FRANCISCO HENRIQUE OTTE VIEIRA DE FARIA

LEARNING ACYCLIC PROBABILISTIC LOGIC PROGRAMS FROM DATA

São Paulo
2018

FRANCISCO HENRIQUE OTTE VIEIRA DE FARIA

LEARNING ACYCLIC PROBABILISTIC LOGIC PROGRAMS FROM DATA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2018

FRANCISCO HENRIQUE OTTE VIEIRA DE FARIA

LEARNING ACYCLIC PROBABILISTIC LOGIC PROGRAMS FROM DATA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

Área de Concentração:
Engenharia de Computação

Orientador:
Fabio Gagliardi Cozman

São Paulo
2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

de Faria, Francisco Henrique Otte Vieira
LEARNING ACYCLIC PROBABILISTIC LOGIC PROGRAMS FROM DATA / F. H. O. V. de Faria -- versão corr. -- São Paulo, 2018.
85 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Machine Learning 2.Probabilistic Logic Programming 3.Parameter Learning 4.Structure Learning 5.Acyclic Probabilistic Models I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

A.M.D.G.

ACKNOWLEDGMENTS

This dissertation could not have been completed without the great support that I have received from so many people over the years. I wish to offer my most heartfelt thanks to the following people:

To my advisor, Fabio Gagliardi Cozman. Thank you for the advice, support, and willingness that ignited in me a true passion for the topics of this research project.

To Denis Deratani Mauá. Thank you for all the useful discussions and rich contributions during the whole development of this work.

To Anna Reali Costa. Thank you for keeping your office door open and available for all the times when I needed it.

To Arthur Colombini Gusmão and Bruno Sguerra. We have collaborated on research, discussed philosophies and how to change the world around us. You have been two of my closest colleagues and closest friends in the last few years, and I wish you all the best in your academic, professional and personal pursuits.

To all my colleagues, Diego Bezerra Lira, Émerson Cruz, Fabio Henrique Santana Machado, Glauber De Bona, Thiago Bueno. Thank you all for your inspiration and support.

To my family. You have encouraged my academic interests from day one. Thank you.

To Toshiba Corporation, Japan. Thank you for funding this research project. It is an honor for me to have taken part in such a prestigious scholarship program.

" "What truly is logic? Who decides reason? [...] It is only in the mysterious equations of love that any logic or reason can be found."

-John Nash-

RESUMO

O aprendizado de um programa lógico probabilístico consiste em encontrar um conjunto de regras lógico-probabilísticas que melhor se adequem aos dados, a fim de explicar de que forma estão relacionados os atributos observados e predizer a ocorrência de novas instanciações destes atributos. Neste trabalho focamos em programas acíclicos, cujo significado é bastante claro e fácil de interpretar. Propõe-se que o processo de aprendizado de programas lógicos probabilísticos acíclicos deve ser guiado por funções de avaliação importadas da literatura de aprendizado de redes Bayesianas. Neste trabalho são sugeridas novas técnicas para aprendizado de parâmetros que contribuem para uma melhora significativa na eficiência computacional do estado da arte representado pelo pacote ProbLog. Além disto, apresentamos novas técnicas para aprendizado da estrutura de programas lógicos probabilísticos acíclicos.

Palavras-Chave – Programação Lógica Probabilística, Aprendizado de Máquina, IA Explicável.

ABSTRACT

To learn a probabilistic logic program is to find a set of probabilistic rules that best fits some data, in order to explain how attributes relate to one another and to predict the occurrence of new instantiations of these attributes. In this work, we focus on acyclic programs, because in this case the meaning of the program is quite transparent and easy to grasp. We propose that the learning process for a probabilistic acyclic logic program should be guided by a scoring function imported from the literature on Bayesian network learning. We suggest novel techniques that lead to orders of magnitude improvements in the current state-of-art represented by the ProbLog package. In addition, we present novel techniques for learning the structure of acyclic probabilistic logic programs.

Keywords – Probabilistic Logic Programming, Machine Learning, Explainable AI.

