default to true.

P	Q	$P \rightarrow Q$
1	1	1
1	0	0
0	1	1
0	0	1

Table 5: Truth table for P $\rightarrow$ Q

The final connective used in propositional logic is the biconditional. Represented as $\leftrightarrow$, the biconditional represents an *if and only if* relationship between two sentences. In order to understand how this connective works, we analyse the meaning of the operator. $\mathbf{P} \leftrightarrow \mathbf{Q}$ can be interpreted as **P** if **Q** and **P** only if **Q**, or more simply, **P** if **Q** and **Q** if **P** or $\mathbf{P} \rightarrow \mathbf{Q} \wedge \mathbf{Q} \rightarrow \mathbf{P}$. Thus, the value for the sentence is true only if both sentences hold the same value.

P	Q	$P \leftrightarrow Q$
1	1	1
1	0	0
0	1	0
0	0	1

Table 6: Truth table for P $\leftrightarrow$ Q

3.1.2 Reasoning with Propositional Logic

Since propositional logic is decidable, that is given any sentence and the truth values for each propositional atom in the sentence, the truth value for the whole sentence can always be computed. This means that for any combination of truth values assigned to each atom in a knowledge base, every sentence can be computed. This is known as a model - a particular assignment of true or false to each atom in the knowledge base. There will always be 2^n models where n is the number of atoms in the knowledge base.

The evaluation of a sentence is the overall value resulting from applying a particular model to that sentence. This can be extended to the evaluation of an entire knowledge base, which is the conjunction of all sentences in the knowledge base evaluated under given model. Due to this, there are 2^n evaluations of any given model.

A query can be made on a knowledge base by checking if the knowledge base allows for the new, query sentence to evaluate to true under any given model. This is known as entailment, and is represented as $\mathbf{KB} \models \mathbf{S}$, where **KB** is the knowledge base and **S** is the query sentence. A query **S** is said to be entailed by a knowledge base **KB** if all models that result in **KB** evaluating to true, also result in **S** evaluating to true [9]. This process of evaluation can be used to query knowledge bases. However, this process is computationally complex due to exponential increase in models with the increase in size of a knowledge base.

3.2 Defeasibility as a means of reasoning about exceptionality

Defeasible reasoning is an extension of propositional logic that deals with reasoning when the knowledge base contains rules that conflict or compete with each other [10]. It is common for a reasonable knowledge base that models the real world to contain conflicting statements. For example, natural moral thought would dictate that one should not steal, but this may clearly be over-ruled by other moral philosophies, such as *"feed your children"* [10]. Propositional logic does not allow for this kind of reasoning, as a sentence in the knowledge base is taken to be fully correct - thus one would be conflicted as to whether one should steal food in order to feed their children.

Defeasible reasoning allows for the concept of exceptionally to exist within a formal knowledge base [11]. This means that relationships between atoms may be described as 'typical', allowing for two statements to conflict and be reasoned by saying that one of the relationships is simply an exception to the norm. Kraus, Lehman & Magido in their paper *Nonmonotonic Reasoning, Preferential Models and Cumulative Logics* [27] introduced the idea of defeasibility in order to describe these non-definitive, more contextual relationships. They used the symbol to denote defeasible entailment. Where in classical logic, the sentence $\mathbf{P} \rightarrow \mathbf{Q}$ is read as *if **P** then **Q***, the defeasible statement P $\mid\!\sim$ Q is read as *if **P** then (typically, normally, apparently) **Q*** [12] [13]. This allows for the statement *"if your children are hungry, typically don't steal in order to feed them"* to exist within a defeasible knowledge base. As we can see, the notion of defeasibility allows for propositional logic to become more expressive in the way that the world can be described. This results in a richer knowledge base and stronger reasoning power.

3.3 Non-Monotonicity of Defeasible Reasoning

Defeasible reasoning allows us to withdraw conclusions that we may have come to from a knowledge base. As new information is added to a knowledge base, we may decide that a deduction we would have made is no longer reasonable, and should be removed from the knowledge base. This is due to defeasible reasoning being non-monotonic. This describes how adding sentences to a knowledge base may decrease the conclusions that can be drawn from that knowledge base [13]. A sentence maybe be added that clears up any uncertainty that a defeasible statement introduced to the system, allowing for a definite conclusion to be drawn [25]. In contrast, propositional logic is monotonic - adding a sentence to the knowledge base can only result in the number of conclusions staying the same size or growing.

