Abductive and Contrastive Explanations for Bayesian Networks and their Computational Complexity

Tijmen van ter Beek (5961564)
Supervised by Silja Renooij and Hans Bodlaender

Combined Master's Thesis in Computing Science & Artificial Intelligence
at Utrecht University

March 30, 2023

Abstract

Contrastive and abductive explanations are types of explanations that can be used to explain a classification by a classifier. An abductive explanation is a minimal subset of input variables that guarantees the observed classification. A contrastive explanation is a minimal subset of input variables that, when changed, could lead to another predicted class. We show that computing these types of explanations for Bayesian network classifiers (Bayesian networks used as classifiers) is NP-hard for contrastive explanations and co-NP-hard for abductive explanations. We define a number of subproblem for finding contrastive and abductive explanations. For contrastive explanations, we find that the problems of computing any contrastive explanation is already NP-hard for Bayesian networks where inference can be applied in polynomial time. We propose a number of simple algorithms that are able to solve the different subproblems by computing inference for multiple subsets of evidence. Lastly we propose an algorithm that can compute abductive and contrastive explanations more efficiently and reduces redundant calculations by looking at the internal structure and values of the Bayesian network. The algorithm solves a more general problem: finding all assignments to a given set of evidence nodes that satisfy a constraint on a hypothesis node. Finding contrastive and abductive explanations is a special case of the algorithm.

Contents

Chapter 1

Introduction

With the increasing use of computer-based decision making in daily life, such as in self driving cars or image recognition, comes an increasing need to explain why specific choices have been made by the computer. One of many methods for computer-based decision making is the use of Bayesian networks (BNs) [20].

Bayesian networks are representations of joint probabilities that use independencies between variables to structure probabilities in a space-efficient and somewhat intuitive manner using a directed graph. Using this structure we can compute conditional probabilities, which is called inference. An example of such a conditional probability could be: the probability that a person has the flu given they have a fever of 39 degrees Celsius, a cough, and a stuffed nose but no other symptoms. In this case the *conditions* are the fever, the cough, and the stuffed nose. Inference calculates the probabilities for all states of a variable (so "flu" or "not flu" in the example) given the conditions. One of the things inference allows us to do is use a Bayesian network as a classifier, e.g. by predicting the state with the highest probability. As such a BN can be considered an alternative to well-known machine learning techniques. In fact, BNs can be automatically constructed from data, but as a result of their interpretability, manual construction is also possible [4].

Bayesian networks are often used in critical situations [36] where transparency is required in the reasoning of a model: consider a medical domain similar to the example with the flu, where a doctor inputs symptoms and a classifier predicts a disease. In this situation, predictions from the classifier may have significant consequences for the patient. It is important that a doctor can look into the reasoning of the classifier, determine whether he agrees with the system and overrule it if the classifier made a wrong prediction (e.g. because certain edge cases were not included in the model). In this sense Bayesian networks already have a benefit over other types of classifiers such as neural networks, since the internal structure of a BN has a much more well-defined meaning: nodes contain the conditional probabilities for a variable given the possible values of its parents in the graph. However, while one could think Bayesian networks are so simple that they do not require any explanation, this has been argued to be false [8]. In combination with the increased interest in and new insights into explainable AI [34] this leads to a renewed interest in explaining Bayesian networks and BN-based classifiers. However, while many types of classifiers can classify in an amount of time that scales polynomially with their size (the number of input and output variables), applying inference to Bayesian networks is NP-complete [6]. This means that known (and likely all possible) algorithms that compute inference take an amount of time that scales exponentially with the size of the Bayesian network. This in turn means that it becomes increasingly costly to compute inference for larger Bayesian networks. A similar problem arises when we want to *explain* Bayesian networks: some types of explanations rely on inference and have complexities that make them costly to compute for larger Bayesian networks. For example, some types of explanations proposed for classifiers are model-agnostic and can be used for any type of classifier, but require a lot of classifications (such as LIME [37], SHAP [32], and finding contrastive and abductive explanations [10][21]). In the case of BN-based classifiers, this may take a long time due to the complexity of inference underlying the classification itself.

In this thesis we focus on explanation of Bayesian networks. More specifically we focus on the computational complexity of finding contrastive and abductive explanations in Bayesian networks and on the design of more efficient algorithms to this end.

The results of this thesis should help in determining whether abductive and contrastive explanations are a relevant and feasible type of explanation for specific situations, and reduce the time in which these explanations can be found. This is important because Bayesian networks are often used in critical situations [36] where it is crucial that predictions can be double-checked and understood by domain experts. If faster algorithms will be found, this may allow types of explanations to become feasible in situations they are currently not. Especially with new laws (such as the GDPR in the European Union) giving people rights to explanations, computing insightful explanations is an important requirement for the application of Artificial Intelligence and classification in practice.

This thesis is the final project for two master's degrees: *Computing Science* and *Artificial Intelligence*. The Computing Science aspects that are prominent in this thesis are algorithms and complexity, while Artificial Intelligence provides insight into explainability. The concepts of classification and Bayesian networks themselves are positioned on the overlap between the two subjects.

This thesis is structured as follows. In chapters 2 and 3 we will further explain the concept of Bayesian networks and briefly discuss which explanations exist that can be used for Bayesian networks in general and BN-based classifiers. Chapter 4 specifies and formalises inference and classification. Chapters 5 and 6 focus on abductive and contrastive explanations and the complexity of finding those in different situations. Chapter 7 proposes an algorithm that is often able to find the abductive and contrastive explanations more efficiently. Lastly, the conclusion will summarise the relevant findings.

Chapter 2

Existing work

This chapter is split into two parts, beginning with some related work about Bayesian networks and their time complexity. The second part will mention different explanations for Bayesian networks that have been considered. In this chapter we will introduce and use specific notation that will be used throughout this thesis. The notation is also listed in Chapter 3.

2.1 Complexity classes

This thesis will focus on the complexity and complexity classes of several problems. We will describe the complexity of several algorithms using big-O notation, which we will assume the reader is familiar with. Additionally, we will use complexity classes such as P, NP, co-NP, and PP. These classes (except P) provide a lower bound for the complexity of a problem, as they show that no currently known algorithm can solve the problem without for example taking an amount of time that scales exponentially with the input. We assume that the reader is familiar with these complexity classes. Additionally, we will use the concept of **fixed-parameter tractability**, which means that, while the general problem is, for example, NP-hard, as long as some specific parameter is fixed (or bound by some constant), the problem can be solved in polynomial time respective to the other parameters. If the reader wants more information about any of these concepts, information about big-O notation, P and NP can be found in "Introduction to Algorithms" by Cormen et al. [7]. P, NP and other complexity classes are also explained in "Computers and Intractability: A Guide to the Theory of NP-Completeness" [16] and "Computational Complexity - A Modern Approach" [2].

2.2 Bayesian Networks

Bayesian networks can be formalized as follows: a Bayesian network $\mathcal{B}$ is a model for a joint probability distribution that consists of a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$ containing nodes $\boldsymbol{V}_G = V_i ... V_n$ representing discrete variables $\boldsymbol{V}_G = V_i ... V_n$. Note that we use the same name for both the node and the variable. $\forall V_i, V_j \in \boldsymbol{V}_G$: if there is an arc from V_i to V_j in $\boldsymbol{A}_G$, V_i is said to be the parent of V_j, and V_j contains probabilities for V_j conditioned on all possible combinations of V_i and the values of the other parents of V_j in its conditional probability table (CPT). For all pairs of nodes $\forall V_u, V_v$: if V_u and V_v are not directly connected with an arc, the model considers V_u and V_v conditionally independent (for some condition). The direction of the arcs does not have to correspond with the direction of causality.

When using Bayesian networks we often assign evidence nodes $\boldsymbol{E}$ and often one hypothesis node V_h. Nodes in $\boldsymbol{E}$ get assigned observed values $\boldsymbol{e}$, on the basis of which the probabilities in V_h get calculated with the use of inference. Prior probabilities $\Pr(V_h)$ are the probabilities without any observed variables (so without any assignment to the evidence), and posterior probabilities are the conditional probabilities $\Pr(V_h|\boldsymbol{e})$ after the state of $\boldsymbol{E}$ has been observed.

An example of a Bayesian network as visualized in GeNie is shown in Figure 2.1

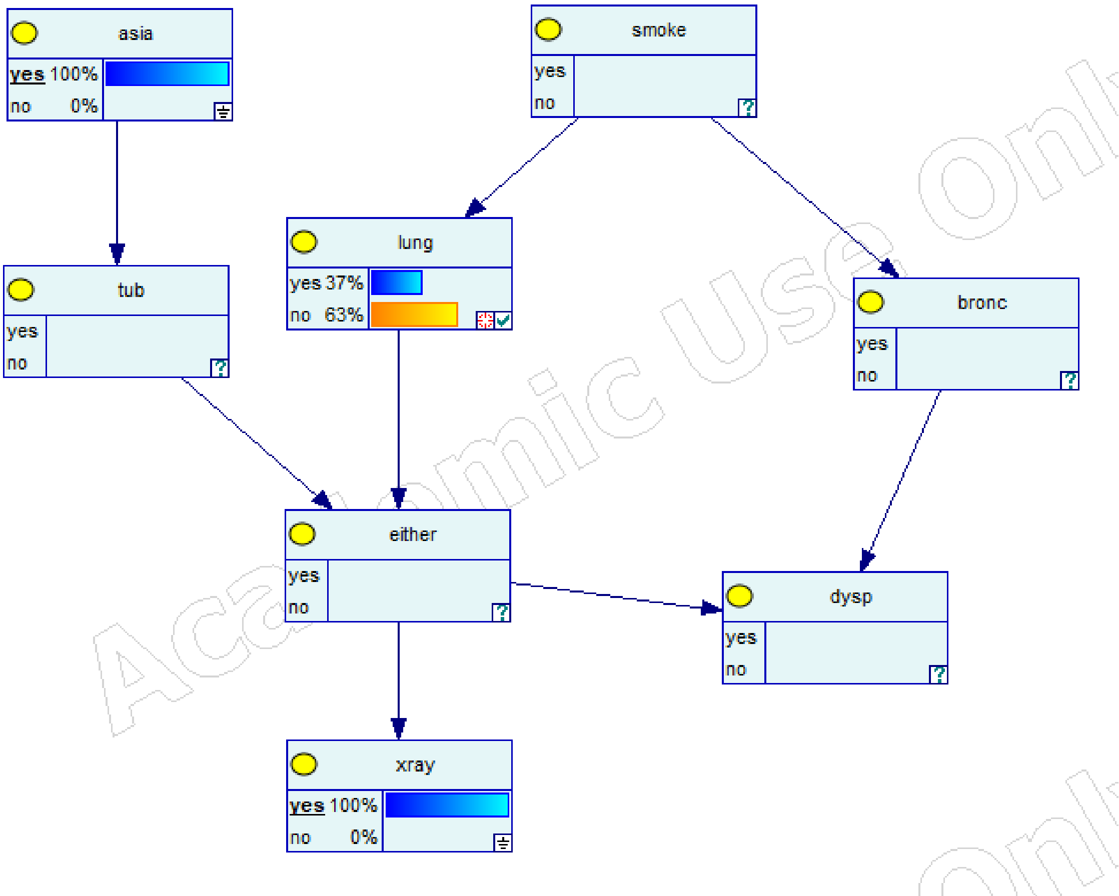


Figure 2.1: A simple Bayesian network known as the Asia-network as visualized in GeNie. The nodes "asia" and "xray" have been given evidence, and the node "lung" has been selected as target/hypothesis node.

A path through the graph not necessarily following the direction of the arcs, i.e. a sequence of adjacent nodes, is called a **chain**. The probabilities for a variable in a Bayesian network can only be influenced by another variable if there is a chain between the variables. However, chains can also be **blocked**. A chain is blocked iff either:

1. the chain passes a node V_x through two incoming arcs (so both arcs point towards V_x) and V_x nor any of its descendants have been assigned evidence.
2. the chain passes a node V_x through at least one outgoing arc, and V_x has been assigned evidence.

If the chain is blocked, the variables cannot affect each other through this chain. If all chains that connect two variables are blocked they are **d-separated** and they are independent given the evidence.

In a Bayesian network-based classifier, the evidence nodes are the input features and the most probable value of the hypothesis node is often used as the predicted class. Classification can then easily be done using inference.

One of the ways to do exact inference is known as the Lauritzen and Spiegelhalter [30] algorithm, which is based on join tree propagation [20]. This algorithm has a running time that generally scales exponentially with the size of the Bayesian network. Assuming P $\neq$ NP, this cannot be prevented, as exact inference has been proven to be NP-hard [6]. An exception to this is when the *treewidth* of the moralization (see below) of the graph is bound by some constant, in which case the problem becomes tractable [25].

2.2.1 Treewidth

To explain the measure of treewidth, it is necessary to discuss some other definitions: moralization, triangulation and tree decomposition.

The moralization of a directed graph is the undirected graph that is obtained when we apply the following steps to a directed graph:

- For every node, we connect all parents of the node.
- We drop all directions: the arcs become edges.

A triangulation of the moralized graph G is a graph that has G as subtree and does not include loops of more than three variables without any pair being adjacent.

A tree decomposition of this triangulated graph G_T is a tree T such that

- each node X_i in T is a bag of nodes which constitute a clique (subgraph where every node is connected to every other node) in G_T
- for every i,j,k, if X_j lies on the path from X_i to X_k in T then $X_i \cap X_k \subseteq X_j$

Treewidth [11] is essentially a measure of how tree-like a graph is. Graphs with a treewidth of 1 are trees. The treewidth of a graph is equal to the minimum of treewidths of all possible tree decompositions. The treewidth of a tree decomposition is one less than the size of its largest clique [25]. It is NP-complete to find the tree decomposition with the smallest treewidth [20]. Finding the treewidth of a graph is also NP-complete [1], though a lot of approximation algorithms exist. There are a number of other ways to compute the treewidth of a graph (i.e. other algorithms for finding the same value), such as the maximum order of a so-called bramble [11].

2.3 Explanation

There are numerous ways a Bayesian network can be explained. Existing research distinguishes 4 types of explanations for Bayesian networks [9]:

- Explanation of evidence

- Explanation of reasoning
- Explanation of the model
- Explanation of decisions

Additionally, common ways to distinguish explanations of classifiers are the distinction between *local* and *global* explanations and between *model-based* and *model-agnostic* [37] explanations. Global explanations explain the model as a whole, while local (also known as instance-dependent) explanations explain a specific classified instance. Model-agnostic explanations can be applied to any classification model but inherently suffer the drawback that they can have no knowledge about the inner workings of the classifier and thus are limited to explaining its input-output behaviour instead of its reasoning. In contrast, model-based explanations only work for a specific model, and could therefore use information about the reasoning of the model.

Recently, it was proposed to use more insights from the social sciences in explainable AI [34]. Specifically, in social sciences research has been done on how humans explain things and what we expect from an explanation. The main proposals are to pay more attention to four aspects:

- Explanations are contrastive. This means that people generally want to know why something was the case instead of another option.
- Explanations are selected. Humans do not explain everything, we select a few reasons (out of possibly very many) which we consider the main reasons, which we give as reasons.
- Probabilities do not matter. Giving reasons is much more important than the probability of something happening (as an explanation).
- Explanations are social. Explanations are relative to the explainer's beliefs about the explainee's beliefs.

In the following sections some existing explanations for Bayesian networks are discussed and categorized given the aforementioned categories.

2.3.1 Explanation of evidence

In explanation of evidence, an explanation is seen as the configuration of a subset of variables present in the Bayesian network, given evidence [9]. This type of explanation is also known as abduction.

Abduction deals with the problem of finding the most probable values for a number of nodes [23]. This problem is known as MPE (most probable explanation) or (partial) MAP (maximum a posteriori hypothesis) depending on whether we have full or partial evidence.

Definition 2.3.1. MPE
Instance: A Bayesian network $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, where the nodes are split between evidence nodes $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ with an evidence assignment $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ and explanation variables $\boldsymbol{M} = \boldsymbol{V}_G \backslash \boldsymbol{E}$.
Output: $\underset{\boldsymbol{m} \in \Omega(\boldsymbol{M})}{\arg\max}(\Pr(\boldsymbol{m}|\boldsymbol{e}))$ I.e. the most probable assignment $\boldsymbol{m}$ to the nodes in $\boldsymbol{M}$ given evidence $\boldsymbol{e}$.

Definition 2.3.2. (Partial) MAP
Instance: A Bayesian network $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, where the nodes are split between evidence nodes $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ with an evidence assignment $\boldsymbol{e} \in \Omega(\boldsymbol{E})$, a set of explanation variables $\boldsymbol{M} \subset (\boldsymbol{V}_G \backslash \boldsymbol{e})$ and intermediate variables $\boldsymbol{I} = \boldsymbol{V}_G \backslash (\boldsymbol{E} \cup \boldsymbol{M})$.
Output: $\underset{\boldsymbol{m} \in \Omega(\boldsymbol{M})}{\arg\max}(\Pr(\boldsymbol{m}|\boldsymbol{e}))$ I.e. the most probable assignment $\boldsymbol{m}$ to the nodes in $\boldsymbol{M}$ given evidence $\boldsymbol{e}$.

The difference between abduction and inference is caused by states with the highest probabilities for a set of nodes not necessarily being the most likely joint value assignment of those nodes. We will provide an example of the difference between Partial MAP and inference. Consider three binary variables A, B and C. The internal structure of the network is not important for this example. The joint distribution can be described as follows: there is always one variable false while two are true but the probabilities for the three possible configurations $\{\{A = T, B = T, C = F\}\{A = T, B = F, C = T\}\{A = F, B = T, C = T\}\}$ are equal. In this case the probability for each node to be true (the result of inference without any evidence nodes) is $\frac{2}{3}$ but the most likely joint value assignment (similar to the result of Partial MAP without evidence nodes) is definitely not $\{True, True, True\}$. MPE and most similar problems are NP-complete but often fixed parameter tractable [23]. Efficient ways to calculate MPE have been researched [17][43].

A related concept is that of MAP-independence [24]. MAP-independence is a notion that builds on the definition of conditional independence as suggested by Pearl [35]. Given variables A, B and C, A is MAP-independent from B given C if B could not influence the explanation for A if it were observed. In this case the explanation refers to the most likely value for variables as discussed before. This measure tries to capture to what extent a variable is relevant for an explanation. MAP-independence is co-NP^{PP}-complete and is fixed parameter tractable given bounded treewidth. A basic algorithm to calculate MAP-independence is given that calculates MAP-independence from a set of variables R in $O(2^{|R|})$ times the time required for the MAP-computation.

2.3.2 Explanation of reasoning

Explanation of reasoning attempts to justify the conclusion of the model and how it was obtained [9]. There are many different types of explanations of reasoning, some of which we will discuss in this section. In this section, explanations will be grouped by the explanation they provide, which means that some frameworks that provide multiple types of explanations will be discussed in several different paragraphs.

Elvira is a software package that was developed by several Spanish universities in a joint research project, and contains multiple types of explanations [28]. It will therefore be mentioned throughout the following paragraphs.

Abductive and contrastive explanations In the case of Bayesian network classifiers, proposed local explanations include abductive/sufficient/PI- explanations [39] and contrastive explanations [21][22][19]. Abductive, sufficient or PI-explanations are different names for a minimal subset of the observations that would be sufficient to conclude the obtained predicted class. In this thesis the term abductive explanation will be used. While they are related concepts, abductive explanations are not to be confused with abduction which was described in Section 2.3.1. Both contrastive and abductive explanations will be described, defined and discussed in more depth in their respective chapters later in this thesis. As an illustration of abductive explanations, consider the following first order logic (FOL) model. The model classifies a variable E to be true or false depending on the observations A, B, C, D, where e.g. the literal A denotes that observation A had value $True$ and $\neg A$ denotes that observation A had value FALSE: $[(A \wedge B) \vee \neg C \vee D \implies E] \wedge [\neg((A \wedge B) \vee \neg C \vee D) \implies \neg E]$. An observation $(A, B, C, \neg D)$ would lead to prediction E. The only abductive explanation for this observation would be the pair (A, B) as the prediction would remain the same as long as these observations keep their value.

An efficient way to calculate abductive explanations is discussed in [39] where the authors find the explanations by converting Bayesian networks to a so-called OBDD and recursively calculating the explanations. This approach does, however, not work for all Bayesian networks.

Contrastive explanations are the minimal subset of observations that would have to change to get another prediction [19]. If we consider the example from before, valid contrastive explanations would be the sets $\{A\}$ and $\{B\}$. There are two types of this explanation: one could only specify the observations that would have to change to get another prediction [19], or one could include the values that these observations would have to take [22]. Both versions are used. In case of binary variables, there is of course no difference. Contrastive explanations can be calculated from just the input and output pairs, but the proposed algorithms by Koopman (2020) require a large number of classifications as they classify for a significant portion of the possible sets of evidence [22]. If any monotonicity relations are known these may help with speeding up the calculation of the sufficient and contrastive explanations [21]. Abductive and contrastive explanations can also be viewed as Minimal Unsatisfiable Subsets (MUS) and Minimal Correction Subsets (MCS) [19]. This allows for applying known algorithms to find abductive and contrastive explanations as long as there is a FOL representation of the classifier, and allows for using an established relationship between MUSs and MCSs, the minimal hitting set relationship (this will be described in Chapter 5), to find abductive explanations if we have contrastive explanations and vice versa. Though algorithms are proposed to find contrastive explanations, their complexity has not been discussed.

Chains of reasoning Elvira [28] uses chains of reasoning originally proposed in INSITE [40]. First, influential evidence nodes are determined by looking at the change in value for the hypothesis node when the observation is omitted. To find the chains of reasoning, all chains between influential observations and the hypothesis node are determined. Using cross-entropy, a distinction is made between significant and insignificant chains. Lastly it is determined whether a chain c is conflicting or consistent with the evidence, by determining if c has the same or the opposite effect on the hypothesis node as the total evidence. BANTER [18] uses a similar concept, though the computations are different. Instead of using the cross-entropy to see how large the difference is, BANTER uses the mutual information. The mutual information is defined as $I(a_i; b_j) = log(P(a_i|b_j)/P(a_i))$. Another similar approach uses the Hellinger distance instead of the cross-entropy or mutual information [26].

Important internal nodes. Some explanations include an indication of which nodes are most important. As seen in the previous paragraphs, there are a number of ways to determine which evidence nodes are most relevant (the influential evidence nodes in Elvira and INSITE for example). Research has also attempted to determine which internal nodes are most relevant to present to the user.

One of the proposed methods converts the Bayesian network to a flow network and then uses the Edmonds-Karp algorithm to calculate the minimum cut [31]. The nodes connected to the minimum cut are proposed to be the most important nodes as they contain many paths from the evidence to the hypothesis node. This is part of a larger explanation that also includes clusters of internal nodes determined using the mutual information that are then converted to explanations. The Edmonds-Karp algorithm has a running time of $O(V \cdot E^2)$ where V is the number of vertices and E the number of edges [15].

Another proposed approach for finding important internal nodes is to consider the Markov blanket of the hypothesis node important and present it to the user [27]. This approach is part of a three level progressive explanation: the first level calculates the important evidence nodes similarly to the approach by INSITE. The second level consists of the internal nodes that are part of the Markov blanket of the evidence node. The third level includes a measure of the effect of evidence nodes on the hypothesis node. Finding the Markov blanket is trivial and does not take a significant amount of time.

Local model-agnostic explanations Local model-agnostic explanations such as LIME[37] and SHAP[32] are also explanations of reasoning, as they attempt to explain the connection between the input and output of a classification. Both SHAP and LIME are explanations that have been proposed as a model-agnostic explanation for black-box classifiers.

LIME creates a local model by training a new classifier on the output of the existing classifier. Given output O as a result of input I it makes small changes to I resulting in $I_1 ... I_n$. This data is then used as input for the original classifier resulting in output $O_1 ... O_n$. The input-output pairs $(I_1, O_1) ... (I_n, O_n)$ are then used to train a new classifier. Often for the new classifier a type of classifier is chosen that is easy to understand (e.g. a linear classifier or naive Bayes).

SHAP uses Shapley values for each input variable to explain the model. To find the Shapley values for a classifier with n input variables, 2^n different models are required, one for every subset of the input variables. For each of these subsets the outcome is then computed with the input filled in. Every input variable then occurs in half (2^{n-1}) of the predictions. Given an input variable x, for every prediction with input variables $x \in V, V \vdash p$, the outcome is compared to the result with the same input variables except l: $L/l \vdash q$. These differences are then weighted to calculate the effect of variable l. The weights are defined such that every subset size contributes equally $(1/n)$ to the total Shapley value. Within these subset sizes every result has the same weight. As an example consider input assignment $v_a = a, v_b = b, v_c = c$ that predicts a value v for some class p. Assume the following predictions:

Labels	Prediction
$\varnothing$	0.5
a	0.1
b	0.4
c	0.6
a, b	0.3
a, c	0.9
b, c	0.7
a, b, c	0.8

Let us calculate the Shapley value for assignment $v_a = a$. There is one subset of size 1 that includes a, namely $\{a\}$. The difference with the set without a ($\varnothing$) is -0.4. There are two subsets of size 2 that include a, namely $\{a, b\}$ and $\{a, c\}$. These have values -0.1 and 0.3. There is one subset of size 3 that includes a, namely $\{a, b, c\}$. Its value is 0.1. The total Shapley value of $v_a = a$ is then $\frac{1}{3} \cdot -0.4 + \frac{1}{6} \cdot -0.1 + \frac{1}{6} \cdot 0.3 + \frac{1}{3} \cdot 0.1 = -\frac{1}{15}$.

Interestingly, we can calculate something similar to a Shapley value using Bayesian networks without having to construct multiple networks. BN classifiers can still make predictions when removing nodes from the evidence set, so one can use the same Bayesian network to make predictions with fewer input variables. However, this can result in a different values than when we would create new BN classifiers, for example when using machine learning to create the Bayesian network. Calculating exact Shapley values for Bayesian networks is NP-hard even with limited treewidth [41].

2.3.3 Explanation of the model

Explanation of the model is also called static explanation [28] and explains the model to a user unfamiliar with Bayesian networks. These explanations are by definition global and model-based, as they explain a Bayesian network without considering a specific classification.

Explanation of nodes Elvira describes nodes by giving their name, states, parents, children, prior and posterior odds, in addition to a purpose and importance factor that are defined by the creator of the network [28].

Another proposed approach is to generate textual descriptions that include information about the conditional probability table of the node, using words such as "unlikely", "fairly unlikely" or "very likely" to describe different probabilities [12]. This results in explanations such as "Cold very commonly (p=0.9) causes sneezing."

Explanation of links In Elvira, links between nodes representing ordinal variables (where the states have an ordering, often FALSE < TRUE is used for example) are explained by checking whether all higher values of the parent nodes result in distinctly higher values of the child node (called positive), result in distinctly lower values of the child node (called negative), do not have any influence (null) or none of the above (undefined) [28]. Elvira is also able to change the thickness of the arcs to indicate the strength of the influence [29].

Explanation of the network Elvira offers additional verbal explanations of the network, and allows the user to select thresholds of the aforementioned importance factor to only show a smaller network that is easier to understand for the user [28].

Other systems that visualize Bayesian networks in some way are HUGIN [33], GeNie [13] and DIAVAL [14]. HUGIN and GeNie are tools that allow the user to create, modify and use Bayesian networks similar to Elvira. DIAVAL is a system that uses a Bayesian network to diagnose heart disease. These systems mostly display the graph structure of the network and sometimes visualize the CPTs of the nodes.

2.3.4 Explanation of Decisions

Explanation of decisions consists of techniques that describe whether the user is ready to make a decision. This includes for example sensitivity analysis. This type of explanations was recently defined by Derks and De Waal [9]. One of the explanations that falls into this category of explanations is the same-decision probability [5]. This is the probability that the same decision would have been made if the true state of unobserved variables was known and is related to the aforementioned concept of MAP-independence. The problem of finding the same-decision probability is PP^{PP}-complete. An algorithm that is fixed-parameter tractable with respect to treewidth has been proposed that gives a bound on the same-decision probability. The same-decision probability is a measure of how certain an outcome is, and can for example be used to determine whether more tests should be done. Explanation of decisions is a newly proposed category of explanations, and is in a sense different from the other categories: while the other categories directly explain some aspect of a model or its classification, this category "explains" whether a user can safely make a decision based on the output of the model. We could consider this type of explanation an expansion of the model to include a measure of uncertainty, rather than a conventional "explanation".

Chapter 3

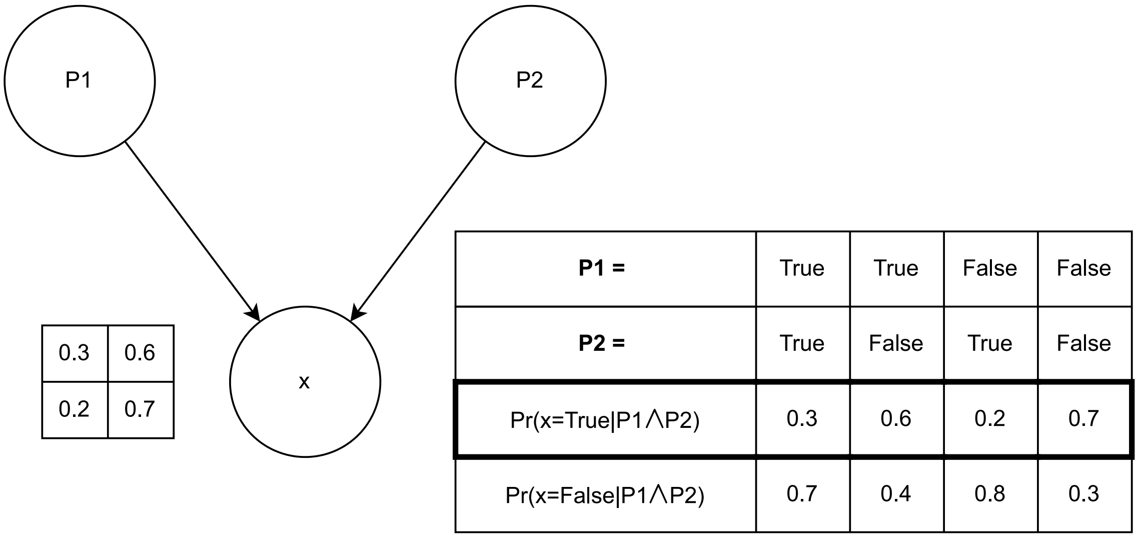
Notation

In the notation of Bayesian networks we will use the following symbols:

$G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$	The directed graph G containing the structure of the Bayesian network. It consists of nodes $\boldsymbol{V}_G$, and arcs $\boldsymbol{A}_G$.
$V_i \in \boldsymbol{V}_G$	Node V_i in the directed graph. This also represents random variable V_i in a Bayesian network.
$(V_i, V_j) \in \boldsymbol{A}_G$	The arc from node V_i to node V_j in the directed graph.
$\boldsymbol{\rho}_{V_i}$	The parents of node V_i. All nodes V_j for which arc (V_j, V_i) exists.
$\boldsymbol{\rho}^*_{V_i}$	The ancestors of node V_i. Recursively defined as the parents of V_i and all ancestors of these parents.
$\boldsymbol{\sigma}_{V_i}$	The children of node V_i. All nodes V_j for which arc (V_i, V_j) exists.
$\boldsymbol{\sigma}^*_{V_i}$	The descendants of node V_i. Recursively defined as the children of V_i and all descendants of these children.
$\boldsymbol{V}$	A set of random variables $\{V_1 ... V_n\}$.
v_i	A value assignment of node V_i of the Bayesian network.
$\boldsymbol{s}$	A value assignment for a set of nodes $\boldsymbol{S} \in \boldsymbol{V}_G$.
$c[n]$	The nth assignment in a collection n.
$\Pr(v_h)$	The probability of hypothesis v_h without any given evidence $\boldsymbol{e}$. Also known as the prior probablity of v_h.
$\Pr(v_h \mid \boldsymbol{e})$	The probability for hypothesis v_h given evidence $\boldsymbol{e}$. Also known as the posterior probablity of v_h.
$\boldsymbol{e}^+$	All evidence in the upper graph of the current node.
$\boldsymbol{e}^-$	All evidence in the lower graph of the current node.
$\boldsymbol{e}^{V_x/V_y}$	All evidence in both the descendants and ancestors of V_x, except the evidence that would become disconnected if we remove the arc between V_x and V_y.
$A \models B$	Semantic consequence; given A, B always holds. In this thesis mostly used to state that given input A a model such as a Bayesian classifier predicts B regardless of the observed values for all evidence nodes that are not in A.
$A \not\models B$	$A \models B$ does not hold.
$\boldsymbol{A} = \underline{\boldsymbol{b}}$	The variables from $\boldsymbol{b}$.
$\Omega(V_x)$	The set of all assignments of states to V_x. Also known as the statespace of V_x.
$\Omega(\boldsymbol{X})$	The set of all combinations of assignments of states to $\boldsymbol{X}$. $\forall \boldsymbol{x} \in \Omega(\boldsymbol{X}) : \underline{\boldsymbol{x}} = \boldsymbol{X}$.
$\Omega(\boldsymbol{X}; \boldsymbol{y})$	The set of all combinations of assignments of states to $\boldsymbol{X}$ where no assignment is the same as in $\boldsymbol{y}$.
$\boldsymbol{a} = \boldsymbol{b}/\boldsymbol{C}$	The assignments from $\boldsymbol{b}$ without the variables in $\boldsymbol{C}$.
$\boldsymbol{a} = \boldsymbol{B} \in \boldsymbol{c}$	The subset of $\boldsymbol{c}$ that corresponds to the variables in $\boldsymbol{B}$. $\boldsymbol{a} \subseteq \boldsymbol{c}$, $\boldsymbol{a} \in \Omega(\boldsymbol{B})$.
T	Abbreviation for the state TRUE for Boolean variables.
F	Abbreviation for the state FALSE for Boolean variables.

To be able to reason about the complexity of explanations in Bayesian networks, some additional notations will be used:

N	The number of nodes in the Bayesian network.
M	The number of arcs in the Bayesian network. $M \leq \frac{N(N-1)}{2}$.
D	The maximum degree in the Bayesian network. $\frac{2M}{N} \leq D \leq N - 1$.
TW	(An upper bound on) the treewidth of the Bayesian network.
$\lvert x \rvert$	The number of elements in a collection x.
τ_i	The size of the probability table of variable V_i. $\tau_i = (\lvert V_i \rvert - 1) \cdot \prod_{V_j \in \boldsymbol{\rho}_G i} \lvert V_j \rvert$.
S	An upper bound/maximum of the number of states any variable has.
$\boldsymbol{\rho}_{max}$	An upper bound/maximum of the number of parents any variable has. $\boldsymbol{\rho}_{max} \leq D$.
$\boldsymbol{\sigma}_{max}$	An upper bound/maximum of the number of children any variable has. $\boldsymbol{\sigma}_{max} \leq D$.
τ_{max}	An upper bound/maximum of largest probability table in the network. $\tau_{max} \leq S^{\boldsymbol{\rho}_{max}+1}$.
I	The runtime of inference.