LIST OF FIGURES

LIST OF TABLES

Nomenclature

θ_{ijk}	Parameter approximating the probability that variable i takes its jth value given that its parents take their kth configuration
$\boldsymbol{\theta}$	Set of parameters approximating a probabilities distribution
LL	Log-Likelihood
BIC	Bayesian Information Criterion
ELL	Expected Log-Likelihood
EM	Expectation Maximization
TP	True observations predicted true by a model (true positives)
TN	False observations predicted false by a model (true negatives)
FP	False observations predicted true by a model (false positives)
FN	True observations predicted false by a model (false negatives)
PLP	Probabilistic logic program
AND	Logic and
OR	Logic or
XOR	Exclusive or
A-PLP	Acyclic probabilistic logic program
AP-PLP	Acyclic propositional probabilistic logic program
AP-PLP(2)	Acyclic propositional probabilistic logic program with at most two parents per variable
$\mathbb{P}(X)$	Probability of an event X
$\mathbb{P}(X \mid Y)$	Probability of an event X given an event Y
$\text{PA}[X]$	Parents of variable X
N_{ijk}	Number of observations where variable i takes its jth value and its parents their kth configuration
N_{jk}	Number of observations where the parents of variable i take their kth configuration

CONTENTS

1 INTRODUCTION

Machine learning develops programs that *learn from experience* [39]; that is, computer programs whose performance at a specific task improves with observation or repetition [31]. Machine learning methods can be classified as supervised or unsupervised, depending on whether feedback is provided to correct the program predictions while the program is trained. If, during training, expected outputs are known, the learning task is called *supervised learning*, which is the case of methods proposed in this work. Conversely, in *unsupervised learning*, the expected outputs are not known at all during training.

The *interpretability* of prediction models obtained through machine learning methods has generated substantial discussion recently. Informally, a model is called interpretable if, from a human perspective, it can be explained [14]. Although this definition may look simple, its application is rather subjective, as it deals with how a prediction model and its human user relate to one another. There has also been an increasing interest in explainable models, which would be capable of explaining their own rationale, providing insights into how features relate to one another [6]. Consider, for instance, *deep learning*, a subfield of machine learning that deals with "deep" *neural networks* (deep in the sense they have typically more than three layers of neurons). Deep learning has found tremendous success in some of the grand challenges in artificial intelligence, like *image recognition* and *natural language processing*. However, it is difficult to argue for its interpretability. This trade-off between prediction accuracy and interpretability has been extensively discussed [4,19,24]; it seems fair to say that, whenever a human is responsible for the consequences of model-based decision making, interpretability becomes crucial.

The meaning of *probabilistic logic programs* (abbreviated PLPs) is relatively transparent and easy to grasp. This sort of combination of logic and probabilities has been advocated for quite some time [18, 38] and different semantics have been proposed to specify probability distributions in logic programming. Sato's distribution semantics [41] is based on probabilistic facts, such as

$$0.3 :: \mathsf{neighbor}(X, Y).,$$

which means that there is a 0.3 probability that any X is a neighbor of Y, and deterministic rules as, for instance

$$\mathsf{calls}(X,Y) :- \mathsf{alarm}(X), \mathsf{neighbor}(X,Y)\mathbf{.}, \tag{1.1}$$

meaning that if $\mathsf{alarm}(X)$ and $\mathsf{neighbor}(X,Y)$ are true, then $\mathsf{calls}(X,Y)$ should also be true. This semantics has been extended to handle probabilistic rules such as [13]:

$$0.7 :: \mathsf{calls}(X,Y) :- \mathsf{alarm}(X), \mathsf{neighbor}(X,Y)\mathbf{.}, \tag{1.2}$$

meaning that if $\mathsf{alarm}(X)$ and $\mathsf{neighbor}(X,Y)$ are true, there is a 0.7 probability that $\mathsf{calls}(X,Y)$ is also true.

The goal of this work is to present algorithms that learn *acyclic probabilistic logic programs* (abbreviated A-PLPs) from data, by adapting techniques from Bayesian network learning. Acyclic programs contain probabilistic facts and rules that generate no cycles, and are particularly easy to understand.

We adopt the syntax of the ProbLog package, a state-of-art probabilistic logic programming implementation based on ProbLog semantics [38]. The facts and rules above are written in ProbLog's syntax. The ProbLog package includes a parameter learning algorithm implementing an EM-like algorithm that resorts to BDD diagrams to speed inference whenever possible [15,20]. However, the resulting learning algorithm is rather slow. In this work, we propose novel techniques that lead to orders of magnitude improvements in parameter learning speed, especially when data are complete. These improvements have been described in the paper:

- "Speeding-up ProbLog's Parameter Learning",

accepted for publication in the Proceedings of the Seventh International Workshop on Statistical Relational AI (StarAI 2017) [11]. We also introduce improvements to parameter learning in the presence of missing data, as described in the paper:

- "Parameter Learning in ProbLog with Probabilistic Rules",

which received the Best Paper Award (3rd Place) in the Symposium on Knowledge Discovery, Mining and Learning (KDMiLe 2017) [10].