3.3.1 Reasoning with a Defeasible Knowledge Base

This notion of defeasibility is only useful if there is an accompanying method of being able to come to logical conclusions about the knowledge base when given a query. Rational Closure is an algorithm that allows entailment of a given query on a knowledge base to be computed.

As described in Casini & Straccia [13], the rational closure algorithm involves ranking each sentence in the knowledge base **K**. The ranking is based on the exceptionality of the sentence - a higher rank indicates that the sentence is deemed to be more exceptional in the context of **K**. This set of ranked propositional statements is known as the *ranked interpretation* **R**. This ranked interpretation can then be queried for entailment as described in section 3.1.2.

3.3.2 Implementing Defeasibility

The rational closure algorithm relies on heavy use of classical propositional reasoning. Due to this, a defeasible reasoner can be implemented as wrapper around a classical reasoner, making a sequence of calls to the classical reasoner in order to reason about a knowledge base. There are a number of reasoners that implement rational closure and thus allow for defeasible statements to be reasoned with. Casini *et al* in their paper *Introducing Defeasibility into OWL Ontologies* [28] describe their approach to modifying a real-world Web Ontology Language (OWL) in order to introduce defeasible statements into the language. They found that implementing defeasible reasoning is feasible, based on generated test data. Despite some limitations, the methods described in the paper pave the way for more sophisticated implementations of defeasibility into real-world ontologies.

4. LOGICAL REASONING IN BAYESIAN NETWORKS

This project aims to combine the implication statements from classical and defeasible reasoning, within a Bayesian Network. A Bayesian Network could be read as saying that if A happens, this implies that B happens, with some prior probability. This is read in a very similar manner to logical implication, A -> B, just with the added probability. Due to this the variables in a Bayesian Network can be viewed as propositional atoms. Given some Bayesian Network, we will try to introduce the logical statements A -> B and A | B into the knowledge base, where A and B are variables in the Bayesian Network. This will produce additional dependencies between variables in the Bayesian Network, or modify already existing dependencies. In this project we will only introduce limited forms of dependence between variables in the network. These dependencies will take the form of classical and defeasible implication. These logical implication sentences will be added after the network has been constructed. We will attempt to answer such questions has "How does this change the network and the algorithms associated with it?", and "How do we reason with this network?".

Initially, we will only attempt to introduce material/ classical implication, (A -> B). Once this is successful, we will introduce defeasible implication (A | B). This will make up the theory part of the project, which will look at how the Bayesian Network algorithms need to be modified in order to accommodate these dependencies. The following two sections will discuss each form of implication.

4.1 Classical Implication

Classical implication, discussed in section 3.1, says that if A happens, then B happens. In a Bayesian Network, this could mean that P(B | A) = 1, given that B is a child of A. This relaxes the requirement of independence of variables. This has the potential of collapsing a network, by merging the two nodes A and B.

4.2 Defeasible Implication

Defeasible implication, discussed in section 3.2, is slightly more complicated. There is no simple explanation as to how the typicality of statements is introduced into Bayesian Networks. This refers to the ranking of statements used in reasoning with defeasible reasoning. How is the quantifiable probability in Bayesian Networks related to the qualitative relationship of typicality. We will attempt to formalise this relationship and work out what this means for Bayesian Networks and the algorithms that are used in reasoning.

4.3 Implementing Implication in Bayesian Networks

My part of the project is specifically looking at implementing the classical and defeasible implication as background knowledge in a Bayesian Network in the algorithms described in section 2.1. This will require a firm understanding of how the algorithms are implemented currently, and also what changes need to be made to describe classical and defeasible implication. This implementation would be done in one of two ways. The first option is to use a standard Bayesian Network algorithm with wrapper around it in order to solve the modified problem. If this does not prove possible, a modified algorithm will be implemented from scratch. This decision will be made as the theory section of the project become clearer.

There are many algorithms for reasoning with a Bayesian

Network. All have advantages and can answer different queries. Not all of these algorithms are amenable to relaxation of independence by introduction of logic. I will only attempt to implement those that lend themselves to logical implication. Work will need to be done managing space and resource requirements of the algorithms, especially with additional computation of the logical statements.