0.3	0.6
0.2	0.7

P1 =	True	True	False	False
P2 =	True	False	True	False
Pr(x=True|P1∧P2)	0.3	0.6	0.2	0.7
Pr(x=False|P1∧P2)	0.7	0.4	0.8	0.3

Figure 3.1: A simplified and non-simplified CPT containing the same conditional probabilities.

We will use a simplified version of conditional probability tables (CPTs). A simplified and non-simplified CPT are shown in Figure 3.1. In the simplified CPT only the conditional probabilities for $X = True$ are shown. The top of the table corresponds to the left parent having value TRUE, while the bottom corresponds with the left parent having value FALSE. The left side of the table corresponds to the right parent having value TRUE, and the right side of the table corresponds with the right parent having value FALSE. This will only be used for binary nodes having up to two binary parents.

In this thesis we will use NP-hardness and NP-completeness proofs. For information on how these proofs work we refer to [7]. In short, an NP-hardness proof of problem $P1$ contains a polynomial-time reduction (with respect to the input) from another NP-hard problem $P2$ to $P1$, and proves that a solution for $P1$ corresponds to a solution of problem $P2$ and vice versa. For a problem to be NP-complete, it must be an NP-hard decision problem, and it must be in NP. I.e. a solution (yes-instance) must be verifiable in polynomial time (again with respect to the original input size) using a certificate that contains evidence of the yes-instance.

A decision problem is a problem where the possible answers are yes and no (e.g. "is $x \in \mathbb{Z}$ prime?").

Chapter 4

Inference & Classification

Since these will be used throughout this thesis, we will give a formal definition of inference and classification and discuss the complexity of both.

4.1 Inference

Inference is the process of computing prior or posterior probabilities for assignments to variables in a Bayesian network. There are multiple ways to achieve this, such as Pearls algorithm [35] and the Lauritzen and Spiegelhalter algorithm [30]. Inference has been shown to be $\#P$-hard [38].

Definition 4.1.1. Inference
Instance $\langle \mathcal{B}, \boldsymbol{e}, v_h, V_h \rangle$**:** A Bayesian network $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$, and a hypothesis $v_h \in \Omega(V_h)$ where $V_h \in \boldsymbol{V}_G$.
Output: The probability $\Pr(v_h|\boldsymbol{e})$

We can also define a related decision problem:

Definition 4.1.2. Inference decision problem
Instance $\langle \mathcal{B}, \boldsymbol{e}, v_h, x, V_h \rangle$**:** A Bayesian network $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$, a hypothesis $v_h \in \Omega(V_h)$ where $V_h \in \boldsymbol{V}_G$ and a constant x for which $0 \leq x \leq 1$.
Question: Is the probability $\Pr(v_h|\boldsymbol{e})$ equal to x?

This problem is at least NP-hard [6]

4.2 Classification

For classification the result of a hypothesis node is converted to a predicted class. A common method is to take the state of the hypothesis variable with the highest probability, but other options are possible. We will henceforth assume this method is used. We can then define this form of classification as follows:

Definition 4.2.1. Bayesian network classification decision problem
Instance $\langle \mathcal{B}, \boldsymbol{e}, V_h, \pi \rangle$**:** A Bayesian network $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ (input), a hypothesis node $V_h \in \boldsymbol{V}_G$, a state $\pi \in \Omega(V_h)$.
Output: True if $\pi = \underset{v_h \in \Omega(V_h)}{\operatorname{argmax}} \Pr(v_h|\boldsymbol{e})$, False otherwise.

From inference as defined in Definition 4.1.1 we can easily get a classification in polynomial time by comparing the probabilities of the states of V_h. We will demonstrate that we can solve the inference decision problem as defined in Definition 4.1.2 in polynomial time given the results of classification on a slightly modified Bayesian network. This means that classification has a complexity that is at least as hard as the inference decision problem and at most as hard as inference.

Theorem 1. Bayesian network classification is at least as hard as the inference decision problem and therefore NP-hard.

Proof.

Transformation: Suppose we are given an instance $\langle \mathcal{B}, \boldsymbol{e}, v_h, x, V_h \rangle$ of the inference decision problem. We now build two instances of the classification problem as follows. For the first instance, we transform the Bayesian network to a network $\mathcal{B}'_1$ by adding a node $V_{h'1}$ as a child of V_h that is the new hypothesis node. We create its CPT as described below such that the network predicts class True if and only if $\Pr(v_h|\boldsymbol{e})$ is at least x. For the second instance, we transform the Bayesian network to a network $\mathcal{B}'_2$ by adding a node $V_{h'2}$ as a child of V_h that is the new hypothesis node. We create its CPT as described below such that the network predicts class True if and only if $\Pr(v_h|\boldsymbol{e})$ is at most x. An example is given in Figure 4.1. We can then classify for both of these networks, and if both of them predict True we know that this probability in the original network was equal to x. We assume that this classifier predicts True if the probabilities for both True and False are 0.5 (so in case of ties).

To get the required output, we will use the following CPTs for $V_{h'1}$ and $V_{h'2}$. When $x = 0$ or $x = 1$, some of these formulas are undefined, and we assume the min and max functions then take the other (defined) value. Note that these CPTs only depend on x and are normal numerical CPTs once x is filled in.

version 1: $V_{h'1}$
$\Pr(V_{h'1} = T|V_h = v_h) = \min(1, \frac{1}{2x})$.
$\Pr(V_{h'1} = T|V_h \neq v_h) = \max(0, \frac{0.5-x}{1-x})$ for any other state than v_h of V_h.
version 2: $V_{h'2}$
$\Pr(V_{h'1} = T|V_h = v_h) = \max(0, \frac{-0.5+x}{x})$.
$\Pr(V_{h'1} = T|V_h \neq v_h) = \min(1, \frac{1}{2-2x})$ or any other state than v_h of V_h.

This describes a CPT such as the one in Figure 4.1 but for any number of states for V_h. Using this transformation we get two networks $\mathcal{B}'_1$ and $\mathcal{B}'_2$ that still use evidence $\boldsymbol{e}$. The difference between the networks is in the two new hypothesis nodes $V_{h'1}$ and $V_{h'2}$. If both predict True the probability of v_h is equal to x. We can use this to solve the inference decision problem using classification.

Polynomial: The size of the altered Bayesian network is only proportional to the size of the original network, and we need to apply classification a constant number of times, making this reduction easily possible in polynomial time.

Correctness: The correctness of this reduction follows directly from two lemmas:

Lemma 1.1. If $\langle \mathcal{B}'_1, \boldsymbol{e}, V_{h'1}, \pi = \text{True} \rangle$ and $\langle \mathcal{B}'_2, \boldsymbol{e}, V_{h'2}, \pi = \text{True} \rangle$ are positive instances of the classification problem, $\langle \mathcal{B}, \boldsymbol{e}, v_h, x, V_h \rangle$ is a positive instance of the inference problem.

I.e. in every situation where both classifiers predict True, x must be equal to $\Pr(v_h|\boldsymbol{e})$.

Proof. A solution for the classification problems means that our two versions both predict True. Using the definition of our CPT, this means that

$$\begin{aligned} &\Pr(V_{h'1} = T|\boldsymbol{e}) \geq 0.5 \\ \iff &\Pr(V_{h'1} = T|v_h) \cdot \Pr(v_h|\boldsymbol{e}) + \Pr(V_{h'1} = T|\neg v_h) \cdot \Pr(\neg v_h|\boldsymbol{e}) \geq 0.5 \\ \iff &\min(1, \frac{1}{2x}) \cdot \Pr(v_h|\boldsymbol{e}) + \max(0, \frac{0.5-x}{1-x}) \cdot (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5 \end{aligned}$$

and

$$\begin{aligned} &\Pr(V_{h'2} = T|\boldsymbol{e}) \geq 0.5 \\ \iff &\Pr(V_{h'2} = T|v_h) \cdot \Pr(v_h|\boldsymbol{e}) + \Pr(V_{h'2} = T|\neg v_h) \cdot \Pr(\neg v_h|\boldsymbol{e}) \geq 0.5 \\ \iff &\max(0, \frac{-0.5+x}{x}) \cdot \Pr(v_h|\boldsymbol{e}) + \min(1, \frac{1}{2-2x}) \cdot (1 - \Pr(v_h|)) \geq 0.5 \end{aligned}$$

We must now prove that this is only true when $x = \Pr(v_h|\boldsymbol{e})$ Let us distinguish the situations where $x \leq 0.5$ and $x > 0.5$:

Case 1: $x \leq 0.5$: in this case $\min(1, \frac{1}{2x}) = 1$, $\max(0, \frac{0.5-x}{1-x}) = \frac{0.5-x}{1-x}$, $\max(0, \frac{-0.5+x}{x}) = 0$ and $\min(1, \frac{1}{2-2x}) = \frac{1}{2-2x}$. We get the formulas $\Pr(v_h|\boldsymbol{e}) + \frac{0.5-x}{1-x} \cdot (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5$ and $\frac{1}{2-2x} \cdot (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5$.

$$\begin{gathered}
\Pr(v_h|\boldsymbol{e}) + \frac{0.5-x}{1-x} \cdot (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5 \\
\iff \Pr(v_h|\boldsymbol{e}) \cdot (1-x) + (0.5-x) \cdot (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5 \cdot (1-x) \\
\iff \Pr(v_h|\boldsymbol{e}) - x\Pr(v_h|\boldsymbol{e}) + (0.5 - 0.5\Pr(v_h|\boldsymbol{e})) - (x - x\Pr(v_h|\boldsymbol{e})) \geq 0.5 - 0.5x \\
\iff \Pr(v_h|\boldsymbol{e}) - x\Pr(v_h|\boldsymbol{e}) + 0.5 - 0.5\Pr(v_h|\boldsymbol{e}) - x + x\Pr(v_h|\boldsymbol{e}) \geq 0.5 - 0.5x \\
\iff 0.5\Pr(v_h|\boldsymbol{e}) \geq 0.5x \\
\iff \Pr(v_h|\boldsymbol{e}) \geq x
\end{gathered}$$

and

$$\begin{gathered}
\frac{1}{2-2x} \cdot (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5 \\
\iff 1 - \Pr(v_h|\boldsymbol{e}) \geq 1 - x \\
\iff -\Pr(v_h|\boldsymbol{e}) \geq -x \\
\iff \Pr(v_h|\boldsymbol{e}) \leq x
\end{gathered}$$

Together this means that $\Pr(v_h|\boldsymbol{e}) = x$ must hold.

Case 2: $x > 0.5$: in this case $\min(1, \frac{1}{2x}) = \frac{1}{2x}$, $\max(0, \frac{0.5-x}{1-x}) = 0$, $\max(0, \frac{-0.5+x}{x}) = \frac{-0.5+x}{x}$ and $\min(1, \frac{1}{2-2x}) = 1$. We get the formulas $\frac{1}{2x} \cdot \Pr(v_h|\boldsymbol{e}) \geq 0.5$ and $\frac{-0.5+x}{x} \cdot \Pr(v_h|\boldsymbol{e}) + (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5$.

$$\begin{gathered}
\frac{1}{2x} \cdot \Pr(v_h|\boldsymbol{e}) \geq 0.5 \\
\iff \Pr(v_h|\boldsymbol{e}) \geq x
\end{gathered}$$

and

$$\begin{gathered}
\frac{-0.5+x}{x} \cdot \Pr(v_h|\boldsymbol{e}) + (1 - \Pr(v_h|\boldsymbol{e})) \geq 0.5 \\
\iff (-0.5+x) \cdot \Pr(v_h|\boldsymbol{e}) + x - x\Pr(v_h|\boldsymbol{e}) \geq 0.5x \\
\iff -0.5\Pr(v_h|\boldsymbol{e}) + x\Pr(v_h|\boldsymbol{e}) + x - x\Pr(v_h|\boldsymbol{e}) \geq 0.5x \\
\iff -0.5\Pr(v_h|\boldsymbol{e}) \geq -0.5x \\
\iff \Pr(v_h|\boldsymbol{e}) \leq x
\end{gathered}$$

Together this means that $\Pr(v_h|\boldsymbol{e}) = x$ must again hold.

■

Lemma 1.2. If $\langle \mathcal{B}, \boldsymbol{e}, v_h, x \rangle$ is a positive instance of the inference problem, $\langle \mathcal{B}'_1, \boldsymbol{e}, V_{h'2}, \pi = \text{TRUE} \rangle$ and $\langle \mathcal{B}'_2, \boldsymbol{e}, V_{h'2}, \pi = \text{TRUE} \rangle$ are positive instances of the classification problem. I.e.

if $\Pr(v_h) = x$, both classifiers must predict TRUE.

Proof. From $\Pr(v_h) = x$ we know that $\Pr(v_h) \leq x$ and $\Pr(v_h) \geq x$ both hold. Since the inequalities above are derived using bi-implications, we can use them to determine that $\min(1, \frac{1}{2x}) \cdot \Pr(v_h) + \max(0, \frac{0.5-x}{1-x}) \cdot (1 - \Pr(v_h)) \geq 0.5$ and $\max(0, \frac{-0.5+x}{x}) \cdot \Pr(v_h) + \min(1, \frac{1}{2-2x}) \cdot (1 - \Pr(v_h)) \geq 0.5$ hold, so the classifier will predict TRUE in both cases, since both values are greater than or equal to 0.5. ■

From the above, we can see that there is a polynomial transformation from the inference problem to the classification problem. As inference is NP-hard [6], it follows that classification is NP-hard under Turing-reductions. □

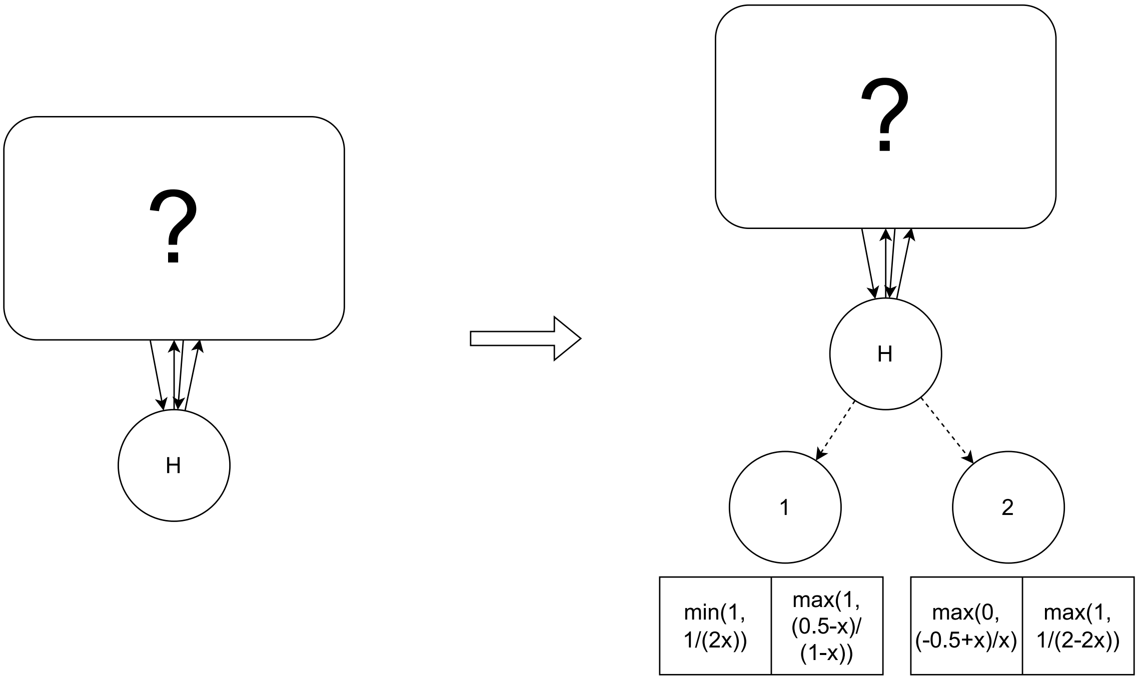


Figure 4.1: The described transformation for a Bayesian network where the hypothesis node is binary. The arrows and question-mark box indicate a unknown Bayesian network that can have any connections to H. Both networks are shown at the same time in this figure, the dotted line indicates that only one of those situations occurs in one network. $V_{h'1}$ and $V_{h'2}$ are called 1 and 2 respectively for the two new networks.

4.2.1 Static evidence nodes

In classification it is commonly the case that every classification uses a different combination of observed states for the same variables. In Bayesian networks and classifiers however, variables do not require an observation. This means classifications using the same Bayesian classifier can differ not only in the assignment of different states to evidence nodes, but the evidence nodes can also change, and even increase or decrease in number. In this thesis however, we will be assuming that the evidence nodes are **static**. By static we mean that the evidence nodes do not change and are a property of the Bayesian network: no evidence nodes can remain unobserved, and no evidence nodes can be added for specific classifications.

Assumption 1. $\underline{e}$ *is constant within the Bayesian network. i.e. every classification or inference will use the same evidence variables.*

Note that this affects the use of $\models$ and $\not\models$: $\boldsymbol{e} \models \pi$ means that given assignments $\boldsymbol{e}$, prediction π is always predicted by the Bayesian classifier. The statement $\boldsymbol{s} \models \pi$ where $\boldsymbol{s} \subset \boldsymbol{e}$ also means that given assignments $\boldsymbol{s}$, prediction π is always predicted by the Bayesian classifier. However, in this case not all evidence variables have been assigned a state. In this case the notation means that π is predicted using any combination of evidence assignments that contains $\boldsymbol{s}$: $\boldsymbol{s} \models \pi \iff \forall \boldsymbol{s'} \in \Omega(\underline{e} \backslash \underline{s}) : \boldsymbol{s} \cup \boldsymbol{s'} \models \pi$.

However, in some situations we could work around this assumption: one could allow the evidence nodes to have an extra option "no assignment". This would transform a Bayesian classifier to a (not ordinary Bayesian) classifier with static evidence without losing classification options. This would then allow us to use the definitions of contrastive and abductive explanations (which will be introduced later in this thesis) for these classifiers. However, this does not work for some of the algorithms later in this thesis that need a normal Bayesian network with static evidence. Specifically, d-separation does not work the same anymore: whether information can pass through evidence nodes now depends on the evidence assignment.

Chapter 5

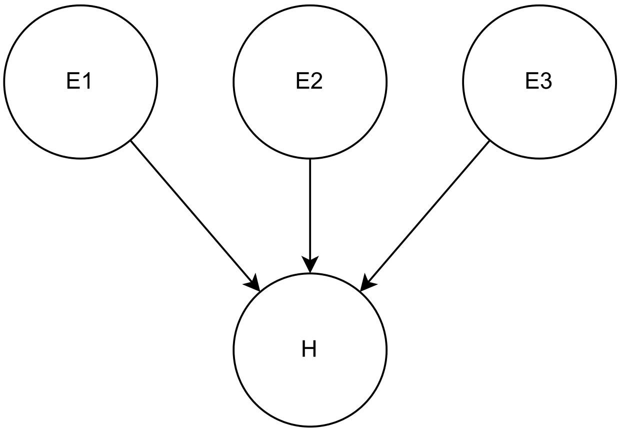
Contrastive Explanations

5.1 Defining the problems

Firstly, there are several definitions for contrastive explanations, but we will adhere to the definition as defined by Ignatiev et al. [19] i.e. a contrastive explanation of any classifier is a minimal set of input variables for which the observed value would have to change to alter the predicted class of the classifier. This definition can be used for any type of classifier, but in this thesis we will mostly consider Bayesian networks used as classifiers. The algorithms that are proposed in this chapter also work for different classifiers, as they only interact with the Bayesian network classifier through classification (from inference). The hardness proofs discussed in this section are specifically for Bayesian network classifiers, though that means it also works for the general problem given an unknown classifier, as that could be a Bayesian network classifier. Note that this definition of contrastive explanations does not specify which class the classifier should predict with the altered input, as long as it is a different class.

Definition 5.1.1. Contrastive explanation
Instance $\langle \mathcal{B}, (\boldsymbol{e}, \pi)\rangle$: A Bayesian network classifier $\mathcal{B}$ and a pair $(\boldsymbol{e}, \pi)$ where $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ and $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Contrastive explanation: Any set $\boldsymbol{S} \in \Omega(\boldsymbol{E})$ for which (1) $\boldsymbol{e} \backslash \boldsymbol{S} \nvDash_{\mathcal{B}} \pi$ and (2) for which no subset is a contrastive explanation: $\forall(\boldsymbol{S'} \subset \boldsymbol{S}) : \boldsymbol{e} \backslash \boldsymbol{S'} \vDash_{\mathcal{B}} \pi$.

We will illustrate the definition of a contrastive explanation using an example: consider Figure 5.1. In this figure we can see a Bayesian network consisting of three evidence nodes labeled "E1" through "E3", a single hypothesis node, and no other nodes. We use this Bayesian network as a classifier that predicts the class with the highest posterior probability. For example consider the case where $E1 = T, E2 = T, E3 = F$, in this case the predicted value for $\Pr(H = T|\boldsymbol{e})$ is higher than $\Pr(H = F|\boldsymbol{e})$ which means the predicted class is T. A contrastive explanation for the set $(\{E1 = T, E2 = T, E3 = F\}$,$T)$ would be a set of evidence variables which, when changed, could change the classification. In this situation the set $\{E3\}$ would suffice, as changing $E3$ would change the classification to F. The set $\{E2, E3\}$ would not be a valid contrastive explanation, as its subset is already a contrastive explanation. As another example, a contrastive explanation for $(\{E1 = F, E2 = F, E3 = F\}$,$T)$ consists of two variables, as changing a single variable does not change the outcome. The valid contrastive explanations in this case are $\{E1, E3\}$ and $\{E2, E3\}$. While $\{E1, E2, E3\}$ results in a different classification, two of its subsets are already contrastive explanations.



E1	E2	E3	P(H=T|E)
T	T	T	0.1
T	T	F	0.7
T	F	T	0.2
T	F	F	0.6
F	T	T	0.2
F	T	F	0.8
F	F	T	0.9
F	F	F	0.6

Figure 5.1: A simple Bayesian network with 3 evidence nodes, a single hypothesis node, and no other nodes.

With this definition, it is possible to distinguish multiple problems that might have different difficulties. We can:

1. Check whether a given input is a contrastive explanation of a given input-output pair.
2. Find any contrastive explanation for a given input-output pair.
3. Find the smallest possible contrastive explanation for a given input-output pair.
4. Calculate all contrastive explanations for a given input-output pair.
5. Calculate how many contrastive explanations there are for a given input-output pair.

The difficulty of these problems also depends on the information we have beforehand: do we know any inputs that result in another predicted class? Similarly, do we already know any abductive explanations (these will be introduced and explained in Section 6)? Known inputs that predict another class may limit the possible contrastive explanations we have to consider. In the next sections, we will discuss the abovementioned problems, possible ways to solve them, and their computational complexity.

It is of course also possible, and often more interesting, to consider a contrastive explanation for a specific alternative class, such as done by Koopman [21]: which inputs would have to change to give us class X instead of the currently predicted class? This is relevant, for instance, when a user wants to gain understanding about the workings of a classifier, and wonders why the classifier did not predict the class the user expected. In such a case we could slightly modify the classifier to use our earlier definition while still solving this problem: when the classifier predicts anything that is not class X, it will predict the original class instead. This way, we have a binary classifier that only classifies X or the original class. Sadly, this modification is not easy to implement using purely a Bayesian network. For example, let us assume we have a hypothesis node with classes a, b and c that can be predicted. Input 1 results in the prediction of probabilities 0.4 for class a and 0.3 for classes b and c. Input 2 results in a prediction of 0.1 for class a and b and 0.8 for class c. If we would want a contrastive explanation for input 2 that would result in class a, input 1 would be a valid option. We cannot simply modify the Bayesian network by adding the probabilities for all other classes, as this would predict c for input 1 instead of a. Because this change can easily be implemented outside the Bayesian network, all methods of finding contrastive explanations that purely use input-output pairs of the classifier can be used to get both types of contrastive explanations. However, without a way to implement this in a Bayesian network (which might be possible but is not trivial) the methods that specifically use Bayesian networks might not work. In short, we can slightly alter classifiers in such a way that our existing definition of a contrastive explanation corresponds to another existing definition of contrastive explanations which allows us to use the same methods to solve it. However, we cannot easily alter a Bayesian classifier in such a way while keeping it a Bayesian classifier, so some ways of finding contrastive explanations that will be introduced will not work.

5.2 Confirmation

Before discussing how to find contrastive explanations, let us first consider the confirmation of a contrastive explanation. To check whether something is a contrastive explanation we have to determine whether a given set of variables $\boldsymbol{S} \subseteq \underline{\boldsymbol{e}}$ is a contrastive explanation of a given input-output pair $(\boldsymbol{e}, \pi)$.

To determine this, two things have to be checked:

1. $\boldsymbol{e} \backslash \boldsymbol{S} \nvDash \pi$: there is an assignment $\boldsymbol{x} \in \Omega(\boldsymbol{S})$ that results in $\boldsymbol{x} \cup (\boldsymbol{e} \backslash \boldsymbol{S}) \vDash \neg\pi$.
2. No subset of $\boldsymbol{S}$ is a contrastive explanation.

5.2.1 Complexity

Even if the first condition above is met (a specific input $\boldsymbol{x}$ is given), the second condition is at least as hard as classification in a Bayesian network. In this case we have the following decision problem:

Definition 5.2.1. Confirmation decision problem
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi, \boldsymbol{x} \rangle$: A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$, and a set of variables $\boldsymbol{S} \subseteq \boldsymbol{E}$ with an assignment $\boldsymbol{x} \in \Omega(\boldsymbol{S})$ for which $\boldsymbol{x} \cup (\boldsymbol{e} \backslash \boldsymbol{S}) \vDash_{\mathcal{B}} \neg\pi$.
Question: Do all sets $\boldsymbol{S'} \subset \boldsymbol{S}$ predict $\boldsymbol{e} \backslash \boldsymbol{S'} \vDash_{\mathcal{B}} \pi$?

Note that this question is identical to the question whether $\boldsymbol{S}$ is a contrastive explanation of $(\boldsymbol{e}, \pi)$.

5.2.1.1 Bounded explanation size

It seems intuitive that when the explanation size grows, the explanation becomes harder to confirm. This is because the number of subsets grows exponentially. However, in this section we will prove that this problem is NP-hard even when we have a bound on the size of the explanation: by using a reduction to a contrastive confirmation decision problem with an explanation of constant size, we both show that the problem is NP-hard, and that it is not fixed-parameter tractable using its explanation size.

"And" and "Or" nodes Before explaining the proof we will introduce specific types of nodes that will be used in the proof and other proofs later in this thesis. We call these types of nodes "And" and "Or" nodes. These nodes act similarly to AND- and OR gates but are generalised to handle one non-Boolean variable. The "And" and "Or" nodes have conditional probability tables as described in Tables 5.1, 5.2, 5.3 and 5.4. The "And" and "Or" nodes use a variable state called s' and s respectively. This state will be given a value when the nodes are used in a proof. Given the following situation the nodes will function as the respective gates with the same name:

- The nodes have two binary parents
- s is set to TRUE, and s' is set to FALSE.

Under these circumstances, the "And" and "Or" node will have a state that is the AND or OR of the states of its parents respectively. For example, consider a scenario where we have an "And"-node with two independent parents of which the first parent $P1$ has a distribution of $\{\Pr(P1 = T) = 0.3, \Pr(P1 = F) = 0.7\}$ and the second parent $P2$ has a distribution of $\{\Pr(P2 = T) = 0.6, \Pr(P1 = F) = 0.4\}$. In this case the "And"-node will have a distribution of $\{\Pr(P2 = T) = 0.18, \Pr(P1 = F) = 0.82\}$. When both parent nodes only have a single distribution (a degenerate distribution), the nodes emulate AND or OR-gates even more: in this case the distribution of the "And" or "Or"-node will also be a degenerate distribution and will have the state that is the logical AND or OR of its parents with probability 1.

	V_b = X
$V_a = T$	X
$V_a = F$	s'

Table 5.1: The "And" node, a simplified truth table of a node with two parents V_x and V_y: V_x can take Boolean states and V_y can have any states. If V_x is TRUE, the state of this node is equal to the state of V_y with probability 1, if V_x is FALSE, the state of this node is state s' with probability 1. Note that this node works as an AND-gate if V_y has Boolean states and s' is set to FALSE.

$\Pr(V_{and} = v_{and} \| V_x \wedge V_y)$	$v_{and} = a$	$v_{and} = b$	$v_{and} = c$	$v_{and} = s'$
$V_x = T$, $V_y = a$	1	0	0	0
$V_x = T$, $V_y = b$	0	1	0	0
$V_x = T$, $V_y = c$	0	0	1	0
$V_x = F$, $V_y = a$	0	0	0	1
$V_x = F$, $V_y = b$	0	0	0	1
$V_x = F$, $V_y = c$	0	0	0	1

Table 5.2: An example of the conditional probability table of an "And" node where variable V_y can take states a, b and c, and s' is not equal to one of these states.

	V_b = X
$V_a = T$	s
$V_a = F$	X

Table 5.3: The "Or"-node, a simplified truth table of a node with two parents V_x and V_y: V_x can take Boolean states and V_y can have any states. If V_x is TRUE, the state of this node is equal to s with probability 1, if V_x is FALSE, the state of this node is equal to the state of V_y with probability 1. Note that this node works as an OR-gate if V_y has Boolean states and s is set to TRUE.

Proof Using these "And" and "Or"-nodes, we will now show that confirming a contrastive explanation is NP-hard.

Theorem 2. The contrastive confirmation decision problem (Definition 5.2.1) with bounded $|\boldsymbol{S}|$ is at least as hard as the Bayesian network classification decision problem (Definition 4.2.1) and therefore NP-hard.

We can provide insight into the complexity of this problem using a reduction from the classification problem as defined in 4.2.1: we will show that for any classification in a Bayesian network classifier $\mathcal{B}$, we can create another Bayesian network classifier $\mathcal{B}'$, such that by finding a contrastive explanation for $\mathcal{B}'$ we can find any classification for $\mathcal{B}$. This would mean that finding contrastive explanations is at least as hard as classification as defined in Definition 4.2.1, since we can do classification by finding a contrastive explanation. We will now give an example of how to transform any Bayesian network classifier $\mathcal{B}$ to the correct $\mathcal{B}'$.

$\Pr(V_{or} = v_{or} \| V_a \wedge V_b)$	$v_{or} = a$	$v_{or} = b$	$v_{or} = c$	$v_{or} = s$
$V_x = T$, $V_y = a$	0	0	0	1
$V_x = T$, $V_y = b$	0	0	0	1
$V_x = T$, $V_y = c$	0	0	0	1
$V_x = F$, $V_y = a$	1	0	0	0
$V_x = F$, $V_y = b$	0	1	0	0
$V_x = F$, $V_y = c$	0	0	1	0

Table 5.4: An example of the conditional probability table of an "Or"-node where variable V_y can take states a b and c, and s is not equal to one of these states

Proof.

Transformation: Suppose we are given an instance $(\mathcal{B}, \boldsymbol{e}_1, V_h, \pi)$ of the Bayesian network classification decision problem. We now build an instance $\langle \mathcal{B}', \boldsymbol{e}_2, \pi, \boldsymbol{x} \rangle$ of the contrastive confirmation decision problem as follows: we transform $\mathcal{B}$ to $\mathcal{B}'$ which corresponds to Figure 5.2: we add two evidence nodes, A and B, and nodes labeled "And" and "Or" to $\mathcal{B}$. These nodes act as the "And" and "Or"-nodes described above. The "And"-node takes as input the Boolean node A and the BN hypothesis node V_h. It "returns" with probability 1 one of the values in $\Omega(V_h) \cup \{s'\}$, where $\Omega(V_h) \cap \{s'\} = \emptyset$. This set of values is then the possible input to the "Or"-node, together with the Boolean node V_B. The "Or"-node "returns" with probability 1 one of the values in $\Omega(V_h) \cup \{s, s'\}$. s in the CPT of the "And"-node will be chosen to be π while s' can have any arbitrary value that is not π. We define the evidence from the contrastive confirmation decision problem to be $\boldsymbol{e}_2 = \boldsymbol{e}_1 \cup \{A = T, B = T\})$. The hypothetical contrastive explanation we want to confirm is $\boldsymbol{S} = \{A, B\}$ and $\boldsymbol{x} = \{A = F, B = F\}$. Note that $\boldsymbol{e}_2 \vDash \pi$ and $\{A = F, B = F\} \cup \boldsymbol{e}_1 \vDash s'$.

polynomial: We add four nodes with a polynomial CPT, making this reduction polynomial.

correctness:

Lemma 2.1. If $\langle \mathcal{B}, \boldsymbol{e}_1, V_h, \pi \rangle$ is a positive instance of the classification problem, $\langle \mathcal{B}', \boldsymbol{e}_2, \pi, \boldsymbol{x} \rangle$ is a positive instance of the contrastive confirmation decision problem.

I.e. all sets $\boldsymbol{S'} \subset \{A, B\}$ predict $\boldsymbol{e}_2 \backslash \boldsymbol{S'} \vDash \pi$ in the modified network.

Proof. $\{A, B\}$ has three subsets:

1. $\boldsymbol{S'} = \{\}$: $(A = T)$, $(B = T)$. In this case we have already seen that $\boldsymbol{e}_2 \vDash s = \pi$
2. $\boldsymbol{S'} = \{B\}$: case 1 or $(A = T)$, $(B = F)$. In this case the "Or"-node still predicts $s = \pi$ by definition.
3. $\boldsymbol{S'} = \{A\}$: case 1 or $(A = F)$, $(B = T)$. In this case the "Or"-node takes the value of "And" and "And" takes the value of the original Bayesian network. Since the original Bayesian network predicts π, the "Or"-node will predict π.

■

Lemma 2.2. If $\langle \mathcal{B}', \boldsymbol{e}_2, \pi, \boldsymbol{x} \rangle$ is a positive instance of the contrastive confirmation decision problem, $\langle \mathcal{B}, \boldsymbol{e}_1, V_h, \pi \rangle$ is a positive instance of the classification problem.

I.e. if for all sets $\boldsymbol{S'} \subset \{A, B\}$, $\boldsymbol{e}_2 \backslash \boldsymbol{S'} \vDash \pi$ is predicted in the modified network, $\boldsymbol{e}_1 \vDash \pi$ in the original network.