In this work, we also propose techniques for learning the structure of A-PLPs (that is, the rules, facts and parameters that best explain some data). To do so, we introduce

an exact score-based approach that has not been investigated so far, to the best of our knowledge. These results are described in the paper:

- "Closed-Form Solutions in Learning Probabilistic Logic Programs by Exact Score Maximization",

accepted for publication in the Proceedings of the 11th International Conference on Scalable Uncertainty Management (SUM 2017) [9].

In Chapter 2 we briefly review some important concepts in Bayesian Networks (Section 2.1) and Logic Programs (Section 2.2). In Chapter 3 we present our contributions. In Section 3.1 we derive an algorithm for PLP parameter learning that relies on expectation calculations and exact maximization of polynomial equations. In Section 3.2 we present a structure learning algorithm that uses insights presented in Section 3.2 for local optimization and constraint programming for global optimization. Experiments and results are reported in Chapter 4.

2 BACKGROUND

Acyclic probabilistic logic programs (A-PLPs) and Bayesian networks are very closely related. This is in fact the reason why we decided to adapt techniques from Bayesian network learning to A-PLP learning. In this chapter we review important concepts from Bayesian Networks (Section 2.1) and Logic Programs (Section 2.2). These concepts will set the basis for understanding the problem we address and the techniques developed in this work.

2.1 Bayesian Networks

A Bayesian network consists of a directed acyclic graph, where each node corresponds to a random variable X associated with a set of local conditional probabilities for values of X given values of its parents $\text{PA}[X]$. We call *family* of X the set consisting of a variable and its parents, $\{X\} \cup \text{PA}[X]$. The directed acyclic graph encodes a set of conditional independence assumptions: Any variable X in a Bayesian network is assumed independent of its non-descendants, given its parents $\text{PA}[X]$. This property is known as *Markov Independence Assumption*.

Bayesian networks are powerful knowledge representation tools that allow reasoning under uncertainty, providing a formal language and inference mechanism for deriving uncertain conclusions. One can, for instance, infer the probability of some hypothesis about the values that non-observed variables could take, given some evidence about the values assumed by observed variables.

A joint space is the set of all possible combinations of values $\{x_1, ..., x_n\}$ that a set of variables $\{X_1, ..., X_n\}$ can take. The probability distribution defined over this joint space is called *joint probability distribution* [31]. Due to Markov independence condition, the joint probability of a specific assignment $\{X_1 = x_1, ..., X_n = x_n\}$ of variables in a Bayesian

network is given by the following factorization:

$$\mathbb{P}(x_1, ..., x_n) = \prod_{i=1}^{n} \mathbb{P}(x_i \mid \text{PA}[X_i] = w_i),$$

where values $\mathbb{P}(x_i \mid \text{PA}[X_i] = w_i)$ are obtained from the local conditional probabilities of variable X_i, and where each w_i is consistent with $\{x_1, \ldots, x_n\}$.

Example 1. *Figure 1 depicts a Bayesian network for the classical Alarm example, where an alarm can be triggered either by a burglary or a fire and, depending on whether the alarm has been activated or not, there will be a probability that neighbor John calls. This Bayesian network lets one answer questions like "What is the probability that there has been a burglary given the fact that John called?". The answer is given by Bayes conditional probability formula:*

$$\mathbb{P}(\mathsf{burglary} \mid \mathsf{johnCalls}) = \frac{\mathbb{P}(\mathsf{johnCalls} \mid \mathsf{burglary}).\mathbb{P}(\mathsf{burglary})}{\mathbb{P}(\mathsf{johnCalls})},$$

where:

$$\mathbb{P}(\mathsf{johnCalls}) = \mathbb{P}(\mathsf{johnCalls} \mid \mathsf{burglary}).\mathbb{P}(\mathsf{burglary}) + \mathbb{P}(\mathsf{johnCalls} \mid \overline{\mathsf{burglary}}).\mathbb{P}(\overline{\mathsf{burglary}})$$

$$\begin{aligned}\mathbb{P}(\mathsf{johnCalls} \mid \mathsf{burglary}) = {} & \mathbb{P}(\mathsf{johnCalls} \mid \mathsf{alarm}).\mathbb{P}(\mathsf{alarm} \mid \mathsf{burglary}) \\ & + \mathbb{P}(\mathsf{johnCalls} \mid \overline{\mathsf{alarm}}).\mathbb{P}(\overline{\mathsf{alarm}} \mid \mathsf{burglary})\end{aligned}$$

$$\begin{aligned}\mathbb{P}(\mathsf{johnCalls} \mid \overline{\mathsf{burglary}}) = {} & \mathbb{P}(\mathsf{johnCalls} \mid \mathsf{alarm}).\mathbb{P}(\mathsf{alarm} \mid \overline{\mathsf{burglary}}) \\ & + \mathbb{P}(\mathsf{johnCalls} \mid \overline{\mathsf{alarm}}).\mathbb{P}(\overline{\mathsf{alarm}} \mid \overline{\mathsf{burglary}}).\end{aligned}$$