Consideration may need to made of existing implementations of Bayesian Network algorithms, such as BNFinder2 [22] and JAGS [23]. We will need to consider whether it is more favourable to modify these already existing Open Source programs to allow for logic, or to implement the complete algorithm from scratch. This second option may turn out to be simpler, as modifying large projects is often not a trivial task.

It must be noted that we are not attempting to implement Bayesian Network learning algorithms. Instead, are looking to answer the question of how to reasoning and answer queries on a Bayesian Network that is already constructed and has had logical implication added to the knowledge base. This means that our implementations will not involve and learning Algorithms, but will assume a Bayesian Network has already been created either via manual construction or automatic learning, and will take this Bayesian Network in as an input.

4.4 Discussion

In natural language and thought we reason in a very similar way to defeasible reasoning in a day to day context. As this is how we commonly construct our augments, it is incredibly useful to be able to add these sorts of arguments to Bayesian Reasoning. This is growing in importance as Bayesian Networks are increasingly used in artificial intelligence. Thus, the ability to subtly modify Bayesian Networks in a logical way has potentially useful gains.

5. RELATED WORK

It would be remiss to not briefly mention probabilistic forms of propositional logic. Combining logic and probability is fundamentally what we intend on achieving in this project, however our method of introducing logic into a Bayesian Network is not the only way of achieving this. Probabilistic Theorem Proving allows for reasoning under uncertainty proving theorems in first-order logic and testing for satisfiability in propositional logic [29]. This allows for exact probabilities to be attatched to propositional sentences.

Probability Logic is another form of knowledge representation which seeks to combine probability theory with deductive logic in order to make rational, formal arguments [30].

6. CONCLUSIONS

Bayesian Networks have been extensively researched, with many practical uses and methods for constructing and evaluating them finding their way into literature [2]. The key insight into their value and performance is due to the assumption of independence between variables that are not directly connected in the network. This greatly simplifies the process of evaluating probabilistic relationships and makes this sort of reasoning feasible.

There are many knowledge representation languages, from first order to propositional logic. We are only interested in limited forms of implication in propositional logic and defeasible reasoning. That is, how can the classical implication $\mathbf{P} \rightarrow \mathbf{Q}$ and defeasible $\mathbf{P} \mathrel{|\!\sim} \mathbf{Q}$ be introduced into another knowledge representation system.

There is not a lot of literature on introducing logic into Bayesian Networks. Cozman, F., and Mauá, D. [26] describe representing Bayesian Networks in a variety of description logics, especially exploring the complexity of inference in these languages. However, there appears to be little in the way of defeasible implication in Bayesian Networks. Due to this gap in knowledge, it provides a good opportunity to explore this topic and the relationship between the two fields.

Briefly, this will involve viewing variables in a Bayesian network as propositional atoms. This allows for logical statements such as implication to be added to a Bayesian network knowledge base, increasing the number of dependencies between variables.

Implementation of a defeasible Bayesian reasoner will take one of two forms. Either an existing Bayesian Network reasoner will be used by implementing a wrapper around it which adds the logical implication statements into the knowledge base, or the algorithm will be implemented from scratch. If the former option is taken, there are many reasoners that can be used by applying a wrapper to them [22] [23].

7. REFERENCES

[1] Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.Morgan Kaufmann, 1988

[2] Darwiche, A. (2010). Bayesian networks. Communications of the ACM, 53(12), 80–90. doi:10.1145/1859204.1859227

[3] Hunter, A. (2002, July 1). Propositional Logic: Deduction and Algorithms (Book Review). Studia Logica: An International Journal for Symbolic Logic. Kluwer Academic Publishers.

[4] van der Hoek, W. (2000). Review: Donald Nute, Defeasible Deontic Logic. Bull. Symbolic Logic, 6(1), 89–94.

[5] Delancey, C. (2017). A concise introduction to logic. Geneseo, NY: Published by Open SUNY Textbooks, Milne Library, State University of New York at Geneseo. Retrieved from https://open.umn.edu/opentextbooks/BookDetail.aspx?bookId=452

[6] Ferreirós, José (2001), "The Road to Modern Logic-An Interpretation", Bulletin of Symbolic Logic, 7 (4): 441–484, doi:10.2307/2687794, JSTOR 2687794.