Proof. For $\boldsymbol{S'} = \{A\}$, $\boldsymbol{e}_2 \backslash \boldsymbol{S'} \vDash \pi$ implies $\boldsymbol{e}_1 \vDash \pi$: we know that $\boldsymbol{e}_1 \cup \{(A = F)\ (B = T)\}$ predicts π, and in this case the "Or"-node has the same probability $\Pr(OR = \pi | \boldsymbol{e}_2 \backslash A \cup (A = F))$ as the "And"-node $\Pr(AND = \pi | \boldsymbol{e}_2 \backslash A \cup (A = F))$ which has the same distribution as the hypothesis node of the original network $\Pr(V_h = \pi | \boldsymbol{e}_2 \backslash A \cup (A = F)) = \Pr(V_h = \pi | \boldsymbol{e}_1)$. The only way for $\mathcal{B}'$ to predict π is if $\mathcal{B}$ predicts π given evidence $\boldsymbol{e}_1$. ■

From the above, we can see that there is a polynomial transformation from the classification decision problem to the contrastive confirmation decision problem. As classification is NP-hard [6], it follows that confirmation is NP-hard. ☐

5.2.2 A naive confirmation algorithm

We can create a simple naive algorithm that checks whether $\boldsymbol{S}$ is a contrastive explanation. This algorithm has a running time of $O(|\boldsymbol{S}|^2 \cdot I \cdot S^{(|\boldsymbol{S}|)})$. Note that S here refers to the maximum number of states a variable can have, while $\boldsymbol{S}$ is a set of variables. This gives us an upper bound on the complexity of the problem, since it is at least solvable in that amount of time. The algorithm simply checks both conditions: an assignment of the set $\boldsymbol{S}$ results in another prediction than π, and no subset of $\boldsymbol{S}$ is a contrastive explanation. The function *Classify* gives the classification of the Bayesian network classifier when given evidence $\boldsymbol{e}$.

Algorithm 1 Contr_check($\boldsymbol{S}$)

```
$\pi = \mathit{Classify}(\boldsymbol{e})$
if $\forall \boldsymbol{x} \in \Omega(\boldsymbol{S}; \boldsymbol{e})(\mathit{classify}(\boldsymbol{x} \cup (\boldsymbol{e} \backslash \boldsymbol{S})) == \pi)$ then
  return FALSE
end if
for $\boldsymbol{S'} \subset \boldsymbol{S}$ do
  for $\boldsymbol{x} \in \Omega(\boldsymbol{S'}; \boldsymbol{e})$ do
    if $\neg \mathit{classify}(\boldsymbol{x} \cup (\boldsymbol{e} \backslash \boldsymbol{S'})) == \pi$ then
      return FALSE
    end if
  end for
end for
return TRUE
```

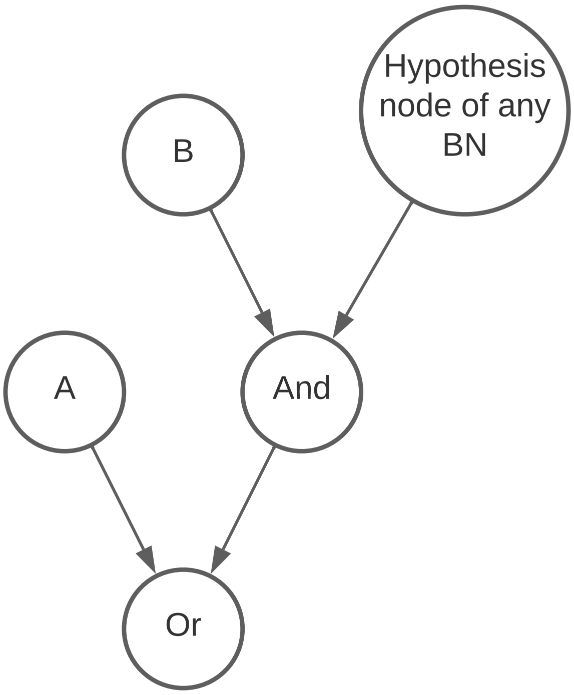


Figure 5.2: A visualization of the transformation in the proof of Theorem 2

5.2.3 Confirmation with known abductive explanations

Ignatiev et al. [19] have shown that, when all abductive explanations $\mathcal{A}$ are known, every contrastive explanation is a minimal hitting set of $\mathcal{A}$.

Definition 5.2.2. Hitting set

Instance $\langle \mathcal{S} \rangle$: A set of sets $\mathcal{S}$
Hitting set: A hitting set m of $\mathcal{S}$ is a set that contains at least one of the elements of every set in $\mathcal{S}$. $\forall s \in \mathcal{S} : m \cap s \neq \emptyset$.

A minimal hitting set (MHS) is simply a hitting set of which no subset is a hitting set.

As an example, consider $\mathcal{S} = \{\{A, B, C\}, \{A, C, D\}, \{C, E\}\}$. Possible minimal hitting sets are now: $\{C\}$, $\{A, E\}$ and $\{B, D, E\}$. Sets such as $\{A, C\}$ are a hitting set but not an MHS, as a subset ($\{C\}$) is already a hitting set.

This means that if all abductive explanations are known, we can easily confirm a contrastive explanation: we simply check if removing any variable from a contrastive explanation $\boldsymbol{S}$ would no longer make it a hitting set for $\mathcal{A}$. For every variable in $\boldsymbol{S}$, we check if every set $\boldsymbol{A} \in \mathcal{A}$ is still hit when it is removed. Confirming a contrastive explanation can be done this way in $O(|\boldsymbol{E}| \cdot |\mathcal{A}| \cdot |\boldsymbol{S}|^2)$ time, where $|\boldsymbol{E}| \cdot |\boldsymbol{S}|$ is the time it takes to trivially check whether $\boldsymbol{S}$ without one variable hits another set (as all sets in $\mathcal{A}$ have at most $|\boldsymbol{E}|$ variables).

5.3 Finding any contrastive explanation

Another problem is finding a contrastive explanation. This problem is defined in Definition 5.3.1.

Definition 5.3.1. Any contrastive explanation
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$: A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: any set $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $\boldsymbol{e} \backslash \boldsymbol{S} \nvDash_{\mathcal{B}} \pi$ and (2) for which no subset is a contrastive explanation: $\forall(\boldsymbol{S'} \subset \boldsymbol{S}) : \boldsymbol{e} \backslash \boldsymbol{S'} \vDash_{\mathcal{B}} \pi$.

5.3.1 Complexity

A prediction has a contrastive explanation if and only if there is any input for which the classifier predicts another class. While most useful classifiers are able to predict multiple classes, one can trivially create a Bayesian network for which only one hypothesis will ever get the highest probability. We will show that even determining whether any contrastive explanation exists is not always possible in a polynomial number of classifications (assuming $P \neq NP$). This means there is no hope of always finding a contrastive explanation in polynomial time, as finding one would also tell us whether one exists. We can show this by proving that it is NP-complete to determine whether there exists a classification with a class other than some specified class π even when classification can be done in polynomial time. We will give a reduction from CNFSAT (satisfiability where the formula is in CNF) that gives us an instance of the multiple classification decision problem as defined in Definition 5.3.3 for which a yes-instance corresponds to a yes-instance of satisfiability.

Definition 5.3.2. CNFSAT **Instance $\langle \mathcal{F} \rangle$:** A formula $\mathcal{F}$ in conjunctive normal form using the variables $\boldsymbol{F}$.
Question: Is there an assignment $\boldsymbol{f} \in \Omega(\boldsymbol{F})$ such that $\mathcal{F}$ evaluates to TRUE?

Definition 5.3.3. Alternative classifications decision problem **Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Question: Is there a different set of evidence $\boldsymbol{a} \in \Omega(\boldsymbol{E})$ for which $\boldsymbol{a} \vDash_{\mathcal{B}} \neg\pi$?

Theorem 3. The alternative classifications decision problem for Bayesian networks on which inference takes polynomial time is NP-complete.

Proof.

Lemma 3.1. The alternative classifications decision problem for Bayesian networks on which inference takes polynomial time is NP-hard.

Lemma 3.2. The alternative classifications decision problem for Bayesian networks on which inference takes polynomial time is in NP. I.e. a solution to the problem can be verified in polynomial time.

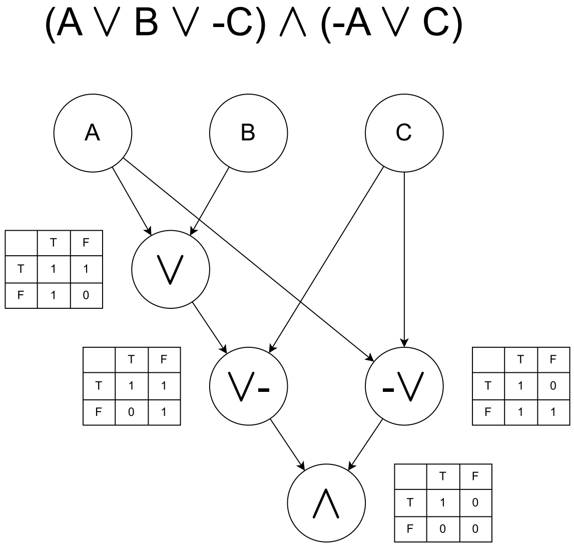
□

The proof for Lemma 3.1:

Proof.

Transformation: Suppose we are given an instance of CNFSAT with logical formula $\mathcal{F}$ containing variables $\boldsymbol{E}$. We now build an instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$ of the alternative classifications decision problem: as we have seen in paragraph 5.2.1.1, we can create BN-nodes that act similarly to AND- or OR gates. If we only use Booleans, probabilities of 1 or 0, define s' as FALSE and s as TRUE, and only use degenerate distributions, they function exactly the same. This means that we can also create a network that represents any logical formula containing only AND and OR. If we (partially) invert the probability table, we can also use negations. Using such a network, we can represent $\mathcal{F}$ and $\boldsymbol{E}$ as a Bayesian network $\mathcal{B} = (G, Pr)$, as pictured in Figure 5.3 for an example formula. The network has the following properties:

1. Each variable in $\boldsymbol{E}$ is represented as a binary root node in G.
2. All binary logical operators in $\mathcal{F}$ are represented by nodes that have two parents, capturing the two logical expressions the operator takes as input. The negation of an input is included in the node for the binary operator.
3. The hypothesis node is the leaf node representing the "outermost" logical operator: the operator that is used last when computing the value of the formula.



	T	F
T	1	1
F	1	0

	T	F
T	1	1
F	0	1

	T	F
T	1	0
F	1	1

	T	F
T	1	0
F	0	0

Figure 5.3: A Bayesian classifier representing a logical formula

As all nodes in $\boldsymbol{E}$ are evidence nodes, classification can be done very efficiently in this Bayesian network: by applying evidence absorption [42], we can, in polynomial time, remove the outgoing arrows from the evidence nodes by including the evidence in the CPTs. We now have a simple singly connected graph which allows us to calculate inference in polynomial time, using for example Pearls algorithm.

This way, we can create a Bayesian network classifier corresponding to any CNFSAT instance that can classify in polynomial time. We can easily create an assignment $\boldsymbol{e}$ where the class FALSE is predicted by using evidence where one of the CNFSAT clauses is FALSE. This can be done assigning the inverse of every literal in the clause to the variables (i.e. $(\neg A, \neg B, C)$ for the example in Figure 5.3). A situation where evidence $\boldsymbol{a}$ makes the prediction TRUE, however, would correspond to a solution of the CNFSAT problem, which is NP-complete. A contrastive explanation to any assignment where $\pi = $ FALSE would predict TRUE and prove a yes-instance of SAT. This means if we can determine whether there is a contrastive explanation using a polynomial number of inferences, we can solve CNFSAT in polynomial time.

polynomial: To convert the CNFSAT instance to a Bayesian network we add exactly one node for every variable and one node for every $\wedge$ and $\vee$. The number of parents is at most two, bounding the size of the probability tables. This means the reduction can be done in polynomial time.

correctness:

Lemma 3.3. If $\langle \mathcal{F} \rangle$ is a positive instance of CNFSAT, $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$ is a positive instance of the alternative classifications decision problem.

Proof. Because the internal nodes function exactly the same as logical operators, the Bayesian network predicts TRUE with a probability of 1 for any $\boldsymbol{a}$ that satisfies $\mathcal{F}$. This answers the alternative classifications decision problem: we now know there exists a set of evidence $\boldsymbol{a}$ that results in a different prediction than $\pi = $ FALSE, so there exists an alternative classification. ■

Lemma 3.4. If $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$ is a positive instance of the alternative classifications decision problem, $\langle \mathcal{F} \rangle$ is a positive instance of SAT.

Proof. A positive instance of the alternative classifications decision problem predicts $\neg\pi = T$ for Bayesian network. Because the internal nodes function exactly the same as logical operators, a prediction of TRUE means the evidence $\boldsymbol{a}$ evaluates $\mathcal{F}$ to TRUE and solves this instance of SAT. ■

From the above, we can see that there is a polynomial transformation from CNFSAT decision problem to the alternative classifications decision problem. As CNFSAT is NP-hard, it follows that the alternative classifications decision problem and therefore the any contrastive explanation problem is NP-hard. □

The proof for Lemma 3.2:

Proof. By definition, classification can in this case be done in polynomial time. By defining our certificate as an evidence assignment $\boldsymbol{a} \in \Omega(\boldsymbol{E})$ for which $\boldsymbol{a} \vDash_{\mathcal{B}} \neg\pi$, we can confirm this in polynomial time using inference. □

5.3.2 Algorithm

Again we will give a naive algorithm that finds a contrastive explanation for any classifier by applying inference a lot of times. Note that such an approach is guaranteed to take an exponential number of inferences:

Theorem 4. It is not possible to create an algorithm that finds a contrastive explanation with a worst-case time that is less than exponential while only interacting with the Bayesian network through inference.

Proof. Since a Bayesian network can represent any distribution, it might be possible that only a single combination of evidence predicts another class, which can be any arbitrary combination of evidence (for example by using the "And"-nodes discussed earlier). This means the only combination of evidence that predicts another class might be the last classification we do regardless of the order in which we do inference, so we know that worst-case we need to do inference for all combinations of evidence to find our contrastive explanation. □

Algorithm 2 Potential_contrastive($\boldsymbol{c}$)

```
for $V_v \in (\underline{\boldsymbol{c}})$ do
    if $classify((\boldsymbol{c} \backslash V_v) \cup (\boldsymbol{a} \backslash (\underline{\boldsymbol{c} \backslash V_v}))) \neq \pi$ then
        return $Potential\_contrastive(\boldsymbol{c} \backslash V_v)$
    end if
end for
return $\underline{\boldsymbol{c}}$
```

To make things slightly less trivial, we also consider the situation where we know a classification $(\boldsymbol{a}, \neg\pi)$ where $\boldsymbol{a} \in \Omega(\boldsymbol{E})$ that predicts another class. In this case we will first compute a potential contrastive explanation in a polynomial number of inferences using Algorithm 2. Assume $\boldsymbol{c}$ initially contains all evidence assignments in $\boldsymbol{a}$ that are different in $\boldsymbol{e}$.

This does not always result in a valid contrastive explanation, just a prediction for which you cannot change *one* more variable towards the original prediction while still getting another predicted class. If any combination of evidence for which a subset of $\underline{\boldsymbol{c}}$ is different to $\boldsymbol{e}$ predicts anything other than π, this is not a contrastive explanation.

However, we now have a smaller subset of variables we have to consider to get at least one contrastive explanation. If $\underline{\boldsymbol{c}}$ is not a contrastive explanation, there must be at least one contrastive explanation as a subset of $\underline{\boldsymbol{c}}$ since $\underline{\boldsymbol{c}}$ can have an assignment $\boldsymbol{c}$ that predicts another class, which means it is only not a contrastive explanation if a subset of $\underline{\boldsymbol{c}}$ is a contrastive explanation.

We can now determine all possible combinations of values for $\boldsymbol{S} = \underline{\boldsymbol{c}}$ using Algorithm 3 starting from the smallest subsets to find out which is the smallest number of changes that results in a different prediction.

Algorithm 3 Any_contrastive($\boldsymbol{S}$)

```
for $\boldsymbol{s} \in \Omega(\boldsymbol{S})$ do
    if $classify(\boldsymbol{s}) \neq \pi$ then
        return $\underline{\boldsymbol{s}}$
    end if
end for
return Error: No contrastive explanation exists
```

Note that the order in which the assignments to evidence are considered matters: the algorithm only works if we consider any assignment that has variables $\boldsymbol{V}$ that are different than in $\boldsymbol{e}$ before all assignments for which a superset $\boldsymbol{V'} \supset \boldsymbol{V}$ are different than in $\boldsymbol{e}$. For clarity of the algorithm this is not shown to be enforced in the pseudocode. The reason this is necessary is that then the second requirement for contrastive explanations is enforced by the order: if a set is a contrastive explanation, it would have terminated the algorithm before testing its superset. Algorithm 4 is identical but includes a way to enforce this order.

The algorithm creates all possible subsets of $\boldsymbol{E}$, and tries all combinations where the variables in the subset have a different assignment than in $\boldsymbol{e}$. These subsets are created by passing over all variables in $\boldsymbol{E}$, and adding them to all subsets, starting with the empty set. When the algorithm creates a subset $\boldsymbol{S}$, a specific node V_n has just been added to the subset. $\boldsymbol{S}$ has, by definition, been created after $\boldsymbol{S} \backslash V_n$. $\boldsymbol{S}$ has also been created after $\boldsymbol{S} \backslash V_{n'}$ for any $V_{n'}$: V_n has also been added to $\boldsymbol{S} \backslash V_{n'}$, but $\boldsymbol{S} \backslash (V_n \wedge V_{n'})$ is earlier in the list than $\boldsymbol{S} \backslash V_n$, because when $V_{n'}$ was added, the version without $V_{n'}$ already existed. This means the order in which the subsets are created guarantees that an assignment to evidence is tested before all assignments where a superset of evidence is different than in $\boldsymbol{e}$.

Algorithm 3/4 runs in $O(I \cdot S^{|\boldsymbol{S}|})$ time, where S again refers to the maximum number of states a variable can have. We have seen that, if are unable to interact with a classifier except through classification (i.e. we cannot use any information about CPTs or the structure of the network directly) and we did not have a potential contrastive explanation, this is an optimal algorithm. This is due to the fact that worst-case, it is always necessary to try every combination of evidence, as we saw in Theorem 4.

Algorithm 4 Any_contrastive($\boldsymbol{S}$)

```
List<Set> sets = $\{\emptyset\}$
for inputItem $\in \boldsymbol{E}$ do
    for set $\in$ sets do
        newSet = set $\cup$ inputItem
        for $\boldsymbol{s} \in \Omega(\text{newSet}; \boldsymbol{e})$ do
            if $((\boldsymbol{e} \backslash \text{newSet}) \cup \boldsymbol{s}) \vDash \neg\pi$ then
                return newSet
            end if
        end for
        temp.add(newSet)
    end for
end for
```

If we did have a potential contrastive explanation however, the time required may be much less: every time we were able to remove a state from $\boldsymbol{S}$, the worst-case time to find a contrastive explanation gets divided by the number of states this node can have.

5.3.3 Known abductive explantions

Just like we saw in Section 5.2.3, we can use the minimal hitting set relationship to easily solve the problem from definition 5.3.1 when all abductive explanations $\mathcal{A}$ are known: given a set $\boldsymbol{V}$ that is initially equal to $\boldsymbol{E}$, for each node $V_x \in \boldsymbol{E}$ we can check if $\boldsymbol{V} \backslash V_x$ still hits all sets in $\mathcal{A}$. If it does, we can remove V_x from $\boldsymbol{V}$ and continue. We only have to make a single pass over the variables, as it will never be possible to remove a variable from V_v that we could not remove before. This can easily be done with a running time of $O(|\mathcal{A}| \cdot |\boldsymbol{E}|^3)$ where $|\boldsymbol{E}|^2$ is the time it takes to trivially check whether any set hits another set (as all sets have at most $|\boldsymbol{E}|$ variables).

5.4 Other problems about finding contrastive explanations

As discussed earlier in Section 5.1, we can distinguish many other problems that deal with finding contrastive explanations. In this section we will discuss those and what we know about their complexity.

5.4.1 Finding the smallest contrastive explanation

While something can only be a contrastive explanation if no subsets are a contrastive explanations, there can still be multiple contrastive explanations of different sizes. A smaller contrastive explanation may be easier to interpret for the user, and therefore more suitable as an explanation. We can define the problem as in Definition 5.4.1

Definition 5.4.1. Smallest contrastive explanation **Instance** $\langle \mathcal{B}, \boldsymbol{e}\pi \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: any set $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $\boldsymbol{e} \backslash \boldsymbol{S} \nvDash_{\mathcal{B}} \pi$ and (2) for which no smaller set is a contrastive explanation: $\forall \boldsymbol{S'} \subseteq \boldsymbol{E} : |\boldsymbol{S'}| < |\boldsymbol{S}| \implies \boldsymbol{e} \backslash \boldsymbol{S'} \vDash_{\mathcal{B}} \pi$.

Trivially, this problem is at least as hard as the problem of finding any contrastive explanation (Definition 5.3.1), as the smallest contrastive explanation is by definition a contrastive explanation and therefore a solution to that problem. This means the problem is NP-hard even when inference can be performed in polynomial time.

5.4.2 Finding the number of contrastive explanations

We can also define the problem of finding the number of contrastive explanations:

Definition 5.4.2. Number of contrastive explanations **Instance** $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: The number of sets $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $\boldsymbol{e} \backslash \boldsymbol{S} \nvDash_{\mathcal{B}} \pi$ and (2) for which no subset is a contrastive explanation: $\forall \boldsymbol{S'} \subset \boldsymbol{S} : \boldsymbol{e} \backslash \boldsymbol{S'} \vDash_{\mathcal{B}} \pi$.

Again, we already have a more general problem for which we have a lower bound on the complexity: finding the number of contrastive explanations immediately tells us whether any contrastive explanations exist, and as we have seen this is equivalent to the question whether any other class than π can be predicted using the Bayesian network. Trivially, this means finding the number of contrastive explanations (Definition 5.4.2) is at least as hard as finding out whether any other prediction than π can occur (Definition 5.3.3) and is therefore NP-hard even when inference can be done in polynomial time.

5.4.3 Finding all contrastive explanations

We can also define the problem of finding all contrastive explanations:

Definition 5.4.3. All contrastive explanations **Instance** $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: All sets $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $\boldsymbol{e} \backslash \boldsymbol{S} \nvDash_{\mathcal{B}} \pi$ and (2) for which no subset is a contrastive explanation: $\forall \boldsymbol{S'} \subset \boldsymbol{S} : \boldsymbol{e} \backslash \boldsymbol{S'} \vDash_{\mathcal{B}} \pi$.

Again, obtaining the answer to this problem would immediately solve the problem of finding out whether any other prediction can occur as defined in Definition 5.3.3. This means the problem is again NP-hard even when inference can be done in polynomial time.

As solving this problem immediately answers all problems related to contrastive explanations before this point, we will provide a trivial algorithm that serves as an upper bound for the time-complexity of any problems for which we have not yet provided an algorithm. The pseudocode can be found in Algorithm 5. The majority of the code makes sure no set is tested twice, and that sets only get tested for a contrastive explanation when their subsets have already been tested and were not a contrastive explanation. This means we only have to test for evidence where all variables in the contrastive explanation get assigned a different variable than in $\boldsymbol{E}$. This is because all combinations of evidence where at least one of those variables has an assignment in $\boldsymbol{E}$ has already been tested and must have predicted π. Line 17 determines whether a configuration of the potential contrastive explanation $\boldsymbol{S}$ predicts another class. The algorithm takes $O(I \cdot S^{|\boldsymbol{E}|})$ time: worst-case it visits all possible configurations of evidence, and classifies once for all of them.

The order of creating the subsets is the same as in the algorithm in Section 5.3, however, this time we have to make sure that we do not consider sets for which a subset has already predicted another class.

Smarter algorithms have been proposed by Koopman [21] and Ignatiev et al. [19]. The algorithm by Koopman is effectively an improvement to the algorithm that classifies a lot of times, while the algorithm by Ignatiev et al. is able to convert some Bayesian networks to a more efficient representation and find explanations from there.

Chapter 7 also introduces an algorithm that utilises the structure of a Bayesian network to reduce the computations required.

Algorithm 5 All_contrastive($\boldsymbol{E}$)

```
List<Set> sets = $\{\emptyset\}$
List<Set> retSets = $\emptyset$
for inputItem $\in \boldsymbol{E}$ do
    for set $\in$ sets do
        any = FALSE
        newSet = set $\cup$ inputItem
        for var $\in$ set do
            if newSet$\setminus$var$\notin$sets then
                any = TRUE
                break
            end if
        end for
        if any then
            continue
        end if
        for $\boldsymbol{s} \in \Omega(\text{newSet}; \boldsymbol{e})$ do
            if $((\boldsymbol{e}\setminus\text{newSet}) \cup \boldsymbol{s}) \vDash \neg\pi$ then
                retSets.add(newSet)
                any = TRUE
                found = TRUE
                break
            end if
        end for
        if $\neg$any then
            sets.add(newSet)
        end if
    end for
end for
return retSets
```

5.5 An overview of complexities

In this section we will give an overview of the discussed complexities and point out which complexities have not been discussed and might be interesting for further research. Consider Table 5.5. Note that a cell in column A always represents a problem that is at least as hard as its neighbour in column B. This is because column B has strictly more information.

No information For cell A1 and A2 the NP-hardness has been proven in Sections 5.2 and 5.3 respectively. A3-A5 are at least as hard as one of these problems as discussed in Section 5.4.

Counterexample Column B is added as we have a better known algorithm for confirming or finding a contrastive explanation when a counterexample is known. There might be more efficient algorithms for some of the other problems as well, but the complexities shown in the table are just a bound on the runtime of one of the simple algorithms we discussed.

All abductive explanations known C5 has been shown to be solvable by finding all MHSs of all abductive explanations [19]. The method Ignatiev et al. proposed to find all contrastive explanations is likely faster than the naive algorithms we discussed before, as it does not depend on inference. This means it is probably a better way to solve C3-C5. The runtime of this algorithm was not explicitly mentioned and may depend on the fastest known algorithms to solve the MHS or vertex cover problem. C1 and C2 can be solved in polynomial time and have been briefly discussed at the end of their respective sections.

	A: Nothing	B: Counterexample $\boldsymbol{C}$	C: All abductive explanations $\mathcal{A}$
1. confirmation of $\boldsymbol{S}$ (Def 5.2.1)	🟧 $O(\lvert\boldsymbol{S}\rvert^2\cdot I\cdot S^{(\lvert\boldsymbol{S}\rvert)})$	$O(\lvert\boldsymbol{S}\rvert^2\cdot I\cdot S^{(\lvert\boldsymbol{S}\rvert)})$	🟩 $O(\lvert\mathcal{A}\rvert\cdot\lvert\boldsymbol{E}\rvert\cdot\lvert\boldsymbol{S}\rvert)$
2. any (Def 5.3.1)	🟥 $O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$	$O(I\cdot S^{\lvert\boldsymbol{C}\rvert})$	🟩 $O(\lvert\mathcal{A}\rvert\cdot\lvert\boldsymbol{E}^2\rvert)$
3. smallest (Def 5.4.1)	🟥 $O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$	$O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$	
4. number (Def 5.4.2)	🟧 $O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$	$O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$	
5. all (Def 5.4.3)	🟥 $O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$, [21]	$O(I\cdot S^{\lvert\boldsymbol{E}\rvert})$	[19]

Table 5.5: An overview of discussed complexities: the rows represent the problem definitions and the columns represent the known information. The cell colours indicate a known NP-hardness proof, while the text contains the worst-case runtime of the discussed algorithms. An orange cell indicates that the problem is NP-hard, a red cell indicates that the problem is NP-hard even when inference is guaranteed to be possible in polynomial time. A green cell indicates that a polynomial time algorithm is known. White means no NP-hardness was discussed for this problem. The rows and columns are named to easily refer to individual cells.

5.6 Discussion

Ignatiev et al. [19] have shown that finding all contrastive explanations given all abductive explanations (see next chapter) can be solved by determining all MHSs. If these problems are equivalent, the known NP-hardness of MHS problems gives us additional information about the NP-hardness of some of the discussed problems. Specifically, we then know that calculating the smallest abductive explanation and calculating all explanations are still NP-hard when given all abductive explanations. This would colour B3, B5, C3 and C5 orange in Table 5.5. However, we would have to prove that the problems are equivalent, which we decided not to include in this thesis.

The runtime of the polynomial algorithms discussed in 5.2.3 and 5.3.3 can be significantly reduced by using a hashset or similar datastructure to store the hypothetical contrastive explanation, and more efficiently calculate the intersections with the known abductive explanations. We expect that this would remove an $|\boldsymbol{E}|$ term from the running time for both algorithms.

Chapter 6

Abductive explanations

In this chapter, we will discuss what abductive explanations are. We will also distinguish between different problems, just as we did for contrastive explanations, and discuss the complexity of these problems. Let us first define abductive explanations:

Definition 6.0.1. Abductive explanation
Instance $\langle \mathcal{B}, (\boldsymbol{e}, \pi)\rangle$: A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, and a pair $(\boldsymbol{e}, \pi)$ where $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ and $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$,
Abductive explanation: Any set $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $(\boldsymbol{S} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$ and (2) no subset is an abductive explanation: $\forall(\boldsymbol{S'} \subset \boldsymbol{S}) : \boldsymbol{S'} \in \boldsymbol{e} \nvDash_{\mathcal{B}} \pi$.

This means that an abductive explanation is a minimal subset of the evidence nodes for which the current assignments guarantee the observed prediction π, regardless of the assignment to the other evidence variables.

As an example, consider Figure 5.1 again. Just like for the contrastive example, we will take a look at the prediction with evidence-classification pair $(\{E1 = T, E2 = T, E3 = F\}, T)$. $\{E3\}$ would be a valid abductive explanation, as all sets where $E3 = F$ predict True. In this case, this is the only abductive explanation. Also note how the fact that this is the only abductive explanation is an example of the MHS (see Section 5.2.3) relationship between the contrastive and abductive explanations: as seen in Chapter 5, the only contrastive explanation in this case is also $\{E3\}$, which is the only MHS of itself.

Consider the other example from Chapter 5. Given the evidence-classification pair $(\{E1 = F, E2 = F, E3 = F\}, T)$, we have seen that the contrastive explanations are $\{E1, E3\}$ and $\{E2, E3\}$. Again, $\{E3\}$ is an abductive explanation, as all sets where $E3 = F$ predict True. This time, however, $\{E1, E2\}$ is also an abductive explanation, as all combinations of evidence where $E1 = F$ and $E2 = F$ predict True, while not all sets where $E1 = F$ do, and not all sets where $E2 = F$ predict True either. Again, we can also look at the MHS relations:

- $\{E3\}$ is an MHS of $\{\{E1, E3\}, \{E2, E3\}\}$
- $\{E1, E2\}$ is an MHS of $\{\{E1, E3\}, \{E2, E3\}\}$
- $\{E1, E3\}$ is an MHS of $\{\{E3\}, \{E1, E2\}\}$
- $\{E2, E3\}$ is an MHS of $\{\{E3\}, \{E1, E2\}\}$

There do not exist any other MHSs of $\{\{E1, E3\}, \{E2, E3\}\}$ or $\{\{E3\}, \{E1, E2\}\}$.

6.1 Defining the problems

We can mostly make the same distinctions as with contrastive explanations: we can distinguish the problem of verifying an abductive explanation, finding any abductive explanation, finding the smallest abductive explanation, finding the number of abductive explanations, and finding all abductive explanations. Just like for contrastive explanations, these definitions and the algorithms in this chapter are model-agnostic and can be used for any classifier, though they are only discussed for Bayesian network classifiers.

Again, we can also distinguish what other information we have: if we know all contrastive explanations, the problem might become easier due to the MHS relation.

Below we will define the problems we will discuss for abductive explanations. For the abductive confirmation decision problem, we assume that we already know that the potential abductive explanation predicts π, as otherwise, it would also include the classification problem. The hardness we will show for this problem also holds if we do not yet know whether the potential abductive explanation predicts π, while that would not work the other way around. The abductive confirmation decision problem only deals with the minimality constraint of the potential abductive explanation.

Definition 6.1.1. Abductive confirmation decision problem
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi, \boldsymbol{x} \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$, a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$, and a set of variables $\boldsymbol{S} \subseteq \boldsymbol{E}$ with corresponding assignment $\boldsymbol{x} = \boldsymbol{S} \in \boldsymbol{e}$ for which $\boldsymbol{x} \vDash_{\mathcal{B}} \pi$.
Question: Do no sets $\boldsymbol{S'} \subset \boldsymbol{S}$ exist such that $(\boldsymbol{S'} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$?

Note that this question is identical to the question of whether $\boldsymbol{S}$ is an abductive explanation of $(\boldsymbol{e}, \pi)$.

Definition 6.1.2. Any abductive explanation
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$ **:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$, and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: Any set $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $(\boldsymbol{S} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$ and (2) for which no subset is an abductive explanation: $\forall (\boldsymbol{S'} \subset \boldsymbol{S}) : (\boldsymbol{S'} \in \boldsymbol{e}) \nvDash_{\mathcal{B}} \pi$.

Definition 6.1.3. Smallest abductive explanation
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: any set $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $(\boldsymbol{S} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$ and (2) for which no smaller set is an abductive explanation: $\forall \boldsymbol{S'} \subset \boldsymbol{E} : |\boldsymbol{S'}| < |\boldsymbol{S}| \implies (\boldsymbol{S'} \in \boldsymbol{e}) \nvDash_{\mathcal{B}} \pi$.

Definition 6.1.4. Number of abductive explanations
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: The number of sets $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $(\boldsymbol{S} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$ and (2) for which no subset is an abductive explanation: $\forall \boldsymbol{S'} \subset \boldsymbol{S} : (\boldsymbol{S'} \in \boldsymbol{e}) \nvDash_{\mathcal{B}} \pi$.

Definition 6.1.5. All abductive explanations
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$ and a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$.
Output: The set of all sets $\boldsymbol{S} \subseteq \boldsymbol{E}$ for which (1) $(\boldsymbol{S} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$ and (2) for which no subset is an abductive explanation: $\forall \boldsymbol{S'} \subset \boldsymbol{S} : (\boldsymbol{S'} \in \boldsymbol{e}) \nvDash_{\mathcal{B}} \pi$.