Local conditional probabilities are usually expressed via *conditional probability tables* (CPTs) that explicitly define the probabilities of a variable X given its parents $\text{PA}[X]$. This is the case of the Bayesian network depicted in Figure 1. A more compact representation is based on *decision graphs*. For instance, suppose we have a set of binary variables $\{X_1, X_2, X_3\}$, where $\text{PA}[X_1] = \{X_2, X_3\}$. The probability distribution of variable X_1 could be encoded via a decision tree like the one illustrated in Figure 2. Note that this graph is *not* a Bayesian network: it is just the representation of a probability distribution *in* a Bayesian network.

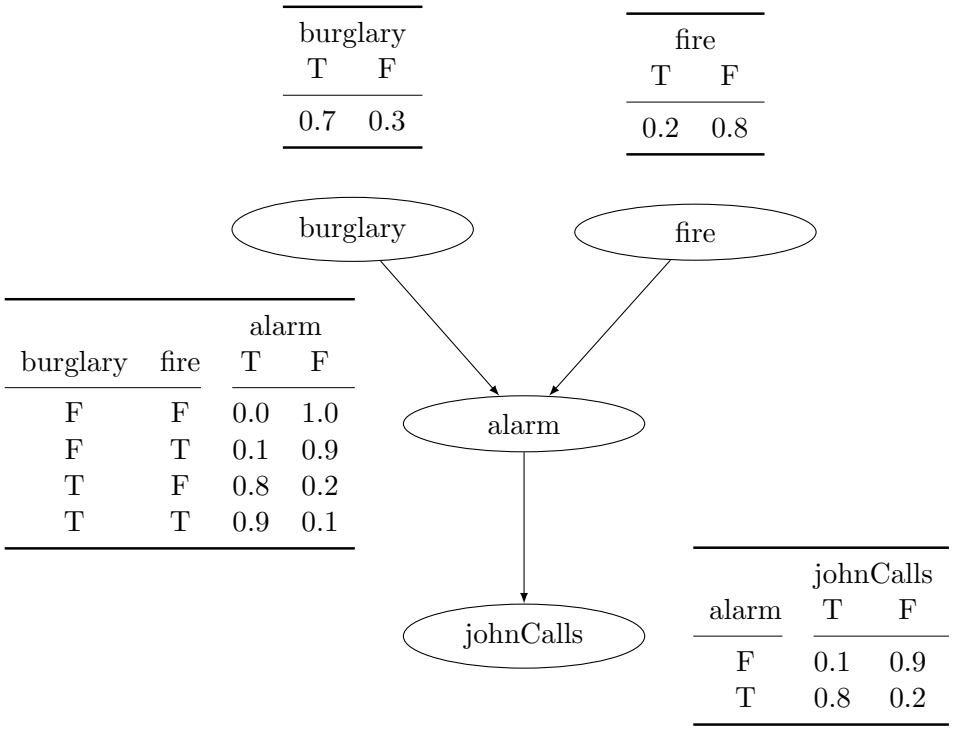


Figure 1: Alarm example Bayesian network.

2.1.1 Scoring Bayesian Networks

Learning a Bayesian network (both its structure and parameters) usually requires comparing the performance of two network candidates. Given a dataset D, which gathers a set of observations on the network variables, there are several ways to score the performance of a Bayesian Network. In this section, we present a very commonly used score: the Bayesian Information Criterion (BIC).

Before formally introducing the BIC score, let us define the *Likelihood* of a Bayesian network B, given a dataset D, as the probability $\mathbb{P}(D \mid B)$ of observing dataset D given the Bayesian network B. Assuming the probability distributions are encoded via flat tables, for a set of observations $d \in D$ we have:

$$L(B, D) = \prod_{d \in D} \prod_{i=1}^{n} \theta_{ijk},$$

where θ_{ijk} is a parameter encoding the probability that variable X_i assumes its jth value once its parents take their kth configuration; that is, $\theta_{ijk} = \mathbb{P}(X_i = x_{ij} \mid \text{PA}[X_i] = w_{ik})$. However, if acyclicity had not been ensured, writing the likelihood expression would not have been that easy.