[7] Bealer, G. (1998). Propositions. (philosophy of language, traditional proposition theory). Mind, 107(425), 1. doi:10.1093/mind/107.425.1

[8] Enderton, H. (2001). A Mathematical Introduction to Logic, second edition, New York: Harcourt Academic Press. ISBN 0-12-238452-0

[9] Eiter, T., & Gottlob, G. (2006). Reasoning under minimal upper bounds in propositional logic. Theoretical Computer Science, 369(1), 82–115. doi:10.1016/j.tcs.2006.07.054

[10] van der Hoek, W. (2000). Review: Donald Nute, Defeasible Deontic Logic. Bull. Symbolic Logic, 6(1), 89–94.

[11] Walton, D. (2011). Reasoning about knowledge using defeasible logic. Argument Computation, 2(2-3), 131–155. doi:10.1080/19462166.2011.637641

[12] Maier, F., & Nute, D. (2010). Well-founded semantics for defeasible logic. Synthese, 176(2), 243–274. doi:10.1007/s11229-009-9492-1

[13] Casini, G., Straccia, U. (2010). Rational Closure for Defeasible Description Logics. LNAI 6341, 77–90.

[14] Hadlock, C. (2005, September 1). Causality: Models, Reasoning, and Inference. Journal of the American Statistical Association. Taylor Francis. doi:10.1198/jasa.2005.s38

[15] Zhao, Y., Chen, Y., Tu, K., Tian, J. (2017). Learning Bayesian network structures under incremental construction curricula. Neurocomputing, 258, 30–40. doi:10.1016/j.neucom.2017.01.092

[16] Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D. (2013). Context-Specific Independence in Bayesian Networks.

[17] Crawley, M. (2012). Bayesian Statistics. The R Book (pp. 752–767). Chichester, UK: John Wiley Sons, Ltd. doi:10.1002/9781118448908.ch22

[18] Mitra, V., Nam, H., Espy-Wilson, C., Saltzman, E., Goldstein, L. (2011). Robust speech recognition with articulatory features using dynamic Bayesian networks. The Journal of the Acoustical Society of America, 130(4), 2408–2408. doi:10.1121/1.3654653

[19] Denoyer, L., Gallinari, P. (2004). Bayesian network model for semi-structured document classification. Information Processing and Management, 40(5), 807–827. doi:10.1016/j.ipm.2004.04.009

[20] Friedman, N.; Linial, M.; Nachman, I.; Pe'er, D. (2000). "Using Bayesian Networks to Analyze Expression Data". Journal of Computational Biology. 7 (3–4): 601–620. doi:10.1089/106652700750050961. PMID 11108481.

[21] Wilczyński, B., Dojer, N. (2009). BNFinder: exact and efficient method for learning Bayesian networks. Bioinformatics, 25(2), 286–287. doi:10.1093/bioinformatics/btn505

[22] Dojer, N., Bednarz, P., Podsiadło, A., Wilczyński, B. (2013). BNFinder2: Faster Bayesian network learning and Bayesian classification. Bioinformatics, 29(16), 2068–2070. doi:10.1093/bioinformatics/btt323

[23] Martyn Plummer (2003). JAGS: A Program for Analysis of Bayesian Graphical Models Using Gibbs Sampling, Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003), March 20–22, Vienna, Austria. ISSN 1609-395X

[24] Darwiche, A. (n.d.). Modeling and reasoning with Bayesian networks (First paperback edition.). New York: Cambridge Univ Press.

[25] Bundy A., Wallen L. (1984) Non-Monotonic Reasoning. In:Bundy A., Wallen L. (eds) Catalogue of Artificial IntelligenceTools. Symbolic Computation (Artificial Intelligence).Springer, Berlin, Heidelberg

[26] Cozman, F., Mauá, D. (2016). The Complexity of Bayesian Networks Specified by Propositional and Relational Languages.

[27] Kraus, S., Lehmann, D., Magidor, M. (2002). Nonmonotonic Reasoning, Preferential Models and Cumulative Logics.

[28] Casini, G., Meyer, T., Moodley, K., Sattler, U. and Varzinczak, I., 2015, October. Introducing defeasibility into OWL ontologies. In International Semantic Web Conference (pp. 409-426). Springer, Cham.

[29] Kautz, H., Singla, P. (2016). Technical Perspective: Combining logic and probability. Communications of the ACM, 59(7), 106–106. doi:10.1145/2936724

[30] Levi, I. (2010). Probability logic, logical probability, and inductive support. Synthese, 172(1), 97–118. doi:10.1007/s11229-009-9474-3