6.2 NP-hardness

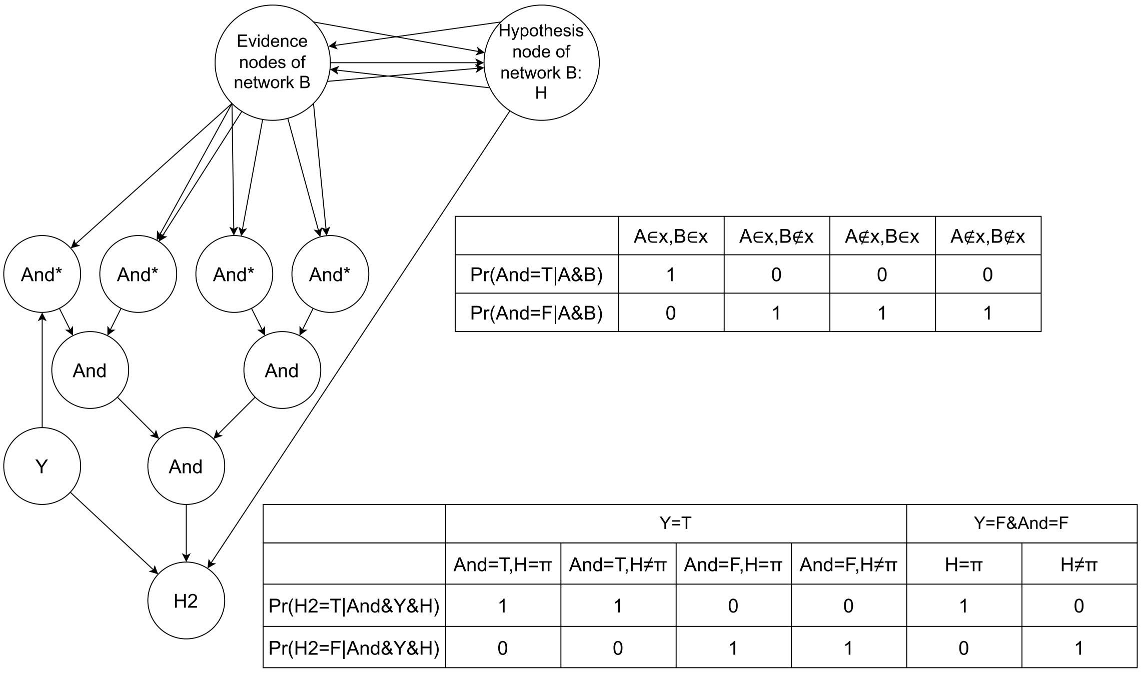
In this section, we will discuss the NP-hardness bounds we found for the problems in the previous section.

We will prove that the abductive confirmation decision problem is co-NP-hard. To do this, we will again use a reduction from the classification problem. To prove co-NP hardness, we will show that the inverse problem of the abductive confirmation decision problem is NP-hard. The inverse of a problem $\mathcal{P}$ is the problem of determining whether a given input is *not* a solution to $\mathcal{P}$. We can define the inverse abductive confirmation decision problem as follows:

Definition 6.2.1. Inverse abductive confirmation decision problem
Instance $\langle \mathcal{B}, \boldsymbol{e}, \pi, \boldsymbol{x} \rangle$**:** A Bayesian network classifier $\mathcal{B}$ including a directed acyclic graph $G = (\boldsymbol{V}_G, \boldsymbol{A}_G)$, a set of evidence $\boldsymbol{e} \in \Omega(\boldsymbol{E})$ where $\boldsymbol{E} \subseteq \boldsymbol{V}_G$, a prediction π for which $\boldsymbol{e} \vDash_{\mathcal{B}} \pi$, and a set

of variables $\boldsymbol{S} \subseteq \boldsymbol{E}$ with corresponding assignment $\boldsymbol{x} = \boldsymbol{S} \in \boldsymbol{e}$ for which $\boldsymbol{x} \vDash_{\mathcal{B}} \pi$.
Question: Does any set $\boldsymbol{S'} \subset \boldsymbol{S}$ satisfy $(\boldsymbol{S'} \in \boldsymbol{e}) \vDash_{\mathcal{B}} \pi$?

Note that this question is identical to the question whether $\boldsymbol{S}$ is **not** an abductive explanation of $(\boldsymbol{e}, \pi)$.



	A∈x,B∈x	A∈x,B∉x	A∉x,B∈x	A∉x,B∉x
Pr(And=T|A&B)	1	0	0	0
Pr(And=F|A&B)	0	1	1	1

	Y=T				Y=F&And=F	
	And=T,H=π	And=T,H≠π	And=F,H=π	And=F,H≠π	H=π	H≠π
Pr(H2=T|And&Y&H)	1	1	0	0	1	0
Pr(H2=F|And&Y&H)	0	0	1	1	0	1

Figure 6.1: A visualisation of the transformation used in the NP-hardness proof of the inverse abductive confirmation decision problem. The original network is abstractly represented using two nodes to denote all evidence nodes and the hypothesis node. A number of nodes are added to the original network. Node Y is an evidence node. The figure contains a description of the CPT of "And*"-nodes and $H2$. Note that in the tables shown, multiple cells of the CPT are merged. For example, $H \neq \pi$ might have multiple options depending on the number of states of H, but they have the same distribution in their CPT as described in the table. Note that the combination of $Y = F$ and $And = T$ cannot occur as Y is one of the ancestors of the bottom "And"-node. This combination is therefore not shown in the table.

Proof.

Transformation: Suppose we are given an instance $(\mathcal{B}, \boldsymbol{e}_1, V_h, \pi_1)$ of the Bayesian network classification decision problem. We now build an instance $\langle \mathcal{B}', \boldsymbol{e}_2, \pi_2 = \text{TRUE}, \boldsymbol{x} = \boldsymbol{e}_2 \rangle$ of the inverse abductive confirmation decision problem as follows: we transform $\mathcal{B}$ to $\mathcal{B}'$ which corresponds to Figure 6.1.

We add one evidence node Y and a hypothesis node $H2$. We connect Y and the evidence nodes of $\mathcal{B}$ to $H2$ using "And" and "And*" nodes as shown in the figure. Every "And" or "And*" node is connected in such a way that it predicts TRUE iff its ancestor evidence nodes all have the assignment they have in $\boldsymbol{x} = \boldsymbol{e}_1 \cup \{Y = T\}$. "And"-nodes have been discussed in Section 5.2.1.1, the CPT of "And*"-nodes is specified in Figure 6.1. The $H2$ node is specified in the figure too and is the new hypothesis node of $\mathcal{B}'$. $H2$ is also connected to the original hypothesis node H and to Y directly. The evidence $\boldsymbol{e}_2$ is equal to $\boldsymbol{e}_2 = \boldsymbol{x} = \boldsymbol{e}_1 \cup \{Y = T\}$. The hypothetical abductive explanation we want to disprove is $\boldsymbol{S} = \underline{\boldsymbol{e}_2}$.

To show that $\boldsymbol{x} \vDash \pi_1$ (which is a requirement for $\langle \mathcal{B}', \boldsymbol{e}_2, \pi_2 = \text{TRUE}, \boldsymbol{x} = \boldsymbol{e}_2 \rangle$ to be a valid instance of the inverse abductive confirmation decision problem), we must look at the definitions of the "And" and "And*" nodes. By definition, the "And*"-nodes predict TRUE if and only if both of their parents have the same value as they do in $\boldsymbol{x}$. This means that they will all have state TRUE with a probability of 1. As the "And"-nodes function identically to a logical and-gate, they will also predict TRUE with probability 1 since both of their parents predict TRUE with probability 1. Regardless of the value of H, the $H2$ node will have state TRUE when the bottom "And"-node has state TRUE and $Y = \text{TRUE}$. This means that the network predicts TRUE with evidence $\boldsymbol{x} = \boldsymbol{e}_2$ due to the CPT of $H2$.

Polynomial: We add $|\boldsymbol{E}| + 2$ nodes with polynomial CPTs, making this reduction polynomial. We need the $|\boldsymbol{E}|$ "And" or "And*" nodes to avoid an exponential increase in the CPT size with respect to the number of evidence nodes if we were to use a single CPT connected to all evidence nodes.

Correctness:

Lemma 4.1. If $\langle \mathcal{B}, \boldsymbol{e}_1, V_h, \pi_1 \rangle$ is a positive instance of the classification problem, then $\langle \mathcal{B}', \boldsymbol{e}_2, \text{TRUE}, \boldsymbol{e}_2 \rangle$ is a positive instance of the inverse abductive confirmation decision problem.

Proof. In this case, we can disprove the potential abductive explanation $\underline{\boldsymbol{e}_2}$ as there is a subset $\underline{\boldsymbol{e}_1} \subset \underline{\boldsymbol{e}_2}$ that predicts TRUE regardless of the value of $\underline{\boldsymbol{e}_2} \backslash \underline{\boldsymbol{e}_1} = Y$. In $\mathcal{B}'$, $\overline{\Pr}(H|\boldsymbol{e}_1)$ will still have the same distribution as $\Pr(H|\boldsymbol{e}_1)$ in $\mathcal{B}$, as Y is d-separated from the original network, because all chains are blocked by $\underline{\boldsymbol{e}_1}$ and the fact that $H2$ is a head-to-head node without any evidence nodes in its descendants. If $\mathcal{B}$ originally predicted π_1, this will make $\mathcal{B}'$ predict $\Pr(H2 = \text{TRUE}|\boldsymbol{e}_2) > 0.5$ when $Y = \text{FALSE}$ (and the bottom "And"-node is also FALSE as a result) by definition of its CPT. This means $\mathcal{B}'$ will predict TRUE regardless of the state of Y, which means $\underline{\boldsymbol{e}_1}$ or one of its subsets must be an abductive explanation. Therefore, $\underline{\boldsymbol{e}_2}$ cannot be one. ■

Lemma 4.2. If $\langle \mathcal{B}', \boldsymbol{e}_2, \text{TRUE}, \boldsymbol{e}_2 \rangle$ is a positive instance of the inverse abductive confirmation decision problem, $\langle \mathcal{B}, \boldsymbol{e}_1, V_h, \pi_1 \rangle$ is a positive instance of the classification problem.

Proof. In other words, we will prove that if $\boldsymbol{x}$ is not an abductive explanation, the original network $\mathcal{B}$ must predict π. The only way for $\boldsymbol{x}$ to not be an abductive explanation is if it has a subset that is an abductive explanation. If a subset of $\boldsymbol{x}$ is an abductive explanation, for at least one of the variables V_i in $\boldsymbol{x}$ the network must predict TRUE for all assignments of V_i. For the values of V_i that is not in $\boldsymbol{x}$, the lowermost "And"-node will have state FALSE with probability 1. This is because one of the "And*" nodes will have state FALSE with probability 1 as one of its parents, V_i, does not have the same value as in $\boldsymbol{x}$.

This further means that the only way for $\mathcal{B}'$ to predict TRUE regardless of the value of V_i is if $H2$ predicts TRUE while the bottom "And"-node has state FALSE with probability 1. The only way for that to happen with the CPT of $H2$ is if $Y = \text{FALSE}$ and $H = \pi$. This means that V_i must be Y, as it is the only way to get Y to be false whenever V_i does not take the value it has in $\boldsymbol{x}$. It furthermore means that the original network must predict π whenever the value of V_i is not equal to its value in $\boldsymbol{x}$, as that is the only other way for $H2$ to predict TRUE. As $V_i = Y$, it does not affect H, and H must predict state π when using evidence $\boldsymbol{e}_1$ (regardless of whether it is in $\mathcal{B}$ or $\mathcal{B}'$). ■

From the above, we can see that there is a polynomial transformation from classification decision problem to the inverse abductive confirmation decision problem. As classification is NP-hard [6], it follows that inverse abductive confirmation is NP-hard and therefore abductive confirmation is co-NP-hard. □

6.3 Algorithms

Algorithms for finding abductive explanations efficiently have been proposed by Ignatiev et al. [19] and Koopman [21]. To compare the complexities of the different problems, we will again describe a few simple naive algorithms that provide an upper bound on the running time.

Algorithm 6 is capable of finding all abductive explanations and serves as an upper bound for all introduced problems, as it can solve all of them.

Algorithm 6 visits all subsets of $\boldsymbol{E}$ to check for the first property of abductive explanations ($\boldsymbol{S} \in \boldsymbol{e} \models \pi$). After it has found all sets for which this property holds, it finds the minimal ones (Line 26). The algorithm has a number of ways to improve its efficiency.

Algorithm 6 All_abductive($\boldsymbol{E}$)

```
 1: List<Set> retSets = $\emptyset$
 2: subsets = $List(\{\boldsymbol{E}\})$
 3: for input_item $\in \boldsymbol{E}$ do
 4:     for subset $\in$ subsets do
 5:         any = FALSE
 6:         for var $\in$ ($\boldsymbol{E}\setminus$ subset) do
 7:             if (subset $\cup$ var) $\notin$ subsets then
 8:                 any = TRUE
 9:                 break
10:             end if
11:         end for
12:         if any then
13:             continue
14:         end if
15:         new_potential = subset $\setminus$ input_item
16:         for $\boldsymbol{s} \in \Omega(\boldsymbol{E}\setminus$new_potential$; \boldsymbol{e})$ do
17:             if new_potential$\cup\boldsymbol{s} \vDash \neg\pi$ then
18:                 any = TRUE
19:                 break
20:             end if
21:         end for
22:         if $\neg$any then
23:             subsets.add(new_potential)
24:         end if
25:     end for
26:     for subset1 $\in$ subsets do
27:         minimal = TRUE
28:         for subset2 $\in$ subsets do
29:             if subset2 $\subset$ subset1 then
30:                 minimal = FALSE
31:                 break
32:             end if
33:         end for
34:         if minimal then
35:             ret_sets.add(subset1)
36:         end if
37:     end for
38: end for
39: return ret_sets
```

- The (sub)sets are checked from large to small (i.e. a set is always checked after all its supersets). If for any set $\boldsymbol{X}$, $\boldsymbol{X} \vDash \pi$ does not hold, all subsets $\boldsymbol{X'} \subset \boldsymbol{X}$ can also never be an abductive explanation, as the same variable assignment invalidates both of these sets. This means we do not have to visit the subsets of sets we already ruled out as abductive explanations, acting as a manner of early-stop. This is checked in Line 6. To enforce that every set is checked after its supersets, the sets are generated as follows: we visit all variables in $\boldsymbol{E}$ in order (Line 3), and create new subsets by removing this variable from every existing subset in the order in which we created them (Line 15). This means that after removing a variable V_y from a set $\boldsymbol{Y}$, we know that $\boldsymbol{Y}\setminus V_y$ has been created later than $\boldsymbol{Y}$. Because we remove new nodes from sets in the order in which the sets were created, this also works for all subsets of $\boldsymbol{Y}$ and $\boldsymbol{Y}\setminus V_y$: the version with V_y is always created before the version without V_y. As this works for all nodes, we know that any set is created after all its supersets. Using induction we can also easily see that this way of generating sets also does not create duplicate sets: when removing a node from all sets, this does not create duplicates or sets that already exist as long as all existing sets do not contain duplicates. As we do not start out with any duplicate sets, no duplicate sets can ever exist.

- When checking if a set $\boldsymbol{X}$ can be an abductive explanation, we know that all of its supersets already satisfy the first condition of abductive explanations, as explained in the previous point. To check whether $\boldsymbol{X}$ could be an abductive explanation, we have to check if there is any combination of assignments to $\boldsymbol{E}\backslash\boldsymbol{X}$ that predicts any other class than π. We have already checked all combinations where at least one variable V_x in $\boldsymbol{E}\backslash\boldsymbol{X}$ has the same assignment as in $\boldsymbol{e}$ (in that case, we already checked this combination when checking whether $\boldsymbol{X} \cup V_x$ could be an abductive explanation). This means we only need to check the combinations where none of these variables have the same value as in $\boldsymbol{e}$. If the variables are binary, we only need to test one assignment per subset, as there are only two possible assignments for each variable. In the pseudocode, this is handled in Line 17

The algorithm runs in $O(I \cdot S^{|\boldsymbol{E}|})$ time, as, in the worst case, it has to apply inference for all subsets of evidence. Here, S is a bound on the number of states a variable can have.

To find any abductive explanation and confirm an abductive explanation, we can also create other algorithms that might be more efficient:

Algorithm 7 finds an abductive explanation by passing over all nodes in $\boldsymbol{S} = \boldsymbol{E}$ once, and checking if π still gets predicted when the node is removed for all assignments $(\boldsymbol{S} \in \boldsymbol{e}) \cup \Omega(\boldsymbol{e}\backslash\boldsymbol{S})$ (i.e., all assignments where nodes in $\boldsymbol{S}$ take their values in $\boldsymbol{e}$, and all other nodes can take any value). If it does, the node will be removed from $\boldsymbol{S}$ permantently. This results in a valid abductive explanation, as it is a minimal set that is still guaranteed to predict π. Calculating this takes $O(|\boldsymbol{E}| \cdot I \cdot S^{|\boldsymbol{E}|})$ time. This means that worst-case, Algorithm 6 for finding all abductive explanations needs less time while still giving the correct results. However, Algorithm 7 also takes at most $O(|\boldsymbol{E}| \cdot I \cdot S^{n-|\boldsymbol{E}|+1})$ time when the abductive explanation we find is of size n. This is because the algorithm only loops over all assignments of the nodes that are not in the explanation (and one node for which we are testing whether it is in the explanation). This means that this algorithm might be useful if it is known that most abductive explanations are relatively large.

Algorithm 7 Any_abductive($\boldsymbol{S}$)

```
for $V_i \in \boldsymbol{S}$ do
    Bool any = FALSE
    for $\boldsymbol{s} \in \Omega(\boldsymbol{e}\backslash(\boldsymbol{S}\backslash V_i))$ do
        if $neg(((\boldsymbol{S}\backslash V_i) \in \boldsymbol{e}) \cup \boldsymbol{s} \vDash \pi)$ then
            any = TRUE
        end if
    end for
    if $\neg$any then
        $\boldsymbol{S} = \boldsymbol{S}\backslash V_i$
    end if
end for
return $\boldsymbol{S}$
```

Algorithm 8 works similarly, but given a potential abductive explanation $\boldsymbol{S}$ it checks for every variable V_i in $\boldsymbol{S}$ whether $\boldsymbol{S}\backslash V_i$ is still guaranteed to predict π without it (if so, a subset of $\boldsymbol{S}$, is an abductive explanation, so $\boldsymbol{S}$ cannot be one). The runtime of this algorithm is $O(S^{(|\boldsymbol{E}|-|\boldsymbol{S}|+1)} \cdot |\boldsymbol{S}| \cdot I)$ time: for every element in $\boldsymbol{S}$, we apply inference for every possible combination of $\Omega(\boldsymbol{E}\backslash\boldsymbol{S})$.

Algorithm 8 Confirm_abductive($\boldsymbol{S}$)

```
for $V_i \in \boldsymbol{S}$ do
    Bool any = FALSE
    for $\boldsymbol{s} \in \Omega(\boldsymbol{e} \backslash (\boldsymbol{S} \backslash V_i))$ do
        if $neg(((\boldsymbol{S} \backslash V_i) \in \boldsymbol{e}) \cup \boldsymbol{s} \vDash \pi)$ then
            any = TRUE
        end if
    end for
    if $\neg$any then
        return FALSE
    end if
end for
return TRUE
```

6.4 An overview of complexities

Just like with contrastive explanations, we can visualize the bounds we have found to the complexity of the problems using a table (see Table 6.1). Column B is identical to the problems for contrastive explanations, because we can use the same MHS property to compute abductive explanations from contrastive ones that we used to compute contrastive explanations from abductive ones. We have seen that the problem from Definition 6.1.1 is co-NP-complete, which explains the colour of cell A1. We will now show that all problems in column A, except A4, are at least as hard as the abductive confirmation problem.

Specifically, they are at least as hard as the abductive confirmation decision problem, where the potential abductive explanation is the entire evidence set $\boldsymbol{E}$. This is a specific case of the general abductive confirmation decision problem. However, as our reduction from the classification problem in Section 6.2 always results in an abductive confirmation decision problem with this property, we know that this special case is also co-NP-hard.

We have seen that all problems can be solved by the same algorithm that, in the worst case, computes inference for all possible combinations of evidence.

Let us see how the results of problem A2, A3 and A5 answer problem A1:

- A2: When we find any abductive explanation, $\boldsymbol{E}$ can only be an abductive explanation if it is the one found, as all other sets are a subset of it. If it is an abductive explanation, it is the only one for the same reasons, and therefore it will be found.
- A3: The smallest abductive explanation is still an abductive explanation, so the same reasoning as in A2 works.
- A5: We can simply check if $\boldsymbol{E}$ is in the list of abductive explanations, though if it is, it must be the only one.

	A: Nothing	B: all abductive explanations $\mathcal{A}$
1. confirmation of $\boldsymbol{S}$ (Def 6.1.1)	$O(S^{(\lvert\boldsymbol{E}\rvert-\lvert\boldsymbol{S}\rvert+1)} \cdot \boldsymbol{S} \cdot I)$	$O(\lvert\mathcal{A}\rvert \cdot \lvert\boldsymbol{E}\rvert \cdot \lvert\boldsymbol{S}\rvert)$
2. any (Def 6.1.2)	$O(I \cdot S^{\lvert\boldsymbol{E}\rvert})$	$O(\lvert\mathcal{A}\rvert \cdot \lvert\boldsymbol{E}\rvert^2)$
3. smallest (Def 6.1.3)	$O(I \cdot S^{\lvert\boldsymbol{E}\rvert})$	
4. number (Def 6.1.4)	$O(I \cdot S^{\lvert\boldsymbol{E}\rvert})$	
5. all (Def 6.1.5)	$O(I \cdot S^{\lvert\boldsymbol{E}\rvert})$	[19]

Table 6.1: An overview of discussed complexities: the rows represent the problem definitions and the columns represent the known information. The cell colours indicate a known hardness proof, while the text contains the worst-case runtime of the discussed algorithms. A purple cell indicates that the problem is co-NP-hard, and a green cell indicates that a polynomial time algorithm is known. White means no NP-hardness was discussed for this problem. The rows and columns are named to easily refer to individual cells.

Chapter 7

Constraint propagation algorithm

In this section, an algorithm will be introduced: the constraint propagation algorithm. This algorithm will help us find abductive and contrastive explanations by answering a more general query: which combinations of assignments to the evidence variables would satisfy a constraint on a probability of interest?

More formally, given a set of evidence variables $\boldsymbol{E}$ and a **constraint** of the form $\Pr(v_a|\boldsymbol{e}) < \alpha$ for some v_a and α, the algorithm produces all variable assignments $\boldsymbol{x}$ for which: $(\boldsymbol{x} \in \Omega(\boldsymbol{E})) \wedge (\Pr(v_a|\boldsymbol{x}) < \alpha)$. The name we will use for this algorithm refers to these constraints that form the backbone of the algorithm. This is unrelated to other constraint-related algorithms such as algorithms that deal with constraint satisfaction problems.

Let us assume that we have a Bayesian network classifier with a binary hypothesis variable V_h. We then know that any combination of evidence that predicts $V_h = T$ must satisfy the constraint $\Pr(V_h = F|\boldsymbol{e}) < 0.5$.

We can now use the algorithm to find all combinations of evidence that result in a specific prediction from the Bayesian network classifier. Both abductive and contrastive explanations can now be found relatively easily from the set of evidence assignments that result in the required prediction. Keep in mind that the evidence assignments are the output of this algorithm, and by definition, the assignment of evidence is not known at many points in the algorithm. Counterintuitively, the notation $\Pr(v_x|\boldsymbol{e})$ for some assignment v_x to some node V_x does not have a set value but is a variable that can take different values depending on the assignments to the evidence nodes. However, the nodes that contain evidence are known and will not change during the algorithm. In this Section, V_x will often refer to the node that the algorithm is currently looking at.

Later in this section, we will discuss more in-depth how to use the algorithm to find abductive and contrastive explanations, and which alterations we could make to do so more efficiently.

To explain the algorithm, we will first introduce a simpler version that only works for networks with a specific tree-like structure, including an example that illustrates how it is able to find the correct results. After this, we will generalise the algorithm to work in a larger subset of singly connected graphs with binary variables. We will also prove discuss the running times, uses, and possible alterations to the algorithm.

7.0.1 Intuition behind the algorithm

The algorithm traverses the Bayesian network similarly to Pearl's belief propagation algorithm [35]. Roughly speaking, Pearl's algorithm obtains probability $\Pr(v_x|\boldsymbol{e})$ for a node $\underline{v_x}$, and uses this information to update $\Pr(v_y|\boldsymbol{e})$ for all states v_y in its neighbours by traversing the network. Each node propagates sufficient information to allow its neighbours to update their probabilities accordingly. Similarly, the constraint propagation algorithm also intends to propagate probabilities but only has a constraint on this probability and not its exact value. Therefore, the algorithm uses the constraint on the probabilities for a node to update the constraints on the probabilities of the neighbouring nodes. To illustrate the workings of the algorithm, consider Figure 7.1. We will describe how the algorithm acts in this specific Situation. The reasoning of the algorithm will be explained later in this chapter. The algorithm starts with a constraint for the hypothesis node H of $\Pr(H = F|\boldsymbol{e}) < 0.5$. Using the CPT of node H, we cannot determine whether and how the constraint can be satisfied yet: $\Pr(H = T|I = T) = 0.7$ would satisfy the constraint while $\Pr(H = T|I = F) = 0.4$ would not satisfy the constraint. Whether the constraint will be satisfied depends on $\Pr(I|\boldsymbol{e})$. We need information about other nodes to continue. The algorithm can determine what this constraint means for the parent of H. $\Pr(H = F|\boldsymbol{e}) < 0.5$ is equivalent to $\Pr(I = F|\boldsymbol{e}) < 0.67$ (the equations to determine this will be introduced and explained later). Again the algorithm will determine whether we can easily answer whether the constraint is satisfied. Since we can again not do this right now, the algorithm will transform the constraint to a constraint about E: $\Pr(E = T|\boldsymbol{e}) < 0.92$. As E is an evidence node we can easily see when this constraint is satisfied - when we have evidence $E = F$, since $\Pr(E = T|E = F) = 0$.

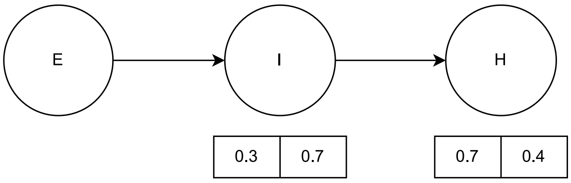


Figure 7.1: A simple Bayesian network with three nodes. E is an evidence node.

A notable difference between Pearl's algorithm and the constraint propagation algorithm is that Pearl's algorithm deals with actual changes to the (belief about) probabilities of nodes: a change in the probabilities of a node changes our knowledge/belief about other adjacent nodes. However, the constraint propagation algorithm deals with a hypothetical Situation: we have no actual knowledge about a node, but we wish to know which knowledge would result in a specific (constraint on a) probability. In essence, we reverse Pearl's algorithm to find the inputs that correspond to the observed output. In other words, the constraints do not contain knowledge that is used in the model, but rather specify our search range: we do not *assume* the constraint can be satisfied in the rest of the algorithm, but instead, we only consider the cases where it is. To make this distinction a bit clearer, consider the trivial Bayesian network in Figure 7.2. If we are applying inference and observe $B = T$, this results in knowledge about A: the posterior probability gets updated to $\Pr(A|B = T)$. In this case, we assume our observation is true, and then inference determines the probabilities for the other nodes given this knowledge. However, if we are applying the constraint propagation algorithm and want to know all observations a for A that would result in $\Pr(B = T|a) > 0.5$, we would have to conclude that there is no such observation. In this case, our search space has become empty.

The problem that this algorithm solves can be calculated by simply computing $\Pr(v_h|\boldsymbol{e})$ for every possible $\boldsymbol{e}$ using inference, and then checking whether the resulting value satisfies the constraint. However, our algorithm offers several advantages over this naive approach and existing algorithms for calculating explanations, such as Koopman and Renooij's algorithm [21] [22], which rely on multiple classifications. The following points outline the benefits of our algorithm, which is designed to be more efficient than existing methods. The implementation of the algorithm will be discussed later in this thesis, but these benefits are presented here to illustrate the purpose of the algorithm.

- Parts of the network that do not affect the outcome are not calculated. Sometimes, creating a new constraint for a neighboring node is unnecessary as we can determine that there are no evidence assignments that satisfy a constraint. Conversely, we can also find situations where all evidence assignments satisfy a constraint. In both cases, the algorithm will terminate if possible.

- Parts of the network that do not change (i.e., for which the posterior probabilities are not affected by a change in evidence) are not calculated multiple times. If we repeatedly classify with different assignments to the evidence nodes, much of the network will be identical for multiple input sets. The algorithm will attempt to reuse these calculations when possible.

So while the algorithm will definitely still have a running time that scales exponentially with

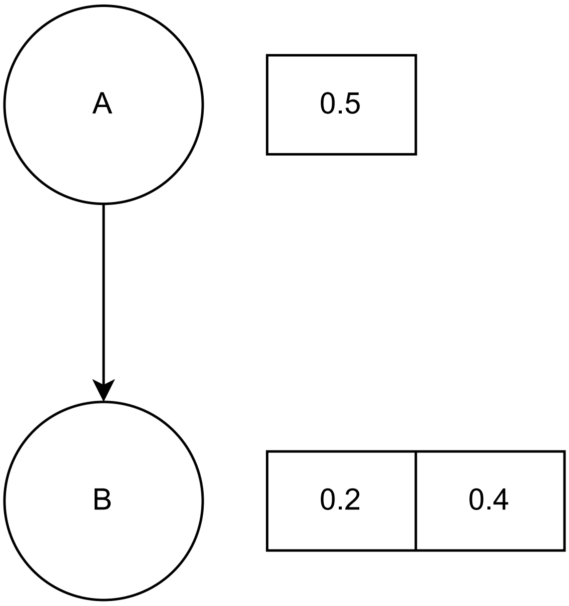


Figure 7.2: A simple Bayesian network with two nodes.

the number of evidence nodes, its aim is to reduce computation time by looking at the network itself as opposed to using inference to classify for a lot of different inputs.

7.1 The algorithm for upside-down binary trees

In this section, the constraint propagation algorithm will be explained in the scenario where we have a specific tree-like structure for our Bayesian network classifier. In Section 7.1.6, Pseudocode Algorithm 9 can be found, which will be referred to throughout this section.

7.1.1 Assumptions

We will start with a number of simplifying assumptions. Some of the implications of these assumptions are immediately stated, while others are discussed in Section 7.4.

Assumption 2. *There is one binary hypothesis node.*

Assumption 3. *The original constraint that is used as input for the algorithm is of the form* $\Pr(v_a|\boldsymbol{e}) < \alpha$ *for some value* $0 < \alpha \leq 1$ *and some assignment* v_h *to* V_h*.*

Assumption 4. *The exact value* α *in the constraint will not occur:* $<$ *and* $\leq$ *are, for all purposes, identical in the constraints of this algorithm.*

For any evidence assignment that satisfies the constraint $\Pr(\neg v_a|\boldsymbol{e}) < 0.5$, we will also say that $\boldsymbol{e}$ **predicts** v_a.

Assumption 4 serves to simplify the maths in this chapter. While the algorithm is, in principle, fully capable of dealing with both inclusive and exclusive inequalities, we would have to repeat a lot of the equations for both of these cases. Using this Assumption, we only have to use <.

Assumption 5. *All nodes use a conditional probability table (CPT) to determine their probabilities.*

We also have some temporary assumptions about the shape of the network:

Assumption 6. *The graph* G *is singly connected.*

Assumption 7. *All nodes in* $\boldsymbol{V}_G$ *are binary.*

Note that Assumptions 3, 7, and 4 also allow us to have constraints of the form $\Pr(v_a|\boldsymbol{e}) > \alpha$ for some value $0 \leq \alpha \leq 1$ and some assignment v_a to V_h, by rewriting it to $\Pr(\neg v_a|\boldsymbol{e}) < (1 - \alpha)$.

Assumption 8. *Every node in G has one child with the exception of the hypothesis node that has no children, and the evidence nodes that have no parents.*

Note that Assumptions 6 and 8 together severely constrain the graph of our network. An example of such a graph can be seen in Figure 7.4. We can interpret our constrained network as an upside-down tree with our hypothesis node as the root and our evidence nodes as leaves. This tree is upside-down in the sense that where normal trees have arcs from the root towards the leaves, the arcs in this network are reversed. Every node has multiple parents instead of multiple children as is usually the case. This notion of an upside-down tree serves the purpose of conveying the strength of the adopted assumptions and providing a way to easily specify this type of graph.

7.1.2 The backbone of the algorithm

At the start of the algorithm, we are given an hypothesis node and a constraint. For this constraint, and all constraints we calculate for other nodes, the constraint can be in one of four Situations. These Situations assume a constraint on the hypothesis node of form $\Pr(v_h|\boldsymbol{e}) < \alpha$. In the pseudocode, this distinction is determined on line 7. We will keep refering to these possible properties of a constraint as numbered **Situations** throughout this chapter. These Situations use the CPT of the current node of interest to distinguish what we know about the constraint without using any information about the evidence or other CPTs.

1. All combinations of the parents fulfill the constraint: $\forall \boldsymbol{p} \in \Omega(\boldsymbol{\rho}_{\underline{v_h}}) : \Pr(v_h|\boldsymbol{p}) < \alpha$
2. Any state for any parent can fulfill the constraint as long as the other parents have an appropriate probability. $\forall v_p \in \Omega(\boldsymbol{\rho}_{\underline{h}}) \exists \boldsymbol{p} \in \Omega(\boldsymbol{\rho}_{\underline{h}} \backslash \underline{v_{p1}}) : \Pr(v_h|\boldsymbol{p} \wedge v_p) < \alpha$. This is the case if no other Situations apply.
3. At least one of the parents has a state that would make it impossible to satisfy the constraint. I.e. in a simplified CPT with two parents, at least one row or column in the CPT has only values that do not fulfill the constraint. $\exists v_p \in \Omega(\boldsymbol{\rho}_{\underline{h}}) \forall \boldsymbol{p} \in \Omega(\boldsymbol{\rho}_{\underline{h}} \backslash \underline{v_p}) : \Pr(v_h|\boldsymbol{p} \wedge v_p) \geq \alpha$.
4. No combination of states for parents would fulfill the constraint, i.e., in the simplified (and possibly inverted) CPT, all values in the probability table are larger than α. $\forall \boldsymbol{p} \in \Omega(\boldsymbol{\rho}_{\underline{h}}) : \Pr(v_h|\boldsymbol{p}) \geq \alpha$.

Examples of when these Situations occur are visualised in Figure 7.3.

0.7	0.6
0.8	0.8

(a)

0.7	0.6
0.2	0.8

(b)

0.7	0.1
0.2	0.3

(c)

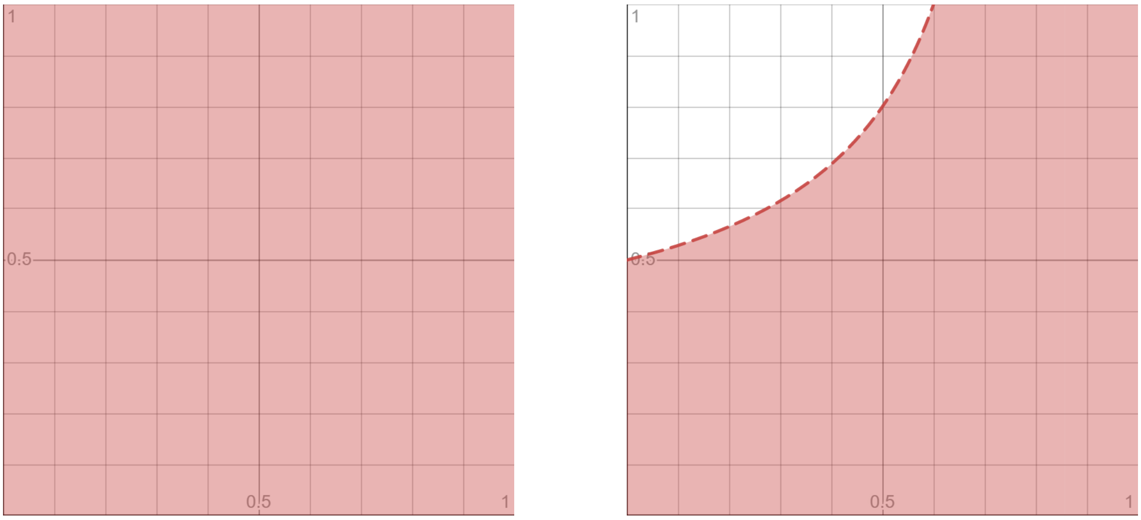
0.2	0.3
0.1	0.2

(d)

Table 7.1: CPTs corresponding to Figure 7.3. The CPTs are shown using the simplified notation we have introduced. The first parent determines the row, while the second parent determines the column.

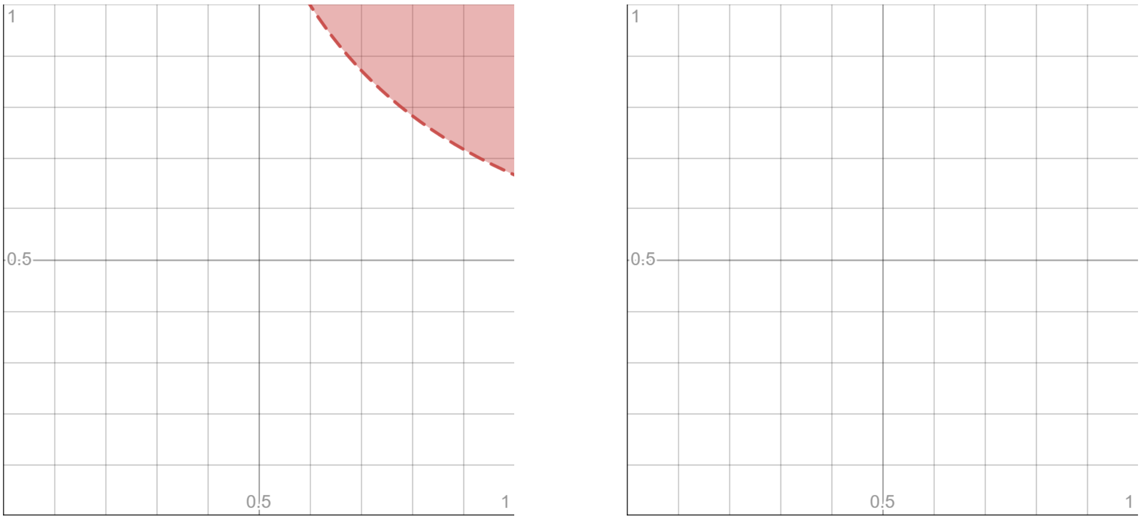
These four Situations determine which action the algorithm will take: they describe how easily this constraint can be satisfied. We will now discuss what each of the Situations mean and how the algorithm can efficiently continue. This corresponds to the contents of the outer if-statements between Lines 8 and 40 in the pseudocode of Algorithm 9.

In Situations 1 (Lines 8 to 14) and 4 (Lines 38 to 40), we are essentially done: we know which evidence assignments fulfill the constraint. In the case of Situation 1, this includes all assignments,



(a) Situation 1

(b) Situation 2



(c) Situation 3

(d) Situation 4

Figure 7.3: Visualisations of Constraints. The axes in these graphs correspond to the possible probabilities of the value TRUE of the binary parents $\{V_{p0}, V_{p1}\}$ of a binary node V_x. V_x has no other neighbours. The x-axis ranges from 0 to 1 corresponding to the probability $\Pr(V_{p0} = T|\boldsymbol{e})$, while the y-axis also ranges from 0 to 1 and corresponding to $\Pr(V_{p1} = T|\boldsymbol{e})$. The red area corresponds to the combinations of parent probabilities that would violate the constraint $\Pr(V_x = F|\boldsymbol{e}) < 0.5$. Each of the subfigures corresponds to a different CPT of V_x. These are listed in Table 7.1.

whereas in Situation 4, no evidence assignment would fulfill the constraint. The pseudocode shows that Situation 4 simply ends there, but Situation 1 does not, which will be explained in Section 7.1.4. In Situation 2 (Lines 15 to 26), we need additional information. We are unable to create a useful constraint for any of the parents, and the algorithm cannot continue by creating a new constraint. In Situation 2, by definition, constraining a single node cannot work: for any value of that node, there is a combination of values of the other parents that still satisfies the original constraint. The solution is to temporarily fall back to our original naive plan: inference for all possible combinations of evidence. However, doing this for only a portion of the graph may allow us to continue more effectively again, so we will only do this for one of our parents V_p (any parent would work, but some options are faster, see Section 7.4). For any assignment of evidence for the evidence nodes in $\boldsymbol{\rho}^*_{V_p}$, we calculate the distribution $\Pr(V_p|\boldsymbol{e})$. This can be done relatively easily given our current assumptions: we have to apply inference in the subgraph containing V_p and its ancestors for every combination of evidence. When inference is complete, we can sum out V_p from the CPT of our original node. However, we can do this for any combination of assignments to the evidence-ancestors of V_n, and for each of these we must continue the algorithm. In the pseudocode, this is the body of the `"for"` on Line 17 and the recursive call on line 20. In Situation 3 (Lines 27 to 37), we can recursively apply this algorithm to at least one of the parent nodes V_p. For all nodes that have a state that makes satisfying the existing constraint impossible regardless of the other parents, we can create a new constraint. We can determine the constraint for V_p from our existing constraint and the values in the CPT. If there are multiple parents that satisfy this criterion, the algorithm can again continue in multiple different ways. The exact calculation for the new constraint will be discussed in Section 7.1.3. Recursively applying this algorithm to V_p should give us all combinations of evidence that satisfy our new constraint on V_p. However, for all of these combinations of evidence we will also need the resulting distribution for $\Pr(V_p|\boldsymbol{e})$, to tighten the constraint on our original node (see Section 7.1.4). This means that the algorithm does not only have to return all combinations of evidence that satisfy our original constraint, but also the corresponding distribution for our current node. Why this is required and how this distribution is obtained is explained and discussed in Section 7.1.4. Just like in Situation 2 we now eliminated one parent from the CPT and have to continue for a number of combinations of evidence. Contrary to Situation 2, we do not necessarily have all combinations of evidence above V_p for which we have to continue the algorithm, but just the combinations that fulfill our new constraint and thus get returned by a recursive call of our algorithm.

In Situations 2 and 3 we now continue the algorithm with one less parent but multiple **parallel calls** or **branches**: we no longer have one situation, but a number of possible CPTs that may all result in different outcomes. Each of these CPTs is the result of a combination of assignments to evidence we have already visited, which will be explained more in-depth in Section 7.1.4. Note that these calls are not necessarily computed in parallel, though this should definitely be possible. We call them parallel calls because they are different calls that all traverse the same tree. In Section 7.4 we will discuss possible ways to deal with this but essentially we have to continue the algorithm for each of these CPTs. Continuing with the next parent will most likely result in multiple parallel calls again, which means we potentially have a parallel call for every combination of evidence assignments in the ancestors of the original node. When we have ended up in Situation 1 or 4, the algorithm is done with this branch and will return the results of all parallel calls it started, together with the distribution of V_h they result in. Note that the algorithm always ends in Situation 1 or 4 since a node needs to have parents to end up in Situation 2 or 3, both of which sum out parents. All branches that contribute to the output have ended in Situation 1, which we will also called **completed** branches or parallel calls.

7.1.3 Calculating new constraints

In this section we will discuss how to use a given constraint $\Pr(v_x|\boldsymbol{e}) < \alpha$ for a node to calculate a new constraint for one of its parents.

Using the definition of our conditional probability table and the knowledge that Assumptions 6 and 8 do not allow any descendants to be evidence nodes, we can rewrite the constraint to:

$$\begin{aligned}&\Pr(v_x|\boldsymbol{e}) < \alpha \iff \\ &\sum_{v_{p0} \in \boldsymbol{\rho}_{V_x}[0]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} (\Pr(v_{p0}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|v_{p0} \wedge \dots \wedge v_{pn})) < \alpha\end{aligned} \tag{7.1}$$

Using the fact that all nodes are binary valued (Assumption 7), we can write out one of the summations for our first parent v_{p0}.

$$\begin{aligned}&\Pr(v_x|\boldsymbol{e}) < \alpha \iff \\ &\sum_{v_{p1} \in \boldsymbol{\rho}_{V_x}[1]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} \Pr(V_{p0} = T|\boldsymbol{e}) \cdot \Pr(v_{p1}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = T \wedge v_{p1} \wedge \dots \wedge v_{pn}) + \\ &\sum_{v_{p1} \in \boldsymbol{\rho}_{V_x}[1]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} \Pr(V_{p0} = F|\boldsymbol{e}) \cdot \Pr(v_{p1}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = F \wedge v_{p1} \wedge \dots \wedge v_{pn}) < \alpha\end{aligned} \tag{7.2}$$

$$\begin{aligned}&\Pr(v_x|\boldsymbol{e}) < \alpha \iff \\ &\Pr(V_{p0} = T|\boldsymbol{e}) \cdot \sum_{v_{p1} \in \boldsymbol{\rho}_{V_x}[1]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} \Pr(v_{p1}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = T \wedge v_{p1} \wedge \dots \wedge v_{pn}) + \\ &\Pr(V_{p0} = F|\boldsymbol{e}) \cdot \sum_{v_{p1} \in \boldsymbol{\rho}_{V_x}[1]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} \Pr(v_{p1}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = F \wedge v_{p1} \wedge \dots \wedge v_{pn}) < \alpha\end{aligned} \tag{7.3}$$

At this point we will rename parts of the formula to keep the formulas clear while we reorder them.

$$\begin{aligned}&f1(\boldsymbol{e}) = \Pr(X|V_{p0} = T \wedge \boldsymbol{e}) = \\ &\sum_{v_{p1} \in \boldsymbol{\rho}_{V_x}[1]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} \Pr(v_{p1}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = T \wedge v_{p1} \wedge \dots \wedge v_{pn})\end{aligned} \tag{7.4}$$

and

$$\begin{aligned}&f2(\boldsymbol{e}) = \Pr(X|V_{p0} = F \wedge \boldsymbol{e}) = \\ &\sum_{v_{p1} \in \boldsymbol{\rho}_{V_x}[1]} \dots \sum_{v_{pn} \in \boldsymbol{\rho}_{V_x}[n]} \Pr(v_{p1}|\boldsymbol{e}) \cdot \dots \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = F \wedge v_{p1} \wedge \dots \wedge v_{pn})\end{aligned} \tag{7.5}$$

Using the fact that the nodes are binary (Assumption 7) and rewriting the formulas gives us the following results:

$$\Pr(v_x|\boldsymbol{e}) < \alpha \iff \Pr(V_{p0} = T|\boldsymbol{e}) \cdot f1(\boldsymbol{e}) + \Pr(V_{p0} = F|\boldsymbol{e}) \cdot f2(\boldsymbol{e}) < \alpha$$

$$\Pr(v_x|\boldsymbol{e}) < \alpha \iff \Pr(V_{p0} = T|\boldsymbol{e}) \cdot f1(\boldsymbol{e}) + (1 - \Pr(V_{p0} = T|\boldsymbol{e})) \cdot f2(\boldsymbol{e}) < \alpha$$

$$\Pr(v_x|\boldsymbol{e}) < \alpha \iff \Pr(V_{p0} = T|\boldsymbol{e}) \cdot f1(\boldsymbol{e}) + f2(\boldsymbol{e}) - \Pr(V_{p0} = T|\boldsymbol{e}) \cdot f2(\boldsymbol{e}) < \alpha$$

$$\Pr(v_x|\boldsymbol{e}) < \alpha \iff \Pr(V_{p0} = T|\boldsymbol{e}) \cdot f1(\boldsymbol{e}) - \Pr(V_{p0} = T|\boldsymbol{e}) \cdot f2(\boldsymbol{e}) < \alpha - f2(\boldsymbol{e})$$

$$\Pr(v_x|\boldsymbol{e}) < \alpha \iff \Pr(V_{p0} = T|\boldsymbol{e}) \cdot (f1(\boldsymbol{e}) - f2(\boldsymbol{e})) < \alpha - f2(\boldsymbol{e})$$

$\Pr(v_x|\boldsymbol{e}) < \alpha \iff [\Pr(V_{p0} = T|\boldsymbol{e}) < \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}]$ when $f1(\boldsymbol{e}) - f2(\boldsymbol{e}) > 0$ or $\Pr(v_x|\boldsymbol{e}) < \alpha \iff [\Pr(V_{p0} = T|\boldsymbol{e}) > \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}]$ when $f1(\boldsymbol{e}) - f2(\boldsymbol{e}) < 0$.

Note that dividing by a value with an unknown sign gives us different inequalities depending on whether the value is positive or negative. The first inequality has the same form as described in Assumption 3 while the second inequality does not. Rewriting the second inequality and using the lack of difference between $<$ and $\leq$ due to Assumption 4 gives us our two possible constraints for V_{p0} of the form specified in Assumption 3: $\Pr(v_x|\boldsymbol{e}) < \alpha \iff [\Pr(V_{p0} = T|\boldsymbol{e}) < \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}]$ when $f1(\boldsymbol{e}) - f2(\boldsymbol{e}) > 0$ or $\Pr(v_x|\boldsymbol{e}) < \alpha \iff [\Pr(V_{p0} = F|\boldsymbol{e}) < 1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}]$ when $f1(\boldsymbol{e}) - f2(\boldsymbol{e}) < 0$. $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ now have unknown values that depend on the known CPT of V_x, but also on the CPTs and evidence in $\boldsymbol{\rho}^*_{V_x} \backslash \boldsymbol{\rho}^*_{V_{p0}}$. $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ get their value depending on the distribution of $\Pr(\boldsymbol{P}|\boldsymbol{e})$ where $\boldsymbol{P} = \boldsymbol{\rho}_{V_x} \backslash V_{p0}$. We want to create a constraint that V_{p0} needs to fulfill in order to satisfy the original constraint on node V_x, regardless of the evidence and CPTs in $\boldsymbol{\rho}^*_{V_x}$.

As $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ are not exactly known, we cannot directly determine a usable α for our constraint yet. Our new constraint for node V_{p0} must follow from the original constraint regardless of the value of the other parents of V_{p0}, so we must find the least restrictive bound on α that any possible combination of $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ can give. Note that while $\boldsymbol{e}$ is not known, it is always the same for $f1$ and $f2$. This means we must maximise $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ or $1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ over $\boldsymbol{e}$ respectively depending on whether $f1(\boldsymbol{e}) - f2(\boldsymbol{e})$ is positive or negative. Note that we can only create a viable constraint if $f1(\boldsymbol{e}) - f2(\boldsymbol{e})$ is either always positive or always negative (and never zero) for all possible combinations of values for $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$. If they are not always positive or always negative we are unable to create a viable constraint and we are in Situation 2 instead of 3: if we can have a positive and a negative value for $f1(\boldsymbol{e}) - f2(\boldsymbol{e})$, this means that depending on the probabilities of $\boldsymbol{\rho}_{V_x} \backslash V_{p0}$, we can have a constraint that limits $\Pr(V_{p0} = T|\boldsymbol{e})$ or $\Pr(V_{p0} = F|\boldsymbol{e})$. Since we cannot take the least restrictive of those constraints, no constraint of the correct form is possible.

We could achieve a viable upper bound for $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ or $1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ respectively by finding a lower bound for $f1(\boldsymbol{e})$ and an upper bound for $f2(\boldsymbol{e})$ or vice versa. However, since $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ are not independent, this value might never be reachable with any probabilities of the parents: $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ might be able to take certain values, but this does not mean they can take these values with the same distribution for the parent nodes because $\boldsymbol{e}$ cannot take different values for $f1$ and $f2$. As an alternative to this naive approach observe that $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ can be interpreted as weighted sums that use the same weights: inside the summations we have terms of $\Pr(v_{p1}|\boldsymbol{e}) \cdot ... \cdot \Pr(v_{pn}|\boldsymbol{e}) \cdot \Pr(v_x|V_{p0} = T \wedge v_{p1} \wedge ... \wedge v_{pn})$. Here $\Pr(v_{p1}|\boldsymbol{e}) \cdot ... \cdot \Pr(v_{pn}|\boldsymbol{e})$ is a weight for value $\Pr(v_x|V_{p0} = T \wedge v_{p1} \wedge ... \wedge v_{pn})$ from the CPT of node V_x. All these weights sum to one, since they are probabilities for mutually exclusive and collectively exhaustive events. We will call the weights w_i and the values from $f1(\boldsymbol{e})$ $x1_i$. Similarly for $f2(\boldsymbol{e})$ we call the values $x2_i$. We can now rewrite the formula $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ to $\frac{\sum w_i \cdot (\alpha - x2_i)}{\sum w_i \cdot (x1_i - x2_i)}$.

Using the properties of the generalized mediant [3] we know that the following property holds for fractions: given a fraction for which the denominator and numerator are both weighted sums with the same weights $\frac{\sum_i w_i \cdot a_i}{\sum_i w_i \cdot b_i}$, the value of the fraction lies between the largest and smallest values of $\frac{a_i}{b_i}$.

This means that the value of $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ is always between the extreme values of $\frac{(\alpha - x2_i)}{(x1_i - x2_i)}$.

Note that this property of the mediant also means that equations such as $\frac{\Pr(v_x|\boldsymbol{e})}{\Pr(\neg v_x|\boldsymbol{e})}$ are bound between the minimum and maximum value that pairs of values from the CPT of V_x can reach, as this is also a weighted sum.

So to determine the maximum or minimum value $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$ can take over all i possible configurations of $\boldsymbol{\rho}_{V_x} \backslash V_{p0}$, we calculate $\frac{(\alpha - x2_i)}{(x1_i - x2_i)}$ when substituting $x1_i$ and $x2_i$ with every possible pair of $\Pr(v_x|V_{p0} = T \wedge v_{p1} \wedge ... \wedge v_{pn})$ and $\Pr(v_x|V_{p0} = F \wedge v_{p1} \wedge ... \wedge v_{pn})$, which are the values that are in the CPT of node V_x.

7.1.4 Returning probabilities

When the algorithm arrives at a node V_x (for example because this is the start node or the algorithm has just calculated a new constraint for this node), it creates a new constraint for one of the parents $V_{p0} = \boldsymbol{\rho}_{V_x}[0]$ when possible based on the constraint for V_x by determining the Situation. Assuming we have Situation 3 and are able to create a new constraint C, the algorithm is then able to find all combinations of evidence in the ancestors of V_{p0} that satisfy C using a recursive application of the algorithm. The next intuitive step would be to continue to the next parent $V_{p1} = \boldsymbol{\rho}_{V_x}[1]$ and again recursively apply the algorithm. However, we now have a problem: if we keep the existing constraint for V_{p1}, the evidence in the ancestors of V_{p0} and V_{p1} may satisfy the constraint individually while not satisfying our original constraint when combined. This is because this approach does not take the effect of the evidence in the ancestors of V_{p0} into account when determining a bound for V_{p1}. We will now explain an approach that does.
If we would know which probabilities V_{p0} would have given the evidence assignments we obtained from the recursive call, we could use this to sum it out of the probability table. We would then end up with a Situation where V_x effectively has one less parent and we can attempt to calculate a constraint for V_{p1} without having uncertainty about V_{p0}. In short, the algorithm will get these probabilities as return values from the recursive call on its parent and use them to change the CPT.

As an example of how this approach works compared to the prior flawed approach consider a node V_x with parents V_{p0} and V_{p1} as described above. We have an original constraint for this node of $\Pr(V_x = F|\boldsymbol{e}) < 0.5$. V_{p0} has a (C)PT that only contains the probability $\Pr(V_{p0} = T) = 0.45$. This means that regardless of its parents the nodes have the same distribution, which is not a situation that would normally occur as there are no dependencies between the node and its parents, but simpler for this example. The (C)PT of V_{p1} only contains the probability $\Pr(V_{p1} = T) = 0.5$. V_x has a CPT as can be seen in Table 7.2.

$\Pr(V_x = T \mid v_{p0} \wedge v_{p1})$	$v_{p1} = (V_{p1} = T)$	$v_{p1} = (V_{p1} = F)$
$v_{p0} = (V_{p0} = T)$	0.3	0.2
$v_{p0} = (V_{p0} = F)$	0.7	0.4

Table 7.2: Initial CPT for V_x

When the algorithm starts, it determines that we are in Situation 3. We have two possible constraints that we can currently create using the method described in Section 7.1.3: $\Pr(V_{p0} = T|\boldsymbol{e}) < 0.5$ or $\Pr(V_{p1} = F|\boldsymbol{e}) < 0.67$:

- For the constraint on V_{p0} we calculate all possible values of $\frac{(\alpha - x2_i)}{(x1_i - x2_i)}$: $\frac{(0.5-0.7)}{(0.3-0.7)} = 0.5$ and $\frac{(0.5-0.2)}{(0.4-0.2)} = -1.5$ Since $f1(\boldsymbol{e}) - f2(\boldsymbol{e})$ is always positive, the maximum value of those formulas becomes our new α.
- For the constraint on V_{p0} all values of $\frac{(\alpha - x2_i)}{(x1_i - x2_i)}$ are: $\frac{(0.5-0.2)}{(0.3-0.2)} = 3$ and $\frac{(0.5-0.4)}{(0.7-0.4)} = 0.33333$ Since $f1(\boldsymbol{e}) - f2(\boldsymbol{e})$ is always negative, the maximum value of $(1-3)$ and $(1-0.33333)$ becomes our new α.

We can fulfill both constraints individually: both constraints are always satisfied. However, the original constraint is not fulfilled: since evidence does not influence the values of V_{p0} and V_{p1}, regardless of evidence we have $\Pr(V_x = T|\boldsymbol{e}) = 0.5 \cdot 0.45 \cdot 0.3 + 0.5 \cdot 0.45 \cdot 0.2 + 0.5 \cdot 0.55 \cdot 0.7 + 0.5 \cdot 0.55 \cdot 0.4 = 0.415$ so the constraint is not satisfied. However, if we would first determine the bound on V_{p0}, recursively apply the algorithm, and return the $\Pr(V_{p0} = T|\boldsymbol{e}) = 0.45$, we could simplify the probability table to Table 7.3. The new constraint (still calculated using the original α) becomes $\Pr(v_{p1} = T|\boldsymbol{e}) < (1 - \frac{(0.5-0.31)}{(0.52-0.31)} = 0.095238)$ which cannot be satisfied.

$\Pr(V_x = T \mid v_{p1})$	$v_{p1} = T$	$v_{p1} = F$
	0.52	0.31

Table 7.3: CPT for V_x with V_{p0} summed out.

This means that after determining all combinations of evidence that satisfy a constraint C on a node V_x, the algorithm has to return the probabilities for V_x to its child for every combination of evidence in $\boldsymbol{\rho}^*_{V_x}$. This again works differently depending on the Situation.

- In Situation 1 we know that all possible combinations of evidence in $\boldsymbol{\rho}^*_{V_x}$ satisfy C. However, as we need $\Pr(V_x|\boldsymbol{e})$, we will calculate them using standard inference, for example using Pearl's algorithm, for every possible combination of evidence. Note that it is possible to get to Situation 4 when all parents have been removed due to Situation 2 or 3, in this case there are no combinations of evidence, and Pearl's algorithm will simply read the probabilities from the CPT and return those.
- In Situation 2 the algorithm removes a parent from V_x after which it is again applied to V_x. This means we are not done with V_x yet and Situation 2 is never the last Situation encountered for a node. This means that $\Pr(V_x|\boldsymbol{e})$ does not have to be returned yet.
- Situation 3 has the same property: it is never the last Situation encountered for a node, so nothing needs to be returned here. For obtaining the value of $\Pr(\boldsymbol{\rho}_{V_x}[0]|\boldsymbol{e})$ to sum out $\boldsymbol{\rho}_{V_x}[0]$, instead of using the algorithm itself like in Situation 2 it uses inference to do so.
- In Situation 4 there are no combinations of evidence that satisfy the constraint, so $\Pr(V_x|\boldsymbol{e})$ is not required.

A call of the algorithm with constraint C on node V_x thus returns the following information:

1. A set of evidence assignments that each assign a state to all evidence nodes in $\boldsymbol{\rho}^*_{V_x}$ that satisfy C: $\boldsymbol{e}^c = \{e_i \in \Omega(\underline{\boldsymbol{e}}) | \Pr(v_x|e_i) < \alpha\}$.
2. For every evidence assignment in the set: the probability distribution for V_x given the evidence: $\Pr(V_x|e_i) \forall e_i \in \boldsymbol{e}^c$.

When a constraint is created for an evidence node, there is only one assignment to the evidence that will satisfy the constraint: assume we have a constraint of $\Pr(v_e|\boldsymbol{e}) < \alpha$ for $v_e \in \Omega(V_e)$, we know that v_e will not satisfy the constraint, but $\neg v_e$ will. The algorithm will return the following information: the assignment $\neg v_e$, and distribution $\Pr(\neg v_e|\boldsymbol{e}) = 1$ (if the evidence is set to $\neg v_e$, V_e will have state $\Pr(\neg v_e|\boldsymbol{e})$ with probability 1 by definition of being an evidence node).

7.1.5 Example

To explain further how the algorithm works we will give an example using the network in Figure 7.4.

In this network we can see 6 evidence nodes labeled $E1$ to $E6$. We have a single hypothesis node H and the network adheres to the assumptions mentioned before. The CPTs are shown in the simplified way as discussed before where the rows indicate the value of the left parent (the top row indicates True, the bottom row indicates False), and the columns indicate the value of the right parent (left column indicates True, right column indicates False). The values inside the tables indicate the conditional probability for this variable to be True.

Consider the situation where we have a prediction False. To find all contrastive explanations we want to know all combinations of evidence that predict $H = T$. This gives us the constraint $\Pr(H = F|\boldsymbol{e}) < 0.5$. We will now manually apply the algorithm starting in node H.

H: The algorithms starts in the hypothesis node. In this case, we have Situation 3: there are values a parent can take that immediately violate the constraint. $\exists v_p \in \Omega(\boldsymbol{\rho}_{\underline{h}}) \forall \boldsymbol{p} \in \Omega(\boldsymbol{\rho}_{\underline{h}} \backslash \underline{v_p})$: $\Pr(v_h | \boldsymbol{p} \wedge v_p) \geq \alpha$. To fill in the formula: $\forall \boldsymbol{p} \in \Omega(\boldsymbol{\rho}_{\underline{h}} \backslash \underline{v_p})$: $\Pr(H = F | I1 = T \wedge \boldsymbol{p}) \geq 0.5$. If $I1$ would be TRUE, this would result in a probability of at most 0.1 for $\Pr(H = T|\boldsymbol{e})$, so a probability of at least 0.9 for $\Pr(H = F|\boldsymbol{e})$.

Since we have determined Situation 3, we can now create a new constraint for $I1$: our way of determining this constraint is to use the following formula: $[\Pr(v_{p0} = T|\boldsymbol{e}) < \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}]$ if $f1(\boldsymbol{e}) - f2(\boldsymbol{e}) > 0$ or $[\Pr(v_{p0} = F|\boldsymbol{e}) < 1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}]$ if $f1(\boldsymbol{e}) - f2(\boldsymbol{e}) < 0$ where $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ are defined in Equations 7.4 and 7.5.

We want a constraint for $I1$, which means that $V_{p0} = I1$. As determined in Section 7.1.3, we will substitute $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ with the combinations of $x1$ and $x2$ in the CPT where $V_{p0} = T$ $(x1 = 0.9, x2 = 0.8)$ and $V_{p0} = F$ $(x1 = 1, x2 = 0.2)$ respectively. This gives us the following two equations:

$$\frac{\alpha - x2_1}{x1_1 - x2_1} = \frac{0.5 - 0.8}{0.9 - 0.8} = -3$$
$$\frac{\alpha - x2_2}{x1_2 - x2_2} = \frac{0.5 - 0.2}{1 - 0.2} = 0.375$$

The first formula would result in the constraint $\Pr(I1 = T|\boldsymbol{e}) < -3$ which cannot be satisfied as probabilities are by definition 0 or higher. The second formula results in the constraint $\Pr(I1 = T|\boldsymbol{e}) < 0.375$, which is a meaningful constraint. As discussed in Section 7.1.3, all combinations of probabilities will result in a constraint between these two bounds, and we can take the loosest constraint by taking the highest α. This means our new constraint is $\Pr(I1 = T|\boldsymbol{e}) < 0.375$.

I1: For node $I1$ We have Situation 3 again, as $E2 = F$ would not allow us to fulfill the condition. This means we can create a new constraint for $V_{p0} = E2$. The formulas give us the following results:

$$\frac{\alpha - x2_1}{x1_1 - x2_1} = \frac{0.375 - 0.9}{0.3 - 0.9} = 0.875$$
$$\frac{\alpha - x2_2}{x1_2 - x2_2} = \frac{0.375 - 0.6}{0.4 - 0.6} = 1.125$$

This gives us the constraints $\Pr(E2 = F|\boldsymbol{e}) < 1 - 0.875$ and $\Pr(E2 = F|\boldsymbol{e}) < 1 - 1.125$. We again take the maximum which gives us a new constraint of $\Pr(E2 = F|\boldsymbol{e}) < 0.125$.

E2: We continue to node $E2$, but since this is an evidence node, we can fulfill the constraint by making the evidence TRUE. This branch of the algorithm is now complete. We have a set of one evidence assignment that does not violate the original constraint, with the corresponding probability of $\Pr(E2|\boldsymbol{e})$. The evidence-distribution pair currently looks like this: $\{(\{E2 = T\}, 1.0)\}$.

I1: Back in node $I1$ we continue the algorithm for every evidence-distribution pair that was returned, though there is only a single pair in this case. For each of the pairs we simplify the conditional probability table by summing out using the distribution that was obtained. In this case we reduce the table using value 1 for $\Pr(E2 = T)$. This results in a table that gives 0.3 when $E1 = T$ and 0.4 when $E1 = F$. Since we still have the constraint $\Pr(I1 = T|\boldsymbol{e}) < 0.375$, $E1 = F$ would violate the constraint, which means we have once again a case of Situation 3. Using the formulas, we obtain:

$$\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})} = \frac{0.375 - 0.4}{0.3 - 0.4} = 0.25$$

which results in a constraint of $\Pr(E1 = F|\boldsymbol{e}) < 1 - 0.25$ so $\Pr(E1 = F|\boldsymbol{e}) < 0.75$.

E1: Just like with $E2$ we can easily satisfy the constraint by setting the evidence for $E1$ to TRUE. This gives us the following return value: $\{(\{E2 = T, E1 = T\}, 1.0)\}$.

I1: When again returning to $I1$ we can use the new information to simplify the CPT as we did before: by using $\Pr(I1 = T) = 1$, we get a (C)PT of $\Pr(I1 = T|\emptyset) = 0.3$. This results in a trivial case of Situation 1 and we can now return with the following set: $\{(\{E2 = T, E1 = T\}, 0.3)\}$.

H: As we now return to H, we can fill in $\Pr(I1|\boldsymbol{e}) = 0.3$. This gives us the new conditional probabilities $\Pr(H = T|I2 = T) = 0.3 \cdot 0.1 + 0.7 \cdot 0.2 = 0.17$ and $\Pr(H = T|I2 = F) = 0.3 \cdot 0 + 0.7 \cdot 0.8 = 0.56$.
This means we are now done with $I1$ and its ancestors and we will continue to $I2$.
We have once again arrived in Situation 3 as our constraint $\Pr(H = F|\boldsymbol{e}) < 0.5$ cannot be fulfilled if $I2$ would have value TRUE. Using the formulas gives us: $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})} = \frac{0.5 - 0.44}{0.83 - 0.44} = 0.15$. This gives us the constraint $\Pr(I2 = T|\boldsymbol{e}) < 0.15$.

I2: Arriving at I2, we see that we are in Situation 2: there is no single parent that could take a value that always ensures the constraint is violated, and it is not the case that the constraint is always satisfied. In this case the algorithm will calculate the result of one of the parent trees for every combination of evidence in the tree. A simple way to choose for which parent we do this would be to take the parent with the lowest number of evidence-ancestors.

In this case we will use standard inference to calculate the distribution of $I2$ for all possible sets of evidence in the subgraph in the direction of $E3$. However, this is only one node, so we will obtain $\Pr(I2|E3 = T)$ and $\Pr(I2|E3 = F)$.

E3: Calculating the possible probabilities of $E3$ is trivial: it is either 0 or 1 depending on the evidence. This is then also the return value of this node. It returns the set $\{(\{E2 = T, E1 = T, E3 = T\}, 1.0), (\{E2 = T, E1 = T, E3 = F\}, 0)\}$.

I2: Since the return set from $E3$ contains two elements, the algorithm will split into two: for both of these elements we will have to run the entire algorithm.
Let us start with $(\{E2 = T, E1 = T, E3 = T\}, 1.0)$. In this case $\Pr(E3 = T|\boldsymbol{e})$ returns a value of 1, which means we can reduce our CPT to $\Pr(I2 = T|I3 = T) = 0.1$ and $\Pr(I2 = T|I3 = F) = 0.3$. Since we still have the constraint $\Pr(I2 = T|\boldsymbol{e}) < 0.15$ we are again in Situation 3. Filling in the formula gives us: $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})} = \frac{0.15 - 0.3}{0.1 - 0.3} = 0.73$ which results in the constraint $\Pr(I3 = F|\boldsymbol{e}) < (1 - 0.73)$ so $\Pr(I3 = F|\boldsymbol{e}) < 0.27$.
The other return value from $I3$ is 0, which results in a CPT containing $\Pr(I2 = T|I3 = T) = 0.8$

and $\Pr(I2 = T|I3 = F) = 0.1$. Filling this in in the formula gives us $\frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})} = \frac{0.15 - 0.1}{0.8 - 0.1} = 0.077$ which gives us the constraint $\Pr(I3 = T|\boldsymbol{e}) < 0.077$.

I3: We now still have two calls of the algorithm with two different constraints. If the constraints would be in the same "direction" we are able to do something more efficient (see Section 7.4), but as they are not we will have to resolve both calls separately.

The first constraint was $\Pr(I3 = F|\boldsymbol{e}) < 0.2692$. Since every conditional probability in the CPT satisfies this constraint we are in Situation 1. Situation 1 means that all combinations of evidence in the ancestors of this node would satisfy the constraint. We could determine the probabilities for all the parents for all possible evidence assignments, but as there are no more unvisited branches in the Bayesian network that is not necessary. If we just need the combinations of evidence that satisfy the original constraint returning back to H does not provide any benefits, so we add the following combinations of evidence to the set of return values of the algorithm:

- $\{E2 = T, E1 = T, E3 = T, E4 = T, E5 = T, E6 = T\}$
- $\{E2 = T, E1 = T, E3 = T, E4 = T, E5 = T, E6 = F\}$
- $\{E2 = T, E1 = T, E3 = T, E4 = T, E5 = F, E6 = T\}$

- $\{E2 = T, E1 = T, E3 = T, E4 = T, E5 = F, E6 = F\}$
- $\{E2 = T, E1 = T, E3 = T, E4 = F, E5 = T, E6 = T\}$
- $\{E2 = T, E1 = T, E3 = T, E4 = F, E5 = T, E6 = F\}$
- $\{E2 = T, E1 = T, E3 = T, E4 = F, E5 = F, E6 = T\}$
- $\{E2 = T, E1 = T, E3 = T, E4 = F, E5 = F, E6 = F\}$

If we would care about the resulting probabilities for the hypothesis node, we could continue the algorithm until we are back at H.

The other constraint was $\Pr(I3 = T|\boldsymbol{e}) < 0.077$. As none of the conditional probabilities in the CPT could satisfy this constraint, we are in Situation 4 and the algorithm ends this branch without adding anything to the output. This means the output of the total algorithm consists of the combinations of evidence added before.

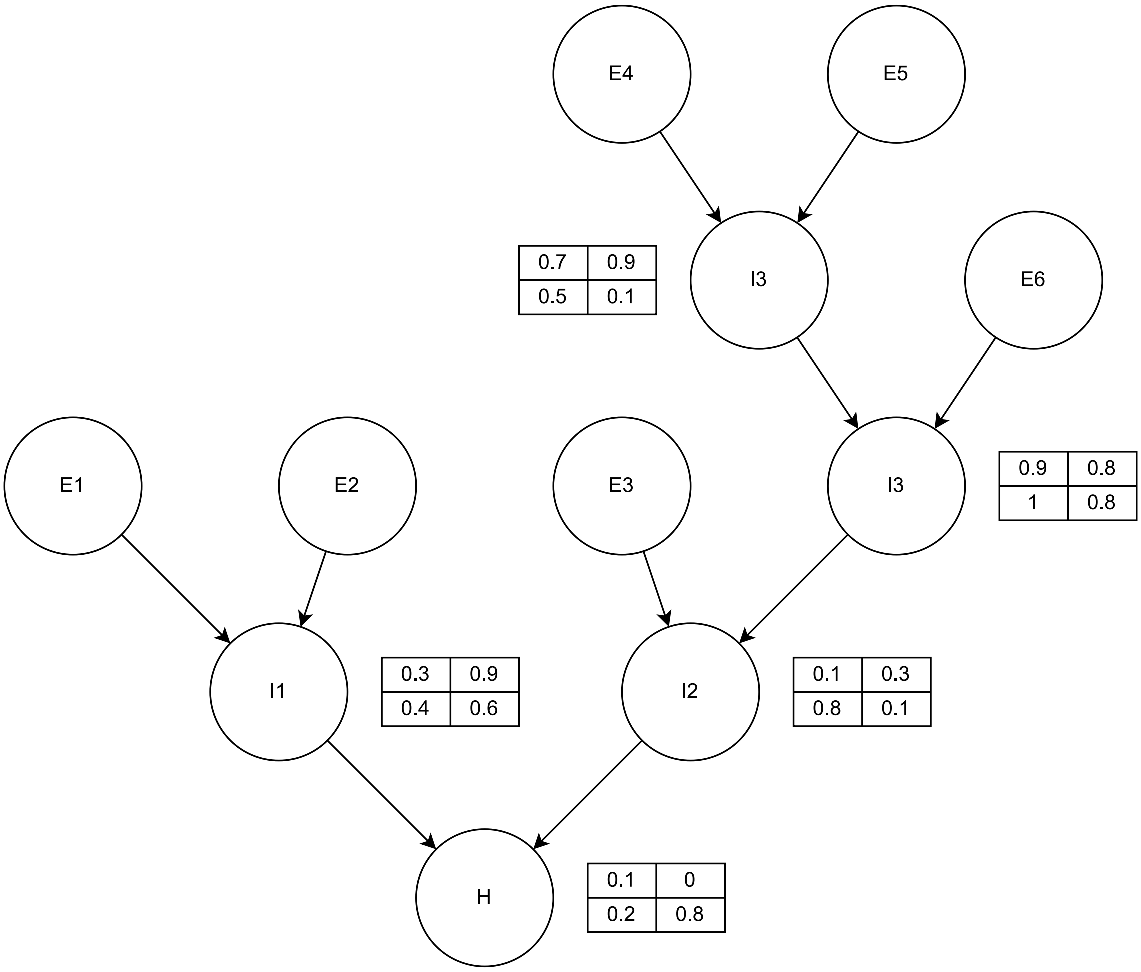


Figure 7.4: A Bayesian network used for the example in Section 7.1.5.

7.1.6 Pseudocode

Algorithm 9 ConstraintPropagationSimpleTree(Node V_n, constraint α)

```
returnSet = $\emptyset$;
if $V_n$ == evidenceNode then
    value = $V_n$.valueThatSatisfies($\alpha$);
    returnSet.add($V_n$ = value, Pr($value$) = 1);
    return returnSet;
end if
situation = checkSituation($V_n$, $\alpha$);
if situation == 1 then
    for evidence in AllAncestorEvidenceCombinations($V_n$) do
        probabilities = DetermineProbabilities($V_n$, evidence);
        returnSet.add(evidence, probabilities);
    end for
    return returnSet;
end if
if situation == 2 then
    $V_p$ = $V_n$.Parents[0];
    for evidence in AllAncestorEvidenceCombinations($V_p$) do
        probability = DetermineProbabilities($V_p$, evidence);
        $V_n$.SimplifyCPT($V_p$, probability);
        recursiveResults = ConstraintPropagationSimpleTree($V_n$, $\alpha$);
        for (evs, prob) in recursiveResults do
            returnSet.add(evs+evidence, prob);
        end for
    end for
    return returnSet;
end if
if situation == 3 then
    ($V_p$, $\beta$) = CalculateNewConstraint($\alpha$);
    parentResults = ConstraintPropagationSimpleTree($V_p$, $\beta$);
    for (evidence, probability) in parentResults do
        $V_n$.SimplifyCPT($V_p$, probability);
        recursiveResults = ConstraintPropagationSimpleTree($V_n$, $\alpha$);
        for (evs, prob) in recursiveResults do
            returnSet.add(evs+evidence, prob);
        end for
    end for
end if
if situation == 4 then
    return $\emptyset$;
end if
```

7.2 Less constrained networks

To actually be able to use the algorithm proposed above, we should generalize it to work for more types of graphs, so in this section we will change Assumption 8 (enforcing the tree-like structure) into Assumption 9. The challenge that this introduces is that evidence can also be in descendants of nodes instead of only in their ancestors. This means the algorithm will have to be able to propagate constraints "down", and the calculation of the constraints will become more complicated.

Assumption 9. *Every node in G has at most one parent if it has any evidence nodes in its descendants.*

The idea behind the constraints remains the same: given a constraint, the algorithm determines the Situation by trying all combinations of neighbouring nodes. If the algorithm can then create a constraint for a neighbouring node, it does so by transforming the existing constraint to a new one using the extreme values the other neighbours can take.

We introduce the notion of **upper graph** and **lower graph**: an upper graph of a node V_x is a part of the graph that becomes disconnected when the arc between V_x and one of its parents is removed from the graph. Similarly, a lower graph is the part of the graph that becomes disconnected when the arc between V_x and one of its children is removed from the graph. Because the graph is singly connected, all parents and children have an upper or lower graph, and they all have distinct upper or lower graphs.

7.2.1 A new type of constraint

For constraints on children we will use a different equation than for constraints on parents. Just like the original constraint is based on the causal parameter in Pearl's algorithm [35], the new constraint is based on the diagnostic parameter. However, in Pearl's algorithm we could (eventually) calculate the exact values and then normalise them. In our case, we only know a bound on the value after normalization. I.e. our constraint is on the relative size of the diagnostic parameters.therefore, The new constraints will have the following form:

$$\frac{\Pr(\boldsymbol{e}^{V_x \setminus V_{p0}} | v_{p0})}{\Pr(\boldsymbol{e}^{V_x \setminus V_{p0}} | \neg v_{p0})} < \beta \tag{7.6}$$

where v_{p0} is any assignment to V_{p0} and V_x is a child of V_{p0}. V_{p0} is the parent from which we created the constraint, so we transformed a constraint for V_{p0} into a constraint for V_x. For the notation used to describe groups of evidence, see Figure 7.5. We will call constraints of the type described in Assumption 3 *parent constraints*, and the new ones *child constraints*. Note that in contrast to α in the parent constraint, β does not have to be between 0 and 1 but can take any positive value. All nodes will only have a single constraint, where the type of constraint depends on whether the constraint was created by a parent or child of the node (a parent constraint is created by a child for its parent, and a child constraint is created by a parent for its child). We will derive child constraints in Section 7.2.3

7.2.2 Determining the Situation

In this section we will redefine the situations discussed in Section 7.1.2 and discuss how to determine the Situation for both parent and child constraints. We will first rewrite both types of constraints to a combination of the "effects" of the neighbours of the node we have a constraint for, and then we will try any combination of those neighbours and use the rewritten constraints to determine the Situation.

7.2.2.1 Rewriting formulas

Parent constraint The parent constraint still has the same form as in the previous chapter:

$$\Pr(v_x | \boldsymbol{e}) < \alpha \tag{7.7}$$

Using Assumption 7 we can rewrite this as:

$$\frac{\Pr(v_x | \boldsymbol{e})}{\Pr(\neg v_x | \boldsymbol{e})} < \frac{\alpha}{1 - \alpha} \tag{7.8}$$

We will now separate the evidence into $\boldsymbol{e}^-$ and $\boldsymbol{e}^+$, where $\boldsymbol{e}^+$ is all evidence that is connected to the node through one of its parents (so in an upper graph), and $\boldsymbol{e}^-$ is connected to the node through one of its children (so in a lower graph). This is also illustrated in Figure 7.5.

$$\frac{\Pr(v_x | \boldsymbol{e}^- \wedge \boldsymbol{e}^+)}{\Pr(\neg v_x | \boldsymbol{e}^- \wedge \boldsymbol{e}^+)} < \frac{\alpha}{1 - \alpha} \tag{7.9}$$

Using Bayes' rule gives us:

$$\frac{\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^+|v_x) \cdot \Pr(v_x)}{\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^+)} \cdot \frac{\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^+)}{\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^+|\neg v_x) \cdot \Pr(\neg v_x)} < \frac{\alpha}{1-\alpha} \tag{7.10}$$

Rewriting and using the independency between $\boldsymbol{e}^+$ and $\boldsymbol{e}^-$ given x.

$$\frac{\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(\boldsymbol{e}^+|v_x) \cdot \Pr(v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x) \cdot \Pr(\boldsymbol{e}^+|\neg v_x) \cdot \Pr(\neg v_x)} < \frac{\alpha}{1-\alpha} \tag{7.11}$$

Using Bayes' rule again:

$$\frac{\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(\boldsymbol{e}^+|v_x) \cdot \Pr(v_x) \cdot \Pr(\boldsymbol{e}^+)}{\Pr(\boldsymbol{e}^-|\neg v_x) \cdot \Pr(\boldsymbol{e}^+|\neg v_x) \cdot \Pr(\neg v_x) \cdot \Pr(\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha} \tag{7.12}$$

$$\frac{\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^+)}{\Pr(\boldsymbol{e}^-|\neg v_x) \cdot \Pr(\neg v_x|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha} \tag{7.13}$$

$$\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha} \tag{7.14}$$

We have now rewritten the parent constraint to a combination of effects from neighbours, meaning in terms of either $\Pr(V_x|\boldsymbol{e}^+)$ or $\Pr(\boldsymbol{e}^-|V_x)$. We will do the same with the child constraint before continuing. After that we will use local information to determine limits to the effects all neighbours can have on whether the constraint is satisfied or not.

Child constraint

Let us start with the child constraint as defined in Equation 7.6:

$$\frac{\Pr(\boldsymbol{e}^{V_x \setminus V_{p0}}|v_{p0})}{\Pr(\boldsymbol{e}^{V_x \setminus V_{p0}}|\neg v_{p0})} < \beta \tag{7.15}$$

We will split the evidence between evidence above V_x and evidence below V_x:

$$\frac{\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^{+\setminus V_{p0}}|v_{p0})}{\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^{+\setminus V_{p0}}|\neg v_{p0})} < \beta \tag{7.16}$$

Conditioning on the values of V_x gives us the following:

$$\frac{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^{+\setminus V_{p0}}|v_{p0} \wedge v_x) \cdot \Pr(v_x|v_{p0})]}{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^- \wedge \boldsymbol{e}^{+\setminus V_{p0}}|\neg v_{p0} \wedge v_x) \cdot \Pr(v_x|\neg v_{p0})]} < \beta \tag{7.17}$$

Since $\boldsymbol{e}^-$ and $\boldsymbol{e}^{+\setminus V_{p0}}$ are independent given V_x:

$$\frac{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_{p0} \wedge v_x) \cdot \Pr(\boldsymbol{e}^{+\setminus V_{p0}}|v_{p0} \wedge v_x) \cdot \Pr(v_x|v_{p0})]}{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|\neg v_{p0} \wedge v_x) \cdot \Pr(\boldsymbol{e}^{+\setminus V_{p0}}|\neg v_{p0} \wedge v_x) \cdot \Pr(v_x|\neg v_{p0})]} < \beta \tag{7.18}$$

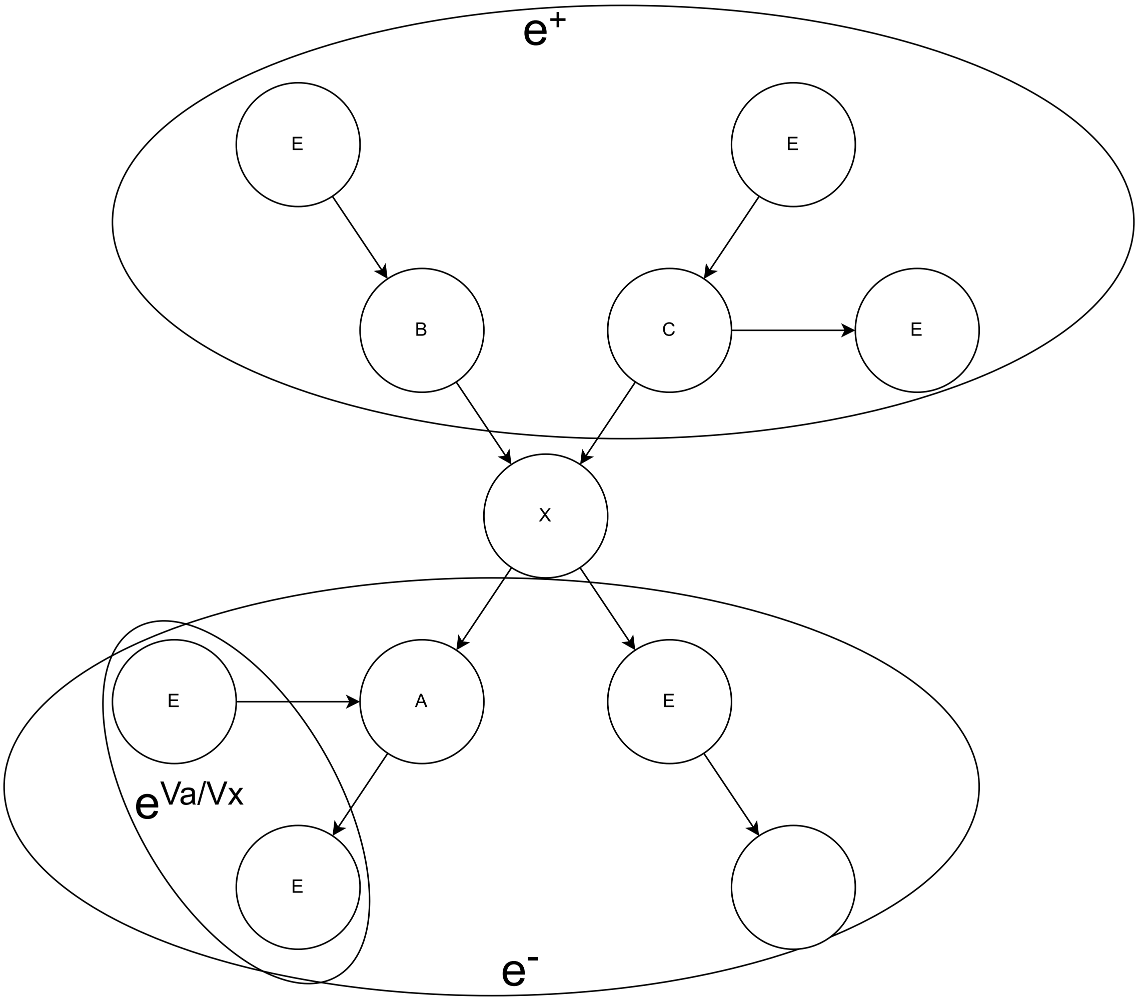


Figure 7.5: An illustration for the meaning of different notations for groups of evidence. $\boldsymbol{e}^{-}$ or $\boldsymbol{e}^{V_x-}$ is all evidence in the lower graphs of V_x, while $\boldsymbol{e}^{+}$ or $\boldsymbol{e}^{V_x+}$ refers to all evidence in the upper graphs of V_x. $\boldsymbol{e}^{V_a \setminus V_x}$ refers to all evidence that affects V_a, except the evidence that would be in a disconnected part of the graph if we remove V_x. In a similar way, we can also have $\boldsymbol{e}^{+\setminus V_c}$, which would refer to all evidence above V_x excluding the evidence in the upper graph of V_c.

Using the conditional independence of $\boldsymbol{e}^-$ and v_{p0} given v_x and Bayes' rule:

$$\frac{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) \cdot \Pr(\boldsymbol{e}^{+\setminus V_{p0}}|v_{p0})]}{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}) \cdot \Pr(\boldsymbol{e}^{+\setminus V_{p0}}|\neg v_{p0})]} < \beta \tag{7.19}$$

$$\frac{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})] \cdot \Pr(\boldsymbol{e}^{+\setminus V_{p0}})}{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})] \cdot \Pr(\boldsymbol{e}^{+\setminus V_{p0}})} < \beta \tag{7.20}$$

$$\frac{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})]}{\sum_{v_x \in \Omega V_x} [\Pr(\boldsymbol{e}^-|v_x) \cdot \Pr(v_x|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})]} < \beta \tag{7.21}$$

Using Assumption 7, we can write out the summation:

$$\frac{\Pr(\boldsymbol{e}^-|V_x = T) \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) + \Pr(\boldsymbol{e}^-|V_x = F) \cdot \Pr(V_x = F|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{\Pr(\boldsymbol{e}^-|V_x = T) \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}) + \Pr(\boldsymbol{e}^-|V_x = F) \cdot \Pr(V_x = F|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})} < \beta \tag{7.22}$$

We will define $\mathbf{X}$ as $\frac{\Pr(\boldsymbol{e}^-|V_x=T)}{\Pr(\boldsymbol{e}^-|V_x=F)}$.

$$\frac{\mathbf{X} \cdot \Pr(\boldsymbol{e}^-|V_x = F) \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) + \Pr(\boldsymbol{e}^-|V_x = F) \cdot \Pr(V_x = F|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{\mathbf{X} \cdot \Pr(\boldsymbol{e}^-|V_x = F) \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}) + \Pr(\boldsymbol{e}^-|V_x = F) \cdot \Pr(V_x = F|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})} < \beta \tag{7.23}$$

$$\frac{\mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) + \Pr(V_x = F|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{\mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}) + \Pr(V_x = F|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})} < \beta \tag{7.24}$$

$$\frac{\mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) + (1 - \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}))}{\mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}) + (1 - \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}))} < \beta \tag{7.25}$$

7.2.2.2 Trying combinations

Equations 7.14 and 7.25 are similar: they only depend on $\mathbf{X}$ and (a part of) $\Pr(v_x|\boldsymbol{e}^+)$. Let us revisit the Situations used in the previous section and their meaning given a constraint:

- If all parent probabilities satisfy the constraint, we are in Situation 1.
- If no probability assignment to parents exists that satisfies the constraint, we are in Situation 4.
- If we are not in Situation 4, but for some assignment of one parent, we always violate the constraint, we are in Situation 3 and can create a new constraint for this parent.
- If none of these apply, we are in Situation 2.

These Situations will essentially remain the same, except we will not only consider parents, but also children. For Situation 3, the new definition is as follows:

- We are in Situation 1 if all combinations of distributions of neighbours satisfy the constraint.
- We are in Situation 3 if some extreme point of a neighbour would lead to a violation of the constraint of the current node and we are not in Situation 4.
- We are in Situation 4 if all combinations of distributions of neighbours violate the constraint.

- If none of these apply, we are in Situation 2.

With extreme points we mean the minimum and maximum "effect" that the neighbour has on the left-hand side of the constraint given unknown distributions of its other neighbours: in case of parents, the extreme points are the degenerate distributions of $\Pr(V_p = T|\boldsymbol{e}^{V_x \setminus V_p}) = 1$ and $\Pr(V_p = T|\boldsymbol{e}^{V_x \setminus V_p}) = 0$. In case of child constraints, they are limited by the CPT of the child V_c which will be discussed in Section 7.2.4. For now, assume we computed the maximum and minimum value that $\frac{\Pr(\boldsymbol{e}^{V_c \setminus V_x}|v_x)}{\Pr(\boldsymbol{e}^{V_c \setminus V_x}|\neg v_x)}$ can take. Situation 3 means that there is some extreme point that a neighbour cannot take without violating the constraint for V_x, while there are also values it can take. From this, we also know that there must be a maximum/minimum value it can take, and therefore a constraint for this neighbour must be possible.

Using Equations 7.14 and 7.25, we can now determine the Situations: essentially we will try to find Situation 3 by setting one neighbour to an extreme point, and testing whether it is possible to satisfy the constraint. If we cannot do this for all degenerate distributions of a neighbour, we are in Situation 4. To check for Situation 1, we test whether it is possible to violate the constraint at all given any possible combination of probabilities of the neighbours. To find whether we can satisfy or violate a constraint we must minimise (for satisfying) or maximise (for violating) the left-hand side of the formula and check whether the inequality holds.

As mentioned before, we will assume we can create the maximum and minimum for formulas $\frac{\Pr(\boldsymbol{e}^{V_c \setminus V_x}|v_x)}{\Pr(\boldsymbol{e}^{V_c \setminus V_x}|\neg v_x)}$ and $\Pr(V_x|\boldsymbol{e}^+)$ assuming the neighbours can take any distribution. The minimum and maximum for the second formula can be found in the CPT of node V_x, and we will discuss how to find the minimum or maximum of the first formula in Section 7.2.4.

We can now find the minimum and maximum of Equations 7.14 and 7.25:

Minimising or maximising Equation 7.14:

$$\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha}$$

$\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)}$ can be expanded to $\prod_{V_c \in \boldsymbol{\sigma}_{V_x}} \frac{\Pr(\boldsymbol{e}^{V_c \setminus v_x}|v_x)}{\Pr(\boldsymbol{e}^{V_c \setminus v_x}|\neg v_x)}$. As all terms are positive and we know how to get the minimum or maximum of these terms, we can easily get the minimum or maximum of the product by using the respective minimum or maximum for each term. For determining Situation 3, we can exclude one of the terms from minimisation, and set it to its maximum instead.

For minimizing or maximizing $\frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)}$ we can just check the CPT of V_x. If we take the extreme point for one of the parents, we just have to look at a part of the CPT (the part that corresponds to the degenerate distribution of this parent).

Let us consider an example. Let us assume we know $\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)}$ lies between 0.5 and 2 (again, how to determine these values will be explained in Section 7.2.4). V_x. $\alpha = 0.5$, V_x has two parents named V_{p0} and V_{p1}, and the CPT of V_x contains the following values: $\Pr(v_x|V_{p0} = T, V_{p1} = T) = 0.8$, $\Pr(v_x|V_{p0} = T, V_{p1} = F) = 0.7$, $\Pr(v_x|V_{p0} = F, V_{p1} = T) = 0.4$, $\Pr(v_x|V_{p0} = F, V_{p1} = F) = 0.3$

If we start by checking whether we are in Situation 1, we must maximise the left-hand side of the equation, and check whether this still satisfies the constraint (if not, there is no way to violate the constraint). For this we take the maximum we found for $\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)}$, which is 2. We also find the entry in the CPT for which $\frac{x}{1-x}$ is highest. In this case, this is the first entry: $\Pr(v_x|V_{p0} = T, V_{p1} = T) = 0.8$. This gives us the following equation:

$$2 \cdot \frac{0.8}{0.2} < \frac{0.5}{1-0.5}$$

which is not satisfied.

We can then check whether we are in Situation 3 and can create a bound for V_{p0}: we will minimise the left-hand side of the equation, while V_{p0} has both of its degenerate distributions. If we cannot satisfy the constraint for one of them, we are in Situation 3. If we cannot satisfy the constraint for both of them, we are in Situation 4.

Let us start with $\Pr(V_{p0} = T|\boldsymbol{e}^{V_{p0}\setminus V_x}) = 1$: in this case, only two of our CPT entries are relevant, namely those that contain $V_{p0} = T$. minimising the left-hand side of the equation now gives:

$$0.5 \cdot \frac{0.7}{0.3} < \frac{0.5}{1 - 0.5}$$

As this does not satisfy the constraint, we are either in Situation 3 or 4.

If we now try to do the same for the degenerate distribution $\Pr(V_{p0} = F|\boldsymbol{e}^{V_{p0}\setminus V_x}) = 1$, the minimised left-hand side of the equation becomes:

$$0.5 \cdot \frac{0.3}{0.7} < \frac{0.5}{1 - 0.5}$$

As this does satisfy the constraint, we are in Situation 3 and can create a new constraint for V_{p0}

Minimising or maximising Equation 7.25:

$$\frac{\mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) + (1 - \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}))}{\mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}) + (1 - \Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}))} < \beta$$

In this case, both the numerator and denominator of the fraction are positive, which means that to maximise the left-hand side of the equation we must maximise the top and minimise the bottom of the fraction, and vice versa for minimising. However, **X** occurs in both the top and bottom of the fraction, and the other parts of the fraction depend on each other, which makes determining the maximum or minimum more complex. Note that if **X** is equal to 0, the left-hand side of the equation reduces to $\frac{1-\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{1-\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})}$, if **X** becomes larger, the left-hand side of the equation moves to $\frac{\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})}$. The value of **X** determines the value between those two points. This means the maximum of the left-hand side of the constraint is either determined by:

- the maximum of **X** and the maximum ratio between $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$.
- the minimum of **X** and the minimum ratio between $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$.

While the minimum of the left-hand side of the constraint is either

- the maximum of **X** and the minimum ratio between $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$.
- the minimum of **X** and the maximum ratio between $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$.

Note that we minimise or maximise the ratio between two values in the CPT, as a certain combination of parent distributions determines both values.

Again, to determine whether we are in Situation 3, we can set one term of **X** to its other extreme point, or consider part of the CPT of node V_x corresponding to an extreme point (degenerate distribution) of a parent.

Let us again consider an example. This example has the same values as the one in the previous paragraph. Here, V_{p0} in the constraint is equal to V_{p0} from the previous example. **X** is still bound between 0.5 and 2. β is 1.

We will again first determine whether we are in Situation 1 by maximising the left-hand side and testing whether we violate the constraint. As we have just seen, to maximise the left-hand side we have two options:

- the maximum of $\mathbf{X}$ and the maximum ratio between $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$.

  In this case, $\mathbf{X}$ is 2, and the maximum ratio of $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$ occurs when $\boldsymbol{e}^{+\setminus V_{p0}} = (V_{p1} = F)$: in this case, $\frac{\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})} = \frac{0.7}{0.3} = 2.33$. The entire equation now becomes:

  $$\frac{2 \cdot 0.7 + 0.3}{2 \cdot 0.3 + 0.7} = \frac{1.7}{1.5} < 1$$

  .

- the minimum of $\mathbf{X}$ and the minimum ratio between $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$.

  In this case, $\mathbf{X} = 0.5$, and the minimum ratio of $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})$ and $\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})$ occurs when $\boldsymbol{e}^{+\setminus V_{p0}} = (V_{p1} = T)$: in this case, $\frac{\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0})}{\Pr(V_x=T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})} = \frac{0.8}{0.4} = 2$. The entire equation now becomes:

  $$\frac{0.5 \cdot 0.8 + 0.2}{0.5 \cdot 0.4 + 0.6} = \frac{0.6}{0.8} < 1$$

  .

In this case, the first option results in the largest left-hand side of the constraint. This does not satisfy the constraint, so we are not in Situation 1.

We should now continue by testing whether are in Situation 3 by testing whether V_{p1} or the children of V_x can take an extreme point that makes it impossible to satisfy the constraint. This is not included in this example, but is done in the examples in Section 7.2.8.

In total, the number of calculations needed to determine the Situation takes $O(2^{\boldsymbol{\rho}_{V_x}} \cdot \boldsymbol{\rho}_{V_x} + 2^{\boldsymbol{\rho}_{V_s}} \cdot \boldsymbol{\sigma}_{V_x})$ calculations: worst-case we have to check for Situation 3 for all parents, where we have to compare $O(2^{\boldsymbol{\rho}_{V_x}})$ values in the CPT of V_x. We also have to compute the bounds that will be discussed in Section 7.2.4 once for each child, which takes $O(2^{\boldsymbol{\rho}_{V_s}})$ time per child V_s. When checking for Situation 3 for children, we only have to change a single value in the equation.

7.2.3 Transforming constraints

In this section we will work out how to create new constraints from existing constraints. Using Assumption 9 we have to consider 3 cases:

1. We have a parent constraint on V_x and we will create another parent constraint for one of its parents.
2. We have a parent constraint on V_x and we will create a child constraint for one of its children.
3. We have a child constraint on V_x and we will create another child constraint for one of its children.

Because of Assumption 9, we do not have to consider the case where we have a child constraint on node V_x and create a parent constraint. For this situation to occur, a node would need to have two parents, while the only path from the node to the starting node of the algorithm is through one of its parents. In this case, Assumption 9 means that there cannot be any evidence in the descendants of V_x. This means that all chains between the parents of V_x are blocked, and it is meaningless to create a constraint for one of its parents, as this does not affect the result of the algorithm. In that case, we already know that the left-hand side of the constraint on V_x will always have value 1 (there is no evidence in this lower graphs of V_{p0}that is affected by the probability of V_{p0}).

Propagating a parent constraint to parents (case 1)

Continuing from Equation 7.14, we can rewrite our formula to:

$$\frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha} \cdot \frac{\Pr(\boldsymbol{e}^-|\neg v_x)}{\Pr(\boldsymbol{e}^-|v_x)} \tag{7.26}$$

If we use an upper bound for $\frac{\Pr(\boldsymbol{e}^-|\neg v_x)}{\Pr(\boldsymbol{e}^-|v_x)}$, the right-hand side of the formula is now the loosest possible constraint. We will rename this constant to γ:

$$\frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \gamma \tag{7.27}$$

$$\frac{\Pr(v_x|\boldsymbol{e}^+)}{1-\Pr(v_x|\boldsymbol{e}^+)} < \gamma \tag{7.28}$$

$$\Pr(v_x|\boldsymbol{e}^+) < \gamma \cdot (1-\Pr(v_x|\boldsymbol{e}^+)) \tag{7.29}$$

$$\Pr(v_x|\boldsymbol{e}^+) < \gamma - \gamma \cdot \Pr(v_x|\boldsymbol{e}^+) \tag{7.30}$$

$$\Pr(v_x|\boldsymbol{e}^+) + \gamma \cdot \Pr(v_x|\boldsymbol{e}^+) < \gamma \tag{7.31}$$

$$(\gamma+1) \cdot \Pr(v_x|\boldsymbol{e}^+) < \gamma \tag{7.32}$$

$$\Pr(v_x|\boldsymbol{e}^+) < \frac{\gamma}{\gamma+1} \tag{7.33}$$

We now have the same starting formula as in Section 7.1.3 with $\alpha = \frac{\gamma}{\gamma+1}$, which means we can follow the same reasoning which gives us Equation 7.34. Recall that $f1(\boldsymbol{e})$ and $f2(\boldsymbol{e})$ are the weighted sum of CPT values where $V_{p0} = T$ and $V_{p0} = F$ respectively, which we minimise or maximise to maximise the right-hand side of the equations.

$$\begin{gathered}
[\Pr(V_{p0} = T|\boldsymbol{e}^{V_{p0}\setminus V_x}) < \frac{\alpha - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}{f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}] \text{ if } f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x}) > 0 \\
\text{or} \\
[\Pr(V_{p0} = F|\boldsymbol{e}^{V_{p0}\setminus V_x}) < 1 - \frac{\alpha - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}{f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}] \text{ if } f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x}) < 0
\end{gathered} \tag{7.34}$$

With α filled in:

$$\begin{gathered}
[\Pr(V_{p0} = T|\boldsymbol{e}^{V_{p0}\setminus V_x}) < \frac{\frac{\gamma}{\gamma+1} - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}{f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}] \text{ if } f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x}) > 0 \\
\text{or} \\
[\Pr(V_{p0} = F|\boldsymbol{e}^{V_{p0}\setminus V_x}) < 1 - \frac{\frac{\gamma}{\gamma+1} - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}{f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x})}] \text{ if } f1(\boldsymbol{e}^{V_{p0}\setminus V_x}) - f2(\boldsymbol{e}^{V_{p0}\setminus V_x}) < 0
\end{gathered} \tag{7.35}$$

Note that because we only considered evidence in the upper graphs of V_x, this became $\boldsymbol{e}^{V_{p0}\setminus V_x}$, as this is the same evidence but from the perspective of V_{p0}.

Propagating a parent constraint to children (case 2)

Again starting from Equation 7.14 and using the independence between children given their common parent:

$$\prod_{V_{ci} \in \boldsymbol{\sigma}_{V_x}} \frac{\Pr(e^{V_{ci}\setminus V_x}|v_x)}{\Pr(e^{V_{ci}\setminus V_x}|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha} \tag{7.36}$$

We will name the child we want to calculate a constraint for V_c and rewrite:

$$\frac{\Pr(e^{V_c \backslash V_x}|v_x)}{\Pr(e^{V_c \backslash V_x}|\neg v_x)} \cdot \prod_{V_{ci} \in \boldsymbol{\sigma}_{V_x} \backslash V_c} \frac{\Pr(e^{V_{ci} \backslash V_x}|v_x)}{\Pr(e^{V_{ci} \backslash V_x}|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha} \tag{7.37}$$

$$\frac{\Pr(e^{V_c \backslash V_x}|v_x)}{\Pr(e^{V_c \backslash V_x}|\neg v_x)} \cdot < \frac{\alpha}{1-\alpha} \cdot \frac{\Pr(\neg v_x|\boldsymbol{e}^+)}{\Pr(v_x|\boldsymbol{e}^+)} \cdot \prod_{V_{ci} \in \boldsymbol{\sigma}_{\underline{v_x}} \backslash V_c} \frac{\Pr(e^{V_{ci} \backslash V_x}|\neg v_x)}{\Pr(e^{V_{ci} \backslash V_x}|v_x)} \tag{7.38}$$

If we again take the upper bounds we established for neighbours except V_c, we have a constraint for $\frac{\Pr(e^{V_c \backslash x}|v_x)}{\Pr(e^{V_c \backslash x}|\neg v_x)}$ which we will call β, as it corresponds to the β for the new child constraint. Note that because of the property of mediants mentioned in Section 7.1.3, this means we have to determine the pair of values $\Pr(\neg v_x|\boldsymbol{e}^+)$ and $\Pr(v_x|\boldsymbol{e}^+)$ in the CPT of V_x for which this value is maximised, to create the loosest possible bound.

$$\frac{\Pr(\boldsymbol{e}^{V_c \backslash V_x}|v_x)}{\Pr(\boldsymbol{e}^{V_c \backslash V_x}|\neg v_x)} < \beta \tag{7.39}$$

Propagating a child constraint to children (case 3)

Continuing rewriting Equation 7.25 gives us:

$$\begin{gathered} \mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) + (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0})) < \\ \beta \cdot \mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0}) + \beta \cdot (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) \end{gathered} \tag{7.40}$$

$$\begin{gathered} \mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) < \beta \cdot \mathbf{X} \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0}) + \\ \beta \cdot (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) - (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0})) \end{gathered} \tag{7.41}$$

$$\begin{gathered} \mathbf{X} \cdot (\Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) - \beta \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) < \\ \beta \cdot (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) - (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0})) \end{gathered} \tag{7.42}$$

Expanding $\mathbf{X}$ again gives us:

$$\begin{gathered} \frac{\Pr(\boldsymbol{e}^-|V_x = T)}{\Pr(\boldsymbol{e}^-|V_x = F)} \cdot (\Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) - \beta \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) < \\ \beta \cdot (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) - (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0})) \end{gathered} \tag{7.43}$$

Using the independence between children and extracting the child V_c we want to create a constraint for:

$$\begin{gathered} \prod_{V_{ci} \in \boldsymbol{\sigma}_{\underline{V_x = T}} \backslash V_c} \frac{\Pr(\boldsymbol{e}^{V_{ci} \backslash V_x}|V_x = T)}{\Pr(\boldsymbol{e}^{V_{ci} \backslash V_x}|V_x = F)} \cdot \frac{\Pr(\boldsymbol{e}^{V_c \backslash V_{p0}}|V_x = T)}{\Pr(\boldsymbol{e}^{V_c \backslash V_x}|V_x = F)} \cdot \\ (\Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) - \beta \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) < \\ \beta \cdot (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) - (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0})) \end{gathered} \tag{7.44}$$

So if $(\Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) - \beta \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0}))$ is positive, this results in the constraint:

$$\begin{gathered} \frac{\Pr(\boldsymbol{e}^{V_c \backslash V_{p0}}|V_x = T)}{\Pr(\boldsymbol{e}^{V_c \backslash V_x}|V_x = F)} < \prod_{V_{ci} \in \boldsymbol{\sigma}_{\underline{V_x = T}} \backslash V_c} \frac{\Pr(\boldsymbol{e}^{V_{ci} \backslash V_x}|V_x = F)}{\Pr(\boldsymbol{e}^{V_{ci} \backslash V_x}|V_x = T)} \\ \cdot \frac{\beta \cdot (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0})) - (1 - \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}))}{(\Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge v_{p0}) - \beta \cdot \Pr(V_x = T|\boldsymbol{e}^{+\backslash V_{p0}} \wedge \neg v_{p0}))} \end{gathered} \tag{7.45}$$

and if it is negative the constraint will be:

$$\begin{aligned} \frac{\Pr(\boldsymbol{e}^{V_c\setminus V_{p0}}|V_x = T)}{\Pr(\boldsymbol{e}^{V_c\setminus V_x}|V_x = F)} &> \prod_{V_{ci}\in\boldsymbol{\sigma}_{\underline{V_x=T}}\setminus V_c} \frac{\Pr(\boldsymbol{e}^{V_{ci}\setminus V_x}|V_x = F)}{\Pr(\boldsymbol{e}^{V_{ci}\setminus V_x}|V_x = T)} \\ &\cdot \frac{\beta\cdot(1-\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})) - (1-\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}))}{(\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) - \beta\cdot\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}))} \end{aligned} \tag{7.46}$$

which is equal to:

$$\begin{aligned} \frac{\Pr(\boldsymbol{e}^{V_c\setminus V_x}|V_x = F)}{\Pr(\boldsymbol{e}^{V_c\setminus V_{p0}}|V_x = T)} &< \prod_{V_{ci}\in\boldsymbol{\sigma}_{V_x}\setminus V_c} \frac{\Pr(\boldsymbol{e}^{V_{ci}\setminus V_x}|V_x = T)}{\Pr(\boldsymbol{e}^{V_{ci}\setminus V_x}|V_x = F)} \\ &\cdot \frac{(\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}) - \beta\cdot\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0}))}{\beta\cdot(1-\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge \neg v_{p0})) - (1-\Pr(V_x = T|\boldsymbol{e}^{+\setminus V_{p0}} \wedge v_{p0}))} \end{aligned} \tag{7.47}$$

Both of these constraints can be transformed to the constraint in Equation 7.6 by taking the right-hand side of Equation 7.45 or 7.47 as de new β and assigning v_x to $V_x = \textsc{True}$ or $V_x = \textsc{False}$ respectively. To determine the constraint that will be passed on we will take the loosest possible bound by maximising the right-hand side of the equation by trying all possible extreme points of neighbours of V_x other than V_{p0} and V_c, as was described in Section 7.2.2.2. β will get the maximum value that can be reached in this manner. If not all constraints are in the same direction we should have ended up in Situation 2.

7.2.4 Lower and upper bounds for children

There have been multiple occasions where we assumed we could create minima and maxima for the equation $\frac{\Pr(\boldsymbol{e}^{V_c-}|v_c)}{\Pr(\boldsymbol{e}^{V_c-}|\neg v_c)}$ or $\frac{\Pr(\boldsymbol{e}^{V_d\setminus V_c}|v_c)}{\Pr(\boldsymbol{e}^{V_d\setminus V_c}|\neg v_c)}$ for some node V_c with assignment v_c and child $V_d \in \boldsymbol{\sigma}_{V_c}$.

We want to have an upper or lower bound that can be found while only looking at the local nodes. This means we have no information about the evidence nodes, and must assume they can take any value. We will only consider the CPTs of V_c and V_d. This means that, for example, $\Pr(\boldsymbol{e}^{V_d-}|v_d)$ can take any value from 0 to 1, and $\Pr(v_c|\boldsymbol{e}^{V_d+})$ can take any weighted sum of the values in the CPT of node V_c.

Lemma 4.3. $\frac{\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c)}{\Pr(\boldsymbol{e}^{V_d\setminus c}|\neg v_c)}$ for any $V_d \in \boldsymbol{\sigma}_{V_c}$ is bound between
$\min\left[\frac{\Pr(V_d=\textsc{True}|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{True}|\neg v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}, \frac{\Pr(V_d=\textsc{False}|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{False}|\neg v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}\right]$ for the combination of parent states that minimise those values and
$\max\left[\frac{\Pr(V_d=\textsc{True}|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{True}|\neg v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}, \frac{\Pr(V_d=\textsc{False}|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{False}|\neg v_c\wedge\boldsymbol{e}^{V_d+\setminus c})}\right]$ for the combination of parent states that maximise those values.

Proof. $\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c)$ is also called the diagnostic parameter in Pearl's algorithm so we know we can calculate it using the following formula:

$$\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c) = \alpha\cdot\sum_{v_d\in\Omega V_d}[\Pr(\boldsymbol{e}^{V_d-}|v_d)\cdot\Pr(v_d|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})]$$

where α is some normalization constant. We know $\sum_{v_d\in\Omega V_d}\Pr(v_d|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})$ is equal to 1. We can see that the difference between $\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c)$ and $\Pr(\boldsymbol{e}^{V_d\setminus c}|\neg v_c)$ is bound by the difference between $\Pr(\boldsymbol{e}^{V_d-}|v_d)$ and $\Pr(\boldsymbol{e}^{V_d-}|\neg v_d)$ (i.e. if $\Pr(\boldsymbol{e}^{V_d-}|v_d)$ and $\Pr(\boldsymbol{e}^{V_d-}|\neg v_d)$ would be the same, $\frac{\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c)}{\Pr(\boldsymbol{e}^{V_d\setminus c}|\neg v_c)}$ would be 1).

Let us assume the difference between $\Pr(\boldsymbol{e}^{V_d-}|v_d)$ and $\Pr(\boldsymbol{e}^{V_d-}|\neg v_d)$ is as high as possible: one of them is 0 and one of them is 1.

If $\Pr(\boldsymbol{e}^{V_d-}|v_d) = 1$ this would mean the highest possible value of $\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c)$ is equal to the highest possible value of $\alpha\cdot\Pr(v_d|v_c\wedge\boldsymbol{e}^{V_d+\setminus c})$. Similarly, the lowest possible value of $\Pr(\boldsymbol{e}^{V_d\setminus c}|v_c)$

is equal to the lowest possible value of $\alpha \cdot \Pr(v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$.

The highest possible value of $\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c) - \Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)$ occurs when $\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)$ is as high as possible and $\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)$ is as low as possible. This means we can rewrite the maximum of $\frac{\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)}{\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)}$ to $\frac{\alpha \cdot \Pr(v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})}{\alpha \cdot \Pr(v_d | \neg v_c \wedge \boldsymbol{e}^{V_d + \setminus c})}$ where $\alpha \cdot \Pr(v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$ is maximised and $\alpha \cdot \Pr(v_d | \neg v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$ is minimised. The alphas cancel out which gives us part of the statement in the lemma.

If we would instead assume $\Pr(\boldsymbol{e}^{V_d -} | \neg v_d)$ to be 1, the highest value of $\frac{\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)}{\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)}$ would similarly become $\frac{\alpha \cdot \Pr(\neg v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})}{\alpha \cdot \Pr(\neg v_d | \neg v_c \wedge \boldsymbol{e}^{V_d + \setminus c})}$ where $\alpha \cdot \Pr(\neg v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$ is maximised and $\alpha \cdot \Pr(\neg v_d | \neg v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$ is minimised.

For the minimum values we just have to swap the minimization and maximization. For example, if $\Pr(\boldsymbol{e}^{V_d -} | v_d) = 1$, the minimum of $\frac{\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)}{\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)}$ becomes $\frac{\alpha \cdot \Pr(v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})}{\alpha \cdot \Pr(v_d | \neg v_c \wedge \boldsymbol{e}^{V_d + \setminus c})}$ where $\alpha \cdot \Pr(v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$ is minimised and $\alpha \cdot \Pr(v_d | \neg v_c \wedge \boldsymbol{e}^{V_d + \setminus c})$ is maximised. ☐

To find a numerical bound we can again use the property of the mediant we used in 7.1.3 and simply compare numbers in the CPT to find the combination of degenerate parent states that would give the maximal or minimal value of the equations in the lemma.

Using this lemma we can take the product of $\frac{\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)}{\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)}$ for all children of V_c to calculate $\frac{\Pr(\boldsymbol{e}^{V_c -} | v_c)}{\Pr(\boldsymbol{e}^{V_c -} | \neg v_c)}$. For example, if we want an upper bound for $\frac{\Pr(\boldsymbol{e}^{V_c -} | v_c)}{\Pr(\boldsymbol{e}^{V_c -} | \neg v_c)}$, we can take the product of all upper bounds for $\frac{\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)}{\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)}$ for all children of V_v, as they are always larger than 0.

7.2.5 Evidence nodes with child constraints

The removal of Assumption 8 also means that the algorithm can now encounter children that are evidence nodes. As we have seen propagating downwards means that we have a different constraint: $\frac{\Pr(\boldsymbol{e}^{V_x \setminus V_{p0}} | v_{p0})}{\Pr(\boldsymbol{e}^{V_x \setminus V_{p0}} | \neg v_{p0})} < \beta$. If V_x is an evidence node, the result depends on any additional parents V_x may have. Because Assumption 9 enforces that V_x then only has one parent, it is easy to determine from the CPT which states would satisfy our constraint: the only evidence in the lower graphs is in V_x, so the left-hand side of the constraint corresponds directly to values in the CPT for both possible assignments to V_x.

7.2.6 Backtracking with child constraints

Just like with the original constraints, we need to return a value to simplify the equation in the previous node, as this means it has more information when trying to determine the situation. Recall Section 7.1.4, where we did this for parent constraints. In the case of the child constraints we need the value of $\alpha \cdot \Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)$ and $\alpha \cdot \Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)$ for some constant α, as this allows us to calculate $\frac{\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c)}{\Pr(\boldsymbol{e}^{V_d \setminus c} | \neg v_c)}$, which we need to simplify equations 7.25 and 7.14. Given

$$\Pr(\boldsymbol{e}^{V_d \setminus c} | v_c) = \frac{1}{\alpha} \cdot \sum_{v_d \in \Omega V_d} [\Pr(\boldsymbol{e}^{V_d -} | v_d) \cdot \Pr(v_d | v_c \wedge \boldsymbol{e}^{V_d + \setminus c})]$$

which we saw before in Section 7.2.4, we can calculate these values using the values we would get from all our neighbours: the product of the values from the children and the probabilities for the parents can be filled in in the formula and in combination with the CPT of node V_d we can strengthen the constraint on our parent node by filling in $\frac{\Pr(\boldsymbol{e}^{V_c -} | v_c)}{\Pr(\boldsymbol{e}^{V_c -} | \neg v_c)}$ for the child we are at. The separate values may later be used to recursively calculate the child constraint return value just as we did here. In Situations 1 and 2 Pearl's algorithm is also able to return those values, as they correspond to its diagnostic parameter.

7.2.7 The updated algorithm

As we can see, the basis of the algorithm essentially does not change. The difference now is that determining the Situations and calculating the new constraints has become much more complex. Additionally, there are different return values depending on the nature of the constraint: for a parent constraint, we return $\Pr(v_x|\boldsymbol{e}_x^V)$, while for child constraints, we return both $\alpha \cdot \Pr(\boldsymbol{e}^{V_x \setminus V_{p0}}|v_{p0})$ and $\alpha \cdot \Pr(\boldsymbol{e}^{V_x \setminus V_{p0}}|\neg v_{p0})$.

7.2.8 Example

In this section we will give two examples of how the algorithm deals with graphs. Example 1 will show how to deal with the new child constraints, while Example 2 will also show how to determine abductive and contrastive explanations using this algorithm.

7.2.8.1 Example 1

Let us start with a simple example of how the algorithm propagates constraints to its children. Consider the example in Figure 7.6. Suppose that we have an initial constraint of $\Pr(V_h = \text{FALSE}|\boldsymbol{e}) < 0.4$.

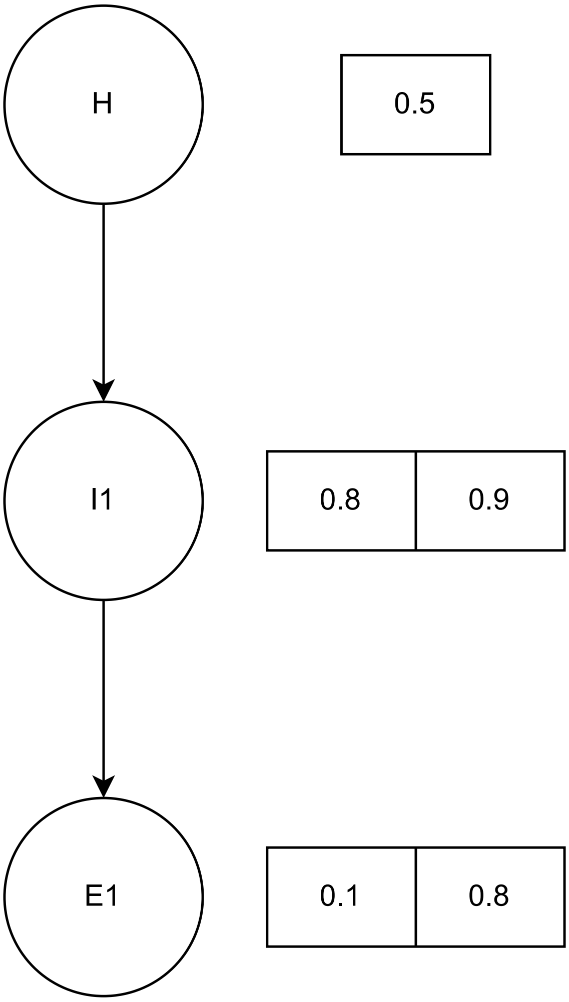


Figure 7.6: The Bayesian network used in Example 1.

H To start we will determine the Situation. Since we have a parent constraint we will use Equation 7.14. We now need to create a lower and upper bound for the effects of all children, and the effects of all combinations of parents. Since we only have a single child, we will now calculate a lower and upper bound for $\dfrac{\Pr(\boldsymbol{e}^{V_{I1}\setminus V_h}|V_h = \textsc{False})}{\Pr(\boldsymbol{e}^{V_{I1}\setminus V_h}|V_h = \textsc{True})}$:

Let us define v_h as $V_h = \textsc{False}$

Using Lemma 4.3 we know that the lower bound is equal to the lowest possible value of $\min\left[\frac{\Pr(V_d=\textsc{True}|v_h \wedge \boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{True}|\neg v_h \wedge \boldsymbol{e}^{V_d+\setminus c})}, \frac{\Pr(V_d=\textsc{False}|v_h \wedge \boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{False}|\neg v_h \wedge \boldsymbol{e}^{V_d+\setminus c})}\right]$. In this case $V_d = V_{I1}$ so $\frac{\Pr(V_d=\textsc{True}|v_h \wedge \boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{True}|\neg v_h \wedge \boldsymbol{e}^{V_d+\setminus c})} = \frac{0.9}{0.8}$ and $\frac{\Pr(V_d=\textsc{False}|v_h \wedge \boldsymbol{e}^{V_d+\setminus c})}{\Pr(V_d=\textsc{False}|\neg v_h \wedge \boldsymbol{e}^{V_d+\setminus c})} = \frac{0.1}{0.2}$. The minimum of those two values is $\frac{0.1}{0.2} = 0.5$. The upper bound is the maximum of the same values, which means it is $\frac{0.9}{0.8} = 1.25$.

To find the Situation, we will determine whether the constraint is satisfied given the lower bound and upper bound of $\dfrac{\Pr(\boldsymbol{e}^{V_{I1}\setminus V_h}|V_h = \textsc{False})}{\Pr(\boldsymbol{e}^{V_{I1}\setminus V_h}|V_h = \textsc{True})}$. The equation to calculate the new constraint is based on Equation 7.14 and using the independency between children when conditioned on their parent:

$$\prod_{V_c \in \boldsymbol{\sigma}_{V_h}} \frac{\Pr(e^{V_c\setminus V_h}|v_h)}{(\Pr(e^{V_c\setminus V_h}|\neg v_h))} \cdot \frac{\Pr(v_h|\boldsymbol{e}^+)}{\Pr(\neg v_h|\boldsymbol{e}^+)} < \frac{\alpha}{1-\alpha}$$

Filling in this formula for the lower bound gives

$0.5 \cdot \frac{0.5}{0.5} < \dfrac{0.4}{0.6}$ which is equal to $0.5 < 0.67$ and satisfies the constraint. The upper bound gives us $2 < 0.67$, in which case the constraint is clearly violated.

This means we are in Situation 3: it is not the case that all combinations of variables violate the constraint or satsify it, but when assigning the lower bound to one of the children, the constraint is no longer satisfiable regardless of the values of the other neighbours of V_h (which do not exist). This means we can create a new constraint for V_{e1}. We will now convert a parent constraint into a child constraint, which uses Equation 7.38.

Our new constraint value β is defined as the highest possible value of $\dfrac{\alpha}{1-\alpha} \cdot \frac{\Pr(\neg v_x|\boldsymbol{e}^+)}{\Pr(v_x|\boldsymbol{e}^+)} \cdot \prod_{V_{ci} \in \boldsymbol{\sigma}_{V_x} \setminus V_{I1}} \frac{\Pr(e^{V_{ci}\setminus x}|\neg v_x)}{\Pr(e^{V_{ci}\setminus x}|v_x)} = \frac{0.4}{0.6} \cdot \frac{0.5}{0.5} \cdot 1 = 0.67$

So our constraint for V_{I1} is $\frac{\Pr(\boldsymbol{e}^{V_{I1}\setminus V_h}|V_h=\textsc{False})}{\Pr(\boldsymbol{e}^{V_{I1}\setminus V_h}|V_h=\textsc{True})} < 0.67$

I1 To determine the Situation we will use Equation 7.25:

$$\frac{\mathbf{X} \cdot \Pr(V_{I1} = \textsc{True}|\boldsymbol{e}^{V_{I1}+\setminus V_h} \wedge V_h = \textsc{False}) + (1 - \Pr(V_{I1} = \textsc{True}|\boldsymbol{e}^{V_{I1}+\setminus V_h} \wedge V_h = \textsc{False}))}{\mathbf{X} \cdot \Pr(V_{I1} = \textsc{True}|\boldsymbol{e}^{V_{I1}+\setminus V_h} \wedge V_h = \textsc{True}) + (1 - \Pr(V_{I1} = \textsc{True}|\boldsymbol{e}^{V_{I1}+\setminus V_h} \wedge V_h = \textsc{True}))} < \beta$$

We can determine in which Situation we are. Since V_{I1} does not have parents beside V_h, the conditional probabilities can directly be found in the CPT and we can fill them in: $\frac{\mathbf{X} \cdot 0.9+0.1}{\mathbf{X} \cdot 0.8+0.2} < 0.67$. To fill in the possible values of $\mathbf{X} = \frac{\Pr(\boldsymbol{e}^{V_{I1}-}|V_{I1=\textsc{True}})}{\Pr(\boldsymbol{e}^{V_{I1}-}|V_{I1=\textsc{False}})}$, we must expand this formula to the product of effects from children and calculate the lower and upper bounds on the values from all children. Since there is only one child this results in $\frac{\Pr(\boldsymbol{e}^{V_{E1}\setminus V_{I1}}|V_{I1=\textsc{True}})}{\Pr(\boldsymbol{e}^{V_{E1}\setminus V_{I1}}|V_{I1=\textsc{False}})}$. Using the same method as in V_h to calculate an upper bound and a lower bound results in values of 4.5 and 0.125.

$\frac{4.5 \cdot 0.9+0.1}{4.5 \cdot 0.8+0.2} < 0.67$ and $\frac{0.125 \cdot 0.9+0.1}{0.125 \cdot 0.8+0.2} < 0.67$ both do not fulfill the constraint, meaning we are in Situation 4 and there are no solutions to the original constraint. The algorithm ends here.

7.2.8.2 Example 2

For this example, we are going to find a minimum abductive explanation for a specific input-output pair for a Bayesian network. Consider Figure 7.7 with input-output pair ($\boldsymbol{e} = \{E1 = T, E2 = T, E3 = T, E4 = T\}, \pi = T\}$. In section 7.5, we will discuss how to use the algorithm to find abductive and contrastive explanations, which will be done in this example (so it might be beneficial to read that first to understand why the algorithm makes certain choices). When having multiple parallel calls, we first consider the parallel call(s) with the least evidence assignments in $\boldsymbol{e}$ for which no parallel call exists with the opposite assignment. As discussed in Section 7.5, if all

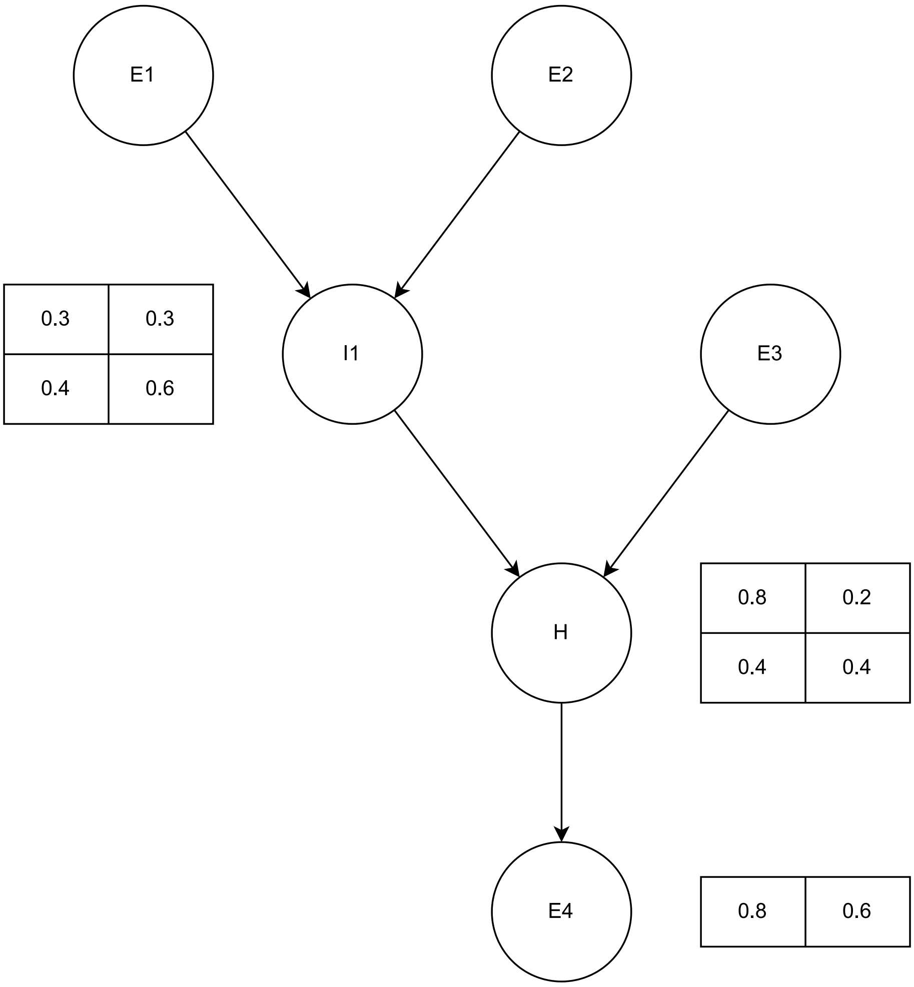


Figure 7.7: The Bayesian network used in Example 2. Note that the CPTs of the evidence nodes without parents are irrelevant as the nodes are guaranteed to be evidence. The CPTs are therefore not included in the figure.

parallel calls with the same subset in $\boldsymbol{e}$ end up in Situation 1, we have found the smallest abductive explanation.

H We start in node V_h with parent constraint $\Pr(H = F|\boldsymbol{e}) < 0.5$. We will first determine the Situation, using Equation 7.14: $\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \dfrac{\alpha}{1-\alpha}$. Here, $v_x = H = F$ and $\alpha = 0.5$. To first test whether we are in Situation 1, we will try to violate the constraint by maximising the left-hand side of the equation. We can do this by getting an upper bound for $\frac{\Pr(\boldsymbol{e}^-|v_x)}{\Pr(\boldsymbol{e}^-|\neg v_x)}$ and $\frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)}$ independently. Using Lemma 4.3 and looking at the CPT of node H, we get an upper bound of $\frac{0.4}{0.2} \cdot \frac{0.8}{0.2} = 8 < \frac{0.5}{1-0.5} = 1$ which violates the constraint. This means we are not in Situation 1.

We will now check whether we are in Situation 3. We will first check whether we can make a constraint for $E4$: can we satisfy the constraint if $E4$ is maximised while the rest of the right-hand side of the equation is minimised? Again using Lemma 4.3 we get $\frac{0.4}{0.2} \cdot \frac{0.2}{0.8} = 0.5 < \frac{0.5}{1-0.5} = 1$ which still satisfies the constraint. This means we are not in Situation 4 as we found a way to satisfy the constraint and we can currently not create a constraint for $E4$.

We will now check whether we can create a constraint for $E3$ by checking if the constraint can be satisfied using all degenerate distributions for $E3$: if V_{e3} is False with probability 1, equation 7.14 becomes: $\frac{0.2}{0.4} \cdot \frac{0.6}{0.4} = 0.75 < \frac{0.5}{1-0.5} = 1$ which also satisfies the constraint.

If $E3$ is True with probability 1, the equation becomes $\frac{0.2}{0.4} \cdot \frac{0.2}{0.8} = 0.125 < \frac{0.5}{1-0.5} = 1$ which also satisfies the constraint. This means we cannot currently create a constraint for $E3$.

Checking whether we can create a constraint for $I1$ gives equations $\frac{0.2}{0.4} \cdot \frac{0.2}{0.8} = 0.125 < \frac{0.5}{1-0.5} = 1$ and $\frac{0.2}{0.4} \cdot \frac{0.6}{0.4} = 0.75 < \frac{0.5}{1-0.5} = 1$ both of which satisfy the constraint. This means we are not in Situation 3 as we also cannot create a constraint for $I1$ currently.

Since we are not in Situation 1, 3 or 4, we must be in Situation 2. This means we should use Pearl's algorithm to get the return values from one of the subgraphs that would be disconnected when we remove an arc to one of our neighbours. As it is generally beneficial to do this for small subgraphs, we will do this for $E4$.

E4 Pearl's algorithm returns two diagnostic parameters for each of two combinations of evidence (so we have to run Pearl's algorithm once for every possible combination of evidence in this subgraph).

Assuming evidence $\{E4 = T\}$ Pearl's algorithm returns $\alpha \cdot \Pr(\boldsymbol{e}^{E4\setminus H}|v_h) = 0.6$ and $\alpha \cdot \Pr(\boldsymbol{e}^{E4\setminus H}|\neg v_h) = 0.8$.

Assuming evidence $\{E4 = F\}$ Pearl's algorithm returns $\alpha \cdot \Pr(\boldsymbol{e}^{E4\setminus H}|v_h) = 0.4$ and $\alpha \cdot \Pr(\boldsymbol{e}^{E4\setminus H}|\neg v_h) = 0.2$.

We now return these two parallel calls back to H. Note that we are still considering abductive explanations of size 0: if both of these parallel calls would end up in Situation 1, we would have an abductive explanation of the empty set (i.e. only π can be predicted by this Bayesian network).

H call 1/2 The first of these parallel calls with evidence $\{E4 = T\}$ simplifies Equation 7.14 to $\frac{0.6}{0.8} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \dfrac{\alpha}{1-\alpha}$. To check whether we are in Situation 1, we will try to violate the constraint by maximising the left-hand side of the equation: $\frac{0.6}{0.8} \cdot \frac{0.8}{0.2} = 3 < \dfrac{0.5}{1-0.5} = 1$ violates the constraint, so we are not in Situation 1.

To determine whether we are in Situation 3 we will test if we can make a new constraint for $E3$ by checking if there exists a degenerate distribution for V_{e3} that makes it impossible to satisfy the constraint: we will minimise the left-hand side of the equation for all combinations of degenerate distributions for $E3$. If $E3 = T$ with probability 1, the minimised left-hand side of Equation 7.14 becomes: $\frac{0.6}{0.8} \cdot \frac{0.2}{0.8} = 0.1875 < \dfrac{0.5}{1-0.5} = 1$ which satisfies the constraint.

If $E3 = F$ with probability 1, the minimised left-hand side of Equation 7.14 becomes: $\frac{0.6}{0.8} \cdot \frac{0.6}{0.4} = 1.125 < \dfrac{0.5}{1-0.5} = 1$ which does not satisfy the constraint. This means we are in Situation 3 and can create a parent constraint for $E3$.

For creating a parent constraint, we will use Equation 7.35. In this case $\gamma = \frac{0.8}{0.6} = 1.33$, which gives us an α of $\frac{1.33}{2.33} = 0.57$. As $\Pr(H = T | E3 = T) \leq \Pr(H = T | E3 = F)$, we must maximise $1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$. This results in $1 - \frac{0.57 - 0.8}{0.2 - 0.8}$ and our new constraint becomes $\Pr(E3 = F | \boldsymbol{e}) < 0.62$.

E3 call 1/2 As $E3$ is an evidence node, there is only one combination of evidence that satisfies the constraint: $\{E3 = T\}$. The evidence set $\{E4 = T, E3 = T\}$ in combination with probability $\Pr(E3 = T | \boldsymbol{e}) = 1$ will be returned to node H. However, this set of evidence considers an abductive explanation of at least size 1: the abductive explanation $E3$, as there is an unterminated call that includes $E3 = T$ and $E4 = F$. We will therefore first consider the other parallel call.

H call 2/2 The other parallel call from $E4$ contains evidence $\{E4 = F\}$ and diagnostic parameters from Pearl's algorithm that fill in Equation 7.14 to become $\frac{0.4}{0.2} \cdot \frac{\Pr(v_x | \boldsymbol{e}^+)}{\Pr(\neg v_x | \boldsymbol{e}^+)} < \dfrac{\alpha}{1 - \alpha}$. We will again test if we are in Situation 1, by maximising the left-hand side of the equation and seeing if we can violate the constraint. This results in equation $\frac{0.4}{0.2} \cdot \frac{0.8}{0.2} = 8 < \dfrac{0.5}{1 - 0.5} = 1$ which violates the constraint, meaning we are not in Situation 1.

To determine whether we are in Situation 3 we will first try to create a constraint for node $E3$ by checking if there exists a distribution for $E3$ that makes it impossible to satisfy the constraint. We will minimise the left-hand side of the equation for all degenerate distributions of $E3$: if $E3 = T$ with probability 1, the minimised left-hand side of Equation 7.14 becomes $\frac{0.4}{0.2} \cdot \frac{0.2}{0.8} = 0.5 < \dfrac{0.5}{1 - 0.5} = 1$ which satisfies the constraint. This means we are not in Situation 4.

If $E3 = F$ with probability 1, the minimised left-hand side of Equation 7.14 becomes $\frac{0.4}{0.2} \cdot \frac{0.6}{0.4} = 3 < \dfrac{0.5}{1 - 0.5} = 1$ which does not satisfy the constraint. This means we can again create a constraint for $E3$ and are in Situation 3.

Filling in Equation 7.35 again with mostly the same values as in call 1, α becomes $\frac{0.5}{1.5} = 0.33$, meaning we get a constraint of $\Pr(E3 = F | \boldsymbol{e}) < 1 - \frac{0.33 - 0.8}{0.2 - 0.8} = 0.22$.

E3 call 2/2 As $E3$ is an evidence node, there is only one combination of evidence that satisfies the constraint: $\{E3 = T\}$. The evidence set $\{E4 = F, E3 = T\}$ in combination with probability $\Pr(E3 = T | \boldsymbol{e}) = 1$ will be returned to node H.

We now know that there cannot be an abductive explanation of size 0, as none of the parallel calls consider $E3 = F$ anymore, meaning no combination of evidence that includes $E3 = F$ can predict π, so all abductive explanations must include $E3 = T$.

H call 1/2 We again arrive at node H with new return values that simplify the equation. The return value from $E3$ reduces $E3$ from the CPT of H and leaves a CPT containing the following conditional probabilities (and by definition the conditional probability of the inverse assignment to H): $\Pr(H = T | I1 = T) = 0.8$ and $\Pr(H = T | I1 = F) = 0.4$. We still have the following equation: $\frac{0.6}{0.8} \cdot \frac{\Pr(v_x | \boldsymbol{e}^+)}{\Pr(\neg v_x | \boldsymbol{e}^+)} < \dfrac{\alpha}{1 - \alpha}$

We will again determine the Situation, starting by testing for Situation 1. To test for Situation 1 we again test whether it is possible to violate the constraint, by maximising the left-hand side of the constraint. This results in $\frac{0.6}{0.8} \cdot \frac{0.6}{0.4} = 1.125 < \dfrac{0.5}{1 - 0.5} = 1$. As this violates the constraint, we are not in Situation 1.

We will again test whether we are in Situation 3 by trying to make constraints for the neighbouring variables. There is only a single neighbour left, $I1$. We will minimise the left-hand side of the equation for all degenerate distributions of $I1$ and try to violate the constraint. If $I1 = T$ with probability 1, the minimised left-hand side of the formula becomes $\frac{0.6}{0.8} \cdot \frac{0.2}{0.8} = 0.1875 < \dfrac{0.5}{1 - 0.5} = 1$ which satisfies the constraint.

If $I1 = F$ with probability 1, the minimised left-hand side of Equation 7.14 becomes: $\frac{0.6}{0.8} \cdot \frac{0.6}{0.4} = 1.125 < \dfrac{0.5}{1 - 0.5} = 1$ which does not satisfy the constraint. This means we are in Situation 3 and can create a parent constraint for $I1$.

For creating a parent constraint, we will use Equation 7.35. In this case $\gamma = \frac{0.8}{0.6} = 1.33$, which gives us an α of $\frac{1.33}{2.33} = 0.57$. As $\Pr(H = T|I1 = T) \leq \Pr(H = T|I1 = F)$, we must maximise $1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$. This results in $1 - \frac{0.57-0.6}{0.2-0.6}$ and our new constraint becomes $\Pr(I1 = F|\boldsymbol{e}^{I1\setminus H}) < 0.925$

H call 2/2 We apply the same summing-out as in the first parallel call, which gives us a CPT containing H): $\Pr(H = T|I1 = T) = 0.8$. We still have the equation $\frac{0.4}{0.2} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \dfrac{\alpha}{1-\alpha}$. Testing if we are in Situation 1 by maximising the left-hand side of this equation gives us $\frac{0.4}{0.2} \cdot \frac{0.6}{0.4} = 3 < \dfrac{0.5}{1-0.5} = 1$. As this violates the constraint we are not in Situation 1.

We will again test whether we are in Situation 3 by trying to make constraints for the neighbouring variables. There is only a single neighbour left, $I1$. We minimise the left-hand side for all degenerate distributions of $I1$ and try to violate the constraint. If $I1 = T$ with probability 1, the minimised left-hand side of the formula becomes $\frac{0.4}{0.2} \cdot \frac{0.2}{0.8} = 0.5 < \dfrac{0.5}{1-0.5} = 1$ which satisfies the constraint.

If $I1 = F$ with probability 1, the minimised left-hand side of Equation 7.14 becomes: $\frac{0.4}{0.2} \cdot \frac{0.6}{0.4} = 3 < \dfrac{0.5}{1-0.5} = 1$ which does not satisfy the constraint. This means we are in Situation 3 and can create a parent constraint for $I1$.

For creating a parent constraint, we will use Equation 7.35. In this case $\gamma = \frac{0.2}{0.4} = 0.5$, which gives us an α of $\frac{0.5}{1.5} = 0.33$. As $\Pr(H = T|I1 = T) \leq \Pr(H = T|I1 = F)$, we must maximise $1 - \frac{\alpha - f2(\boldsymbol{e})}{f1(\boldsymbol{e}) - f2(\boldsymbol{e})}$. This results in $1 - \frac{0.33-0.6}{0.2-0.6}$ and our new constraint becomes $\Pr(I1 = F|\boldsymbol{e}^{I1\setminus H}) < 0.325$.

I1 call 1/2 Parallel call 1 has constraint $\Pr(I1 = F|\boldsymbol{e}^{I1\setminus H}) < 0.925$. Equation 7.14 gives us $\frac{\Pr(\boldsymbol{e}^{-\setminus H}|v_x)}{\Pr(\boldsymbol{e}^{-\setminus H}|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \dfrac{0.925}{1-0.925} = 12.3$.

Again, we first test for Situation 1 by maximising the left-hand side of the constraint and checking whether this violates the constraint. As $I1$ only has a single child, we can ignore the term $\boldsymbol{e}^{-\setminus H}$ in the equation. Looking at the CPT of $I1$ we find an upper bound of $\frac{0.7}{0.3} = 2.33$ for $\frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)}$. Filling in the Equation 7.14 gives us $2.33 < \dfrac{0.925}{1-0.925} = 12.3$. As this does not violate the constraint, we are in Situation 1.

This means all combinations of evidence that contain $\{E4 = T, E3 = T\}$ will predict π. This tells us that there exists an abductive explanation of size at most 2. However, there might still exist an abductive explanation of $\{E3 = T\}$, depending on whether the other parallel call also ends up in Situation 1 with all its branching calls. This is the end of this parallel call, as we do not need any return value.

I1 call 2/2 Parallel call 2 has constraint $\Pr(I1 = F|\boldsymbol{e}^{I1\setminus H}) < 0.325$. Equation 7.14 gives us $\frac{\Pr(\boldsymbol{e}^{-\setminus H}|v_x)}{\Pr(\boldsymbol{e}^{-\setminus H}|\neg v_x)} \cdot \frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)} < \dfrac{0.325}{1-0.325} = 0.48$.

Again, we first test for Situation 1 by maximising the left-hand side of the constraint and checking whether this violates the constraint. Looking at the CPT of $I1$ gives us an upper bound of $\frac{0.7}{0.3} = 2.33$ for $\frac{\Pr(v_x|\boldsymbol{e}^+)}{\Pr(\neg v_x|\boldsymbol{e}^+)}$. Filling in Equation 7.14 gives us $2.33 < \dfrac{0.325}{1-0.325} = 0.48$. As this violates the constraint, we are not in Situation 1.

We will now try to create a constraint for $E2$ to help determine whether we are in Situation 3. We minimise the left-hand side for the degenerate distributions of $E2$. If $I2 = T$ with probability 1, Equation 7.14 becomes $2 \cdot \frac{0.6}{0.4} = 1.5 < \dfrac{0.325}{1-0.325} = 0.48$ which does not satisfy the constraint.

If $I2 = F$ with probability 1, Equation 7.14 becomes $2 \cdot \frac{0.4}{0.6} = 0.66 < \dfrac{0.325}{1-0.325} = 0.48$ which does not satisfy the constraint. This means we are in Situation 4: no combination of evidence that includes $\{E4 = T, E3 = F\}$ will satisfy the constraint. This parallel call ends here.

The minimum abductive explanation we found was $\{E4 = T, E3 = T\}$, as it is the smallest subset of $\boldsymbol{e}$ for which all assignments occur a completed parallel call, without the parallel call containing assignments that invalidate the abductive explanation (see Section7.5). In this case, the abductive explanation is equal to the assignments of the only call that ended in Situation 1.

7.3 Running time

In this section we will analyze the running time of the algorithm. We will also compare it to running Pearl's algorithm for every combination of evidence, as that is the most straightforward alternative way to get the solution.

7.3.1 Time per node

Let us start by determining the computation per node per (parallel) call. The main computations in every node consist of finding the Situation and possibly calculating a new constraint. This takes $O(2^{\boldsymbol{\rho}_{V_x}} \cdot \boldsymbol{\rho}_{V_x} + 2^{\boldsymbol{\rho}_{V_s}} \cdot \boldsymbol{\sigma}_{V_x}) = O(2^{\boldsymbol{\rho}_{max}} \cdot D)$ calculations as we saw in Section 7.2.2.2, where D is a bound on the degree all nodes, and $\boldsymbol{\rho}_{max}$ is a bound on the number of parents a node has. In contrast, Pearl's algorithm combines values for every combination of parent assignments, but only calculates two values per child. It also has to create messages for all its parents, giving it a computation time of $O(2^{\boldsymbol{\rho}_{V_x}} \cdot \boldsymbol{\rho}_{V_x})$ per call per node. Note that while the weakest-bound calculations (see Section 7.4.2) are able to limit the number of constraints for each CPT to two, more parallel calls can still mean that we visit a node multiple times. This means each node can be visited for each combination of evidence in its relevant subgraph (the entire graph minus the lower graph or upper graph of the neighbour that created the constraint for V_x). We call the number of relevant evidence nodes $|er|$. For every recursive call, we have to find situations and calculate a new constraint. In total, the computations for a single node take $O(2^{|er|} \cdot 2^{\boldsymbol{\rho}_{max}} \cdot D)$ time. For reference, combining the computation time of Pearl's algorithm for a single node would result in $O(2^{|\boldsymbol{e}|} \cdot 2^{\boldsymbol{\rho}_{V_x}} \cdot \boldsymbol{\rho}_{V_x} \cdot \boldsymbol{\rho}_{V_x})$ (once for each combination of evidence).

7.3.2 Total computation time

However, this computation per node only works if we actually use the algorithm for this node. In Situations 1 and 2 we apply Pearl's algorithm to a portion of the graph instead. This means this portion of the graph takes $O(2^{|er|} \cdot 2^{\boldsymbol{\rho}_{max}} \cdot \boldsymbol{\rho}_{max})$ time per node, where $|er|$ is again the number of relevant evidence nodes. If we would be able to run the entire algorithm without encountering Situation 1 or 2, the total worst-case computation time would be $O(N \cdot 2^{\boldsymbol{\rho}_{max}+|et|} \cdot D)$. When encountering Situation 1 or 2, we use Pearl's algorithm for a part of the graph, which does not change the complexity. Note that the worst-case time complexity of running Pearl's algorithm for all combinations of evidence ($O(N \cdot 2^{|\boldsymbol{E}|} \cdot 2^{\boldsymbol{\rho}_{max}} \cdot \boldsymbol{\rho}_{max})$) is slightly faster than the worst-case time complexity for the constraint propagation algorithm. However, determining the state of a single variable halves the total computation time (the number of possible combinations of evidence gets halved). Additionally, the time it takes per node in the constraint propagation algorithm depends on the number of relevant evidence nodes, while Pearl's algorithm is run for all combinations of evidence for all nodes. We expect that in practice the algorithm is much faster than running Pearl's algorithm $2^{|\boldsymbol{E}|}$ times, especially if the size of the result is small (few combinations of evidence will actually satisfy the constraint, so there will be relatively few parallel calls). Consider the example from Section 7.1.5: here we can manually run the algorithm, where running Pearl's algorithm $2^6 = 64$ times would take much more time. Of course this example is not necessarily representative for the structure of Bayesian networks in practice, but it illustrates the improvement in calculation time that is possible using this algorithm. It would be interesting to implement the algorithm and experimentally compare its speed against alternative calculation methods such as multiple calls of Pearl's algorithm or the Lauritzen and Spiegelhalter [30] algorithm.

7.4 Possible extensions

There are two different types of extensions to the algorithm we will discuss here: simple changes to the algorithm that may reduce computation time or make the algorithm applicable in different situations, and generalisations of the algorithm to more general networks, but are not discussed in this thesis and might be interesting for further research.

7.4.1 Alterations

Weakest-bound calculations As the algorithm propagates its constraints through the graph, it splits into more and more parallel calls. All of these calls have a constraint and will return all combinations of evidence that satisfy it, combined with the return value(s) obtained with this combination of evidence. This means that the result of a stronger constraint will always be a subset of the combinations of evidence that were obtained with the original (weaker) bound. This means we only have to calculate all results for the loosest constraint on a node, after which we can check for each of the resulting values whether it also satisfies the other constraints. In this case, we want to continue all parallel calls until they have created constraints for the same node, instead of completing one call entirely before moving on to the next one.

Search around known evidence In many practical applications one might not want to know all possible combinations of evidence that satisfy a constraint, but all combinations of evidence that have fewer than x changes from a certain combination of evidence E. An example for this would be if we want to know whether there exists a contrastive explanation of size at most x. This can be done by keeping track of how many assignments to evidence have changed compared to E inside each parallel call of the algorithm. If the parallel call would return a combination of evidence that has more than x assignments that are different than in E, these are removed from the return set. When in Situation 1 or 2, we also do not have to try all combinations of evidence, but merely the combinations that have less changes than are still allowed given the current changes. This alteration does not combine well with the weakest-bound calculations, as a stronger constraint may have fewer variables that differ from $\boldsymbol{e}$ and may therefore have results that the weaker constraint does not have. If multiple constraints on a node have the same number of changeable evidence nodes left, we can still calculate the result for the weakest constraint.

Early-stop In Situation 1 the algorithm normally has to return the resulting values of the node for all combinations of evidence. This information will be used to remove nodes from the calculations. However, if this is the last branch of the algorithm we do not need this information and can immediately return the evidence combinations. If the actual values for the hypothesis node that satisfy the constraint are also requested, this is of course not possible: in some situations it may be useful to not only know which combinations of evidence would satisfy the constraint, but also which result all of these combinations of evidence have on the probabilities of the hypothesis node, though this is, for example, not required for finding abductive or contrastive explanations.

Heuristic propagation The algorithm often has to choose to which neighbour to continue: in Situations 2 and 3, there are often multiple directions in which it can continue. We can create heuristics that can help determine how the algorithm could continue to need as little computation time as possible. For example in Situation 2 it is generally beneficial to continue to the tree with the fewest evidence nodes. This costs the least amount of time while hopefully giving us enough information to change the Situation in the node. In Situation 3 it is also beneficial to visit small subgraphs first, as this tightens the constraint for the larger subgraph, which hopefully lowers the number of parallel calls in the larger subgraph, thus improving the chances to get to Situation 1 or 4. The time it takes to calculate the size of subraphs is negligible compared to the total running time of the algorithm, so we expect that this heuristic will improve the total running time. We will leave more complex heuristics as a subject for further research.

Problem flipping The algorithm can find all possible combinations of evidence that satisfy the constraint much faster if there are fewer results. If we want to know all combinations of evidence that satisfy a constraint $\Pr(v_h|\boldsymbol{e}) < \alpha$ which we know is satisfied by the large majority of all possible combinations of evidence, we may use the constraint $\Pr(\neg v_h|\boldsymbol{e}) < 1 - \alpha$ instead. This gives us the set of all combinations of evidence that do not satisfy the constraint, which we can easily convert to the solution of the original problem by subtracting the result from the set of all possible combinations of evidence.

7.4.2 Suggestions

All singly connected graphs The most obvious limitation of the constraint propagation algorithm in its current form is that it only works for some highly constrained graphs. For at least generalizing the algorithm to work in all singly connected graphs, the algorithm would need to be able to transform existing child constraints to new parent constraints. Though the algorithm is in its current state not applicable to many Bayesian networks that are used in practice, it should be able to serve as a proof of concept for an algorithm that is able to effectively find combinations of assignments to nodes that satisfy a condition of some node.

Non-binary networks The current algorithm only works for Bayesian networks with only binary nodes. We can convert Bayesian networks with nodes with any number of states to binary Bayesian networks by representing any n-ary node by a chain of $n-1$ binary nodes, that each have the original parents and children. However, it may be more efficient to generalize the algorithm to work with non-binary nodes and thus get rid of Assumption 7.

Non-singly-connected graphs The current algorithm only works for singly connected graphs. It would be very useful to generalise the algorithm so that it does not need Assumption 6 and works with graphs that are not singly-connected. For example, Pearl's algorithm can be used to approximate probabilities in any Bayesian network by essentially propagating messages around until the network converges (which does not always happen) [35]. There might be a similar thing that can be done for the constraint propagation algorithm, where the algorithm has constraints that converge to a certain point until we know whether we can satisfy them. However, this might have edge-cases where it continues indefinitely. Additionally, we no longer have the same disconnected upper and lower graphs that are used in the algorithm, so some alterations will be required.

Another way would be to convert the network to a singly connected graph by removing loops using loop cutsets. However, this is also hard to do in combination with the constraint propagation algorithm, as the algorithm would then find possible allocations of loop cutset evidence that would satisfy the constraint. However, this evidence is meaningless and does not help us with finding possible assignments of the original evidence in any obvious way.

Other types of constraints Similarly we could remove Assumption 3: if the algorithm could be generalized to accept constraints that have a different form, the algorithm could be used for other interesting applications. For example, if the algorithm would accept multiple constraints at the same time, we could look for evidence assignments that have almost the same prediction as our current prediction of $\Pr(v_x) = y$, by having the constraints $\Pr(v_x) < y + \epsilon$ and $\Pr(v_x) < y - \epsilon$ for some small ϵ. As this would likely produce a small output, the expected calculation time is also much lower. We expect that this addition to the algorithm would be relatively simple to create. We could also attempt to work with constraints of any form, though that is likely to be harder if at all possible. It would also be interesting to see if the algorithm can be altered to allow simultaneous constraints on multiple evidence nodes (removing Assumption 2).

Reverse Pearl's While this algorithm propagates constraints, it works very similarly to directly reversing inference (i.e. finding a combination of evidence that results in an exact distribution of the hypothesis node). By using almost the same formulas but equalities instead of inequalities, it should be possible to change the algorithm to find out which combination(s) of evidence could have been entered to result in this specific prediction. However, this has little real-world meaning as this only works for the exact predictions of $\Pr(v_h|\boldsymbol{e})$ that the Bayesian network can output. For explainability reasons however, this might be useful to do.

7.5 Applications

In this section we will discuss the possible applications of the algorithm.

7.5.1 Contrastive and abductive explanations

The original intention behind the constraint propagation algorithm is to be able to calculate contrastive and abductive explanations more efficiently. A straightforward method would be to find all combinations of evidence that predict π, and determining the abductive and contrastive explanations from there. However, by slightly modifying the algorithm we can do this more efficiently.

Note that a parallel call including assignment to evidence $\boldsymbol{a}$ that is completed guarantees that $\boldsymbol{a} \vDash x$ where x is the starting constraint of the algorithm. However, the reverse is not true. If $\boldsymbol{a} \vDash x$ (i.e. all combinations of evidence that include $\boldsymbol{a}$ satisfy constraint x), all assignments in $\boldsymbol{a}$ end in Situation 1, but not necessarily in the same parallel call. We might for example have the following completed parallel calls: $\{\boldsymbol{a} \wedge A = T\}$, $\{\boldsymbol{a} \wedge A = F \wedge B = T\}$, $\{\boldsymbol{a} \wedge A = F \wedge B = F\}$. Together, this still means that $\boldsymbol{a} \vDash x$, but this information is only contained in the total set of completed calls.

Minimum contrastive explanations Finding a minimum contrastive explanation is straightforward: if we run the constraint propagation algorithm with the constraint that $\neg\pi$ is predicted and a parallel call is completed (reaches Situation 1) with a set of assignments to evidence $\boldsymbol{p}$, this means that $\boldsymbol{p} \vDash \neg\pi$. The assignments that are different from $\boldsymbol{e}$ correspond to a potential contrastive explanation: there exists an assignment to those variables that is different from $\boldsymbol{e}$ and guarantees the prediction of $\neg\pi$. As all potential contrastive explanations occur in some branch of the algorithm, we are only interested in the parallel calls where this set is minimal.

The parallel call that reaches Situation 1 with fewest evidence-assignments that are different from $\boldsymbol{e}$ corresponds to a contrastive explanation that includes all evidence-assignments in that parallel call that are different from $\boldsymbol{e}$. To find these parallel calls efficiently, we can pause all parallel calls that do not have the fewest evidence assignments that are different from $\boldsymbol{e}$ and end the algorithm when the first parallel call is completed.

All contrastive explanations Conversely, for finding all contrastive explanations we can pause all parallel calls which have the following properties:

1. Given the set $\boldsymbol{C}$ of variables for which the evidence assignment in this parallel call is different from $\boldsymbol{e}$, there exists another parallel call where variables $\boldsymbol{C'} \subset \boldsymbol{C}$ have a different assignment than in $\boldsymbol{e}$.
2. The parallel call with $\boldsymbol{C'}$ has either not terminated or ended in Situation 1.

If all parallel calls are paused, the algorithm can terminate.

Minimum abductive explanations Abductive explanations are slightly more complex: we now want the minimal set for which $\boldsymbol{p} \vDash \pi$. If we run the constraint propagation algorithm with the constraint that π is predicted and a branch completes with a set of assignments $\boldsymbol{p}$, this corresponds to an abductive explanation $\boldsymbol{A}$ iff all variables in $\boldsymbol{A}$ occur in $\boldsymbol{p}$ with the same assignment as in $\boldsymbol{e}$ and all other variables either

- do not occur in $\boldsymbol{e}$
- occur in $\boldsymbol{e}$, but also occur in another completed parallel call with opposite assignment, that also contains all variables in $\boldsymbol{A}$, and for which this same property holds.

See also the examples earlier in this subsection.

To find a minimum abductive explanation, for each parallel call we keep track of the minimal size of abductive explanation for which these properties hold or could hold depending on unterminated calls, and pause all parallel calls for which this size is larger than the minimum one. We return the first abductive explanation that we confirm, and terminate the algorithm.

All abductive explanations Finding all abductive explanations works extremely similar, but we terminate all parallel calls that only correspond to abductive explanations for which a subset is already a confirmed abductive explanation, and do not stop the algorithm after finding the first abductive explanation. See also example 2 in Section 7.2.8.

7.5.2 Improve understanding of Bayesian networks

Another use of this algorithm is to be able to examine a Bayesian network if it produces unexpected results. It is likely able to get a lot of evidence-output pairs much faster than Pearl's algorithm is able to, which is beneficial if there is ever a reason to turn a Bayesian network inside-out to determine what went wrong.

Algorithm 10 ConstraintPropagationSCG(Node V_n, constraint α)

```
returnSet = $\emptyset$;
if $V_n$ == evidenceNode and constraint.type == Parent then
    value = $V_n$.valueThatSatisfies($\alpha$);
    returnSet.add($V_n$ = value, Pr($value$) = 1);
    return returnSet;
end if
if $V_n$ == evidenceNode and constraint.type == Child then
    for State $v_s \in V_n$ do
        if $v_s$.satisfies($\alpha$) then
            retVal = calculateReturnValue($\alpha$, $v_s$)
            returnSet.add($V_n = v_s$, retVal);
            return returnSet;
        end if
    end for
end if
situation = checkSituation($V_n$, $\alpha$);
if situation == 1 then
    for evidence in AllSubgraphEvidenceCombinations($V_n$) do
        returnValues = DetermineReturnValues($V_n$, evidence);
        returnSet.add(evidence, returnValues);
    end for
    return returnSet;
end if
if situation == 2 then
    $V_p$ = $V_n$.Parents[0];
    for evidence in AllSubgraphEvidenceCombinations($V_p$) do
        returnValues = DetermineReturnValues($V_p$, evidence);
        $V_n$.SimplifyCalculation($V_p$, returnValues);
        recursiveResults = ConstraintPropagationSCG($V_n$, $\alpha$);
        for (evs, val) in recursiveResults do
            returnSet.add(evs+evidence, val);
        end for
    end for
    return returnSet;
end if
if situation == 3 then
    ($V_p$, $\beta$) = CalculateNewConstraint($\alpha$);
    neighbourResults = ConstraintPropagationSCG($V_p$, $\beta$);
    for (evidence, value) in neighbourResults do
        $V_n$.SimplifyCalculation($V_p$, value);
        recursiveResults = ConstraintPropagationSCG($V_n$, $\alpha$);
        for (evs, val) in recursiveResults do
            returnSet.add(evs+evidence, val);
        end for
    end for
end if
if situation == 4 then
    return $\emptyset$;
end if
```

Chapter 8

Conclusion

In Chapter 2 we saw that there are several types of explanations that can easily be computed for Bayesian networks and Bayesian classifiers, such as chains of reasoning, some measures of important internal nodes, and explanations of nodes and links. However, certain types of explanations are extremely expensive to compute for Bayesian networks of larger size, such as abduction, Shapley values, or MAP-independence. Two of these types of explanations that are expensive to compute are abductive and contrastive explanations. We proved in Chapters 5 and 6 that it is likely not possible (impossible assuming $P \neq NP$) to compute abductive or contrasive explanations in polynomial time relative to the size of the Bayesian network. However, finding these explanations has a different complexity depending on which subproblem you want to solve: for example whether you want to check whether something is a contrastive explanation or if you want to find a minimal contrastive explanation. Additionally, knowing all abductive explanations greatly helps with finding all contrastive explanations and vice versa. The complexities found in Chapters 5 and 6 are not exhaustive: for some subproblems we have not determined whether they can be solved in polynomial time, and for many subproblems we have only shown hardness and not completeness for a complexity class, which means they may be in an even harder complexity class. In these chapters we have also introduced some simple algorithms that solve the introduced subproblems. These simple algorithms can be used for any type of classifier, and do not look at the internal structure of the (Bayesian) classifier, but use inference to obtain individual predictions from the Bayesian classifier. These algorithms provide upper bounds for the running time needed to solve these problems, but it is likely possible to create algorithms that take less time to solve the subproblems. In Chapter 7, we introduced a more complicated algorithm to find all abductive or contrastive explanations. This algorithm is tailored to Bayesian classifiers specifically and uses a branch-and-bound-like strategy of traversing the network to minimise the computations required to find all abductive or contrastive explanations. The proposed algorithm solves a more general query of finding all evidence assignments that would satisfy a specific type of constraint, which can then be used to compute abductive or contrastive explanations. The algorithm should in practice take less time than existing alternatives, because it is designed to minimise calculating the same partial results multiple times. Additionally, it only takes the parts of the network into account that could influence the result of the algorithm. The (constraint propagation) algorithm so far works for specific types of graphs only: it is constrained to a singly connected binary Bayesian network, where some nodes may have at most one parent. The most prohibitive limitation of the algorithm currently is that it only works for these specific networks. In Section 7.4.2 we have suggested numerous additions to the algorithm to make it run faster or able to deal with more types of graphs, some of which are left as subject for further research.

Bibliography

[1] Arnborg, S., Corneil, D. G., and Proskurowski, A. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic Discrete Methods 8*, 2 (1987), 277–284.

[2] Arora, S., and Barak, B. *Computational Complexity - A Modern Approach.* Cambridge University Press, 2009.

[3] Beck, A., Bleicher, M. N., and Crowe, D. W. *Excursions into Mathematics: The Millennium Edition.* CRC Press, 2000.

[4] Cheng, J., Greiner, R., Kelly, J., Bell, D., and Liu, W. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence 137*, 1 (2002), 43–90.

[5] Choi, A., Xue, Y., and Darwiche, A. Same-decision probability: A confidence measure for threshold-based decisions. *International Journal of Approximate Reasoning 53*, 9 (2012), 1415–1428.

[6] Cooper, G. F. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence 42*, 2 (1990), 393–405.

[7] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to algorithms, fourth edition.* MIT press, 2022.

[8] Dawid, A. P. Beware of the DAG! In *Proceedings of Workshop on Causality: Objectives and Assessment at NIPS 2008* (Whistler, Canada, 12 Dec 2010), I. Guyon, D. Janzing, and B. Schölkopf, Eds., vol. 6 of *Proceedings of Machine Learning Research*, PMLR, pp. 59–86.

[9] Derks, I. P., and de Waal, A. A taxonomy of explainable Bayesian networks. In *Artificial Intelligence Research* (Cham, 2020), A. Gerber, Ed., Springer International Publishing, pp. 220–235.

[10] Dhurandhar, A., Pedapati, T., Balakrishnan, A., Chen, P.-Y., Shanmugam, K., and Puri, R. Model agnostic contrastive explanations for structured data. *arXiv preprint arXiv:1906.00117* (2019).

[11] Diestel, R. *Graph Theory*, fifth ed., vol. 173 of *Graduate Texts in Mathematics.* Springer, Heidelberg; New York, 2017, pp. 355–382.

[12] Druzdzel, M. J. *Probabilistic reasoning in decision support systems: from computation to common sense.* Carnegie Mellon University, 1993. PhD thesis.

[13] Druzdzel, M. J. SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference* (1999), AAAI '99/IAAI '99, p. 902–903.

[14] Díez, F., Mira, J., Iturralde, E., and Zubillaga, S. DIAVAL, a Bayesian expert system for echocardiography. *Artificial Intelligence in Medicine 10*, 1 (1997), 59–73.

[15] Edmonds, J., and Karp, R. M. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM) 19*, 2 (apr 1972), 248–264.

[16] Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[17] Guo, H., Boddhireddy, P. R., and Hsu, W. H. An ACO algorithm for the most probable explanation problem. In *AI 2004: Advances in Artificial Intelligence* (Berlin, Heidelberg, 2005), G. I. Webb and X. Yu, Eds., Springer Berlin Heidelberg, pp. 778–790.

[18] Haddawy, P., Jacobson, J., and Kahn, C. E. BANTER: a Bayesian network tutoring shell. *Artificial Intelligence in Medicine 10*, 2 (1997), 177–200.

[19] Ignatiev, A., Narodytska, N., Asher, N., and Marques-Silva, J. From contrastive to abductive explanations and back again. In *AIxIA 2020 – Advances in Artificial Intelligence* (Cham, 2021), M. Baldoni and S. Bandini, Eds., Springer International Publishing, pp. 335–355.

[20] Jensen, F. V. *Causal and Bayesian Networks.* Springer New York, New York, NY, 2001, pp. 3–34.

[21] Koopman, T. Computing contrastive, counterfactual explanations for Bayesian networks. Master's thesis, Utrecht University, 2020.

[22] Koopman, T., and Renooij, S. Persuasive contrastive explanations for Bayesian networks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (Cham, 2021), J. Vejnarová and N. Wilson, Eds., Springer International Publishing, pp. 229–242.

[23] Kwisthout, J. Most probable explanations in Bayesian networks: Complexity and tractability. *International Journal of Approximate Reasoning 52*, 9 (2011), 1452–1469.

[24] Kwisthout, J. Explainable AI using MAP-independence. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (Cham, 2021), J. Vejnarová and N. Wilson, Eds., Springer International Publishing, pp. 243–254.

[25] Kwisthout, J. H. P., Bodlaender, H. L., and van der Gaag, L. C. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence* (2010), p. 237–242.

[26] Kyrimi, E. *Bayesian Networks for Clinical Decision Making: Support, Assurance, Trust.* PhD thesis, Queen Mary University of London, 2019.

[27] Kyrimi, E., and Marsh, W. A progressive explanation of inference in 'hybrid' Bayesian networks for supporting clinical decision making. In *Proceedings of the Eighth International Conference on Probabilistic Graphical Models* (Lugano, Switzerland, 06–09 Sep 2016), A. Antonucci, G. Corani, and C. P. Campos, Eds., vol. 52 of *Proceedings of Machine Learning Research*, PMLR, pp. 275–286.

[28] Lacave, C., and Diez, F. J. Explanation for causal Bayesian networks in Elvira. In *Proceedings of the Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-2002)* (2002), Citeseer, pp. 43–48.

[29] Lacave, C., Oniśko, A., and Díez, F. J. Use of Elvira's explanation facility for debugging probabilistic expert systems. *Knowledge-Based Systems 19*, 8 (2006), 730–738.

[30] Lauritzen, S. L., and Spiegelhalter, D. J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological) 50*, 2 (1988), 157–194.

[31] Leersum, J. v. Explaining the reasoning of Bayesian networks with intermediate nodes and clusters. Master's thesis, Utrecht University, 2015.

[32] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), NIPS'17, p. 4768–4777.

[33] Madsen, A. L., Lang, M., Kjærulff, U. B., and Jensen, F. The Hugin tool for learning Bayesian networks. In *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (Berlin, Heidelberg, 2003), T. D. Nielsen and N. L. Zhang, Eds., Springer Berlin Heidelberg, pp. 594–605.

[34] Miller, T. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence 267* (2019), 1–38.

[35] Pearl, J. 11 reverend bayes on inference engines: A distributed hierarchical. *Probabilistic and Causal Inference: The Works of Judea Pearl* (1982), 129.

[36] Pourret, O., Na, P., Marcot, B., et al. *Bayesian networks: a practical guide to applications.* John Wiley & Sons, 2008.

[37] Ribeiro, M. T., Singh, S., and Guestrin, C. "Why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2016), KDD '16, Association for Computing Machinery, p. 1135–1144.

[38] Roth, D. On the hardness of approximate reasoning. *Artificial Intelligence 82*, 1 (1996), 273–302.

[39] Shih, A., Choi, A., and Darwiche, A. A symbolic approach to explaining Bayesian network classifiers. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence* (2018), IJCAI'18, p. 5103–5111.

[40] Suermondt, H. J. *Explanation in Bayesian belief networks.* PhD thesis, Stanford University, 1992.

[41] Van den Broeck, G., Lykov, A., Schleich, M., and Suciu, D. On the tractability of SHAP explanations. *Proceedings of the AAAI Conference on Artificial Intelligence 35*, 7 (May 2021), 6505–6513.

[42] Van der Gaag, L. C. On evidence absorption for belief networks. *International Journal of Approximate Reasoning 15*, 3 (1996), 265–286.

[43] Yuan, C., Lu, T.-C., and Druzdzel, M. J. Annealed MAP. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* (Arlington, Virginia, USA, 2004), UAI '04, AUAI Press, p. 628–635.