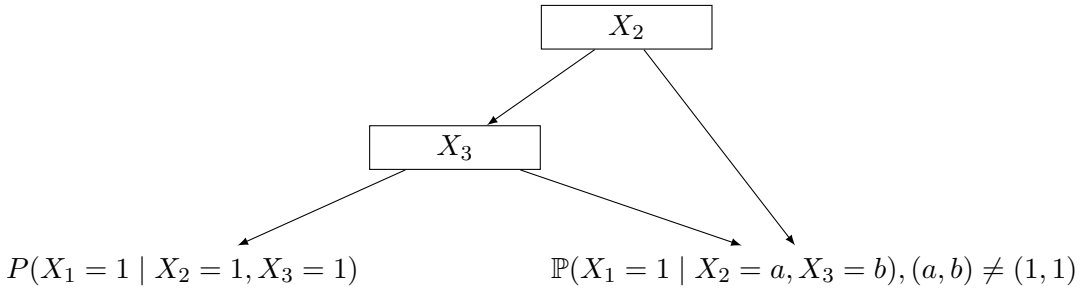


Figure 2: Decision graph encoding the probability distribution of variable X_1 given its parents X_2 and X_3.

Note that, as $\theta_{ijk} \in [0, 1]$, if we have a large dataset to process, we will soon need to handle very small numbers. Therefore, instead of the Likelihood, we usually use the *Log-Likelihood* score:

$$LL(B, D) = \log \prod_{d \in D} \prod_{i=1}^{n} \theta_{ijk}.$$

This Log-Likelihood expression can be rewritten as follows:

$$LL(B, D) = \sum_{i=1}^{n} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk}.log(\theta_{ijk}),$$

where n is the number of variables represented in B, q_i is the number of states for variable X_i, r_i is the number of configurations the parents of a variable X_i can assume and N_{ijk} corresponds to the number of observations where $X_i = x_{ij}$ and $\text{PA}[X_i] = w_{ik}$ in dataset D. Notice the Log-Likelihood is *decomposable*, as it can be computed locally and then summed for all subgraphs of a graphical model. This is an important property, which is the basis of most learning algorithms for graphical models.

A major concern in machine learning is *overfitting*. Overfitting refers to a situation where a statistical model fits very well to training data, but is not capable of generalizing and predicting new data. The Bayesian Information Criterion adds to the Log-Likelihood a penalization over the number of parameters $\mid \Theta \mid$ needed to represent the probabilities distribution:

$$BIC(B, D) = LL(B, D) - \frac{1}{2} ln(N) \mid \Theta \mid .$$

When local probabilities distributions are expressed via complete CPT's, there is a different parameter θ for each possible combination the parents of a variable can assume. The penalization favors models with fewer parameters, hence fewer parents per variable. This is a common strategy used to avoid overfitting in learning tasks, as suggested by Occam's razor principle: Prefer the simplest hypothesis that fits the data [31].

2.1.2 Parameter Learning

Learning a Bayesian Network requires building a dependency graph over variables $\mathbf{X}$ and describing the probability distribution of each variable X given its parents $\text{PA}[X]$. In parameter learning, the dependency graph is already known and we just need to learn the parameters defining the local probability distribution for each family. In this section, we present the *Maximum Likelihood* approach for estimating the parameters of a Bayesian network and the *Expectation Maximization* algorithm, which can be used to maximize the Log-likelihood of a Bayesian Network.

Maximum Likelihood The *Maximum Likelihood* approach is to find the model parameters that maximize the likelihood value. When the local probability distributions are expressed via flat tables in a Bayesian network, we can easily determine the value of θ_{ijk} that maximizes the likelihood by simply counting the occurrences of each family configuration in the training set.

$$\underset{\theta_{ijk}}{\text{argmax}}\, L(B \mid D) = \frac{N_{ijk}}{N_{ik}}.$$

Example 2. *Table 1 lists a set of three fully observed examples for the Bayesian network of Figure 1. For the observations in Table 1, the likelihood expression would be written as:*

$$\begin{aligned} L(B, D) = &\mathbb{P}(\mathsf{burglary}, \overline{\mathsf{fire}}, \mathsf{alarm}, \mathsf{johnCalls}). \\ &\mathbb{P}(\overline{\mathsf{burglary}}, \overline{\mathsf{fire}}, \overline{\mathsf{alarm}}, \overline{\mathsf{johnCalls}}). \\ &\mathbb{P}(\overline{\mathsf{burglary}}, \mathsf{fire}, \mathsf{alarm}, \mathsf{johnCalls}), \end{aligned}$$

and due to Markov independence assumption, factorized as:

$$\begin{aligned} L(B, D) = &\mathbb{P}(\mathsf{johnCalls} \mid \mathsf{alarm}).\mathbb{P}(\mathsf{alarm} \mid \mathsf{burglary}, \overline{\mathsf{fire}}).\mathbb{P}(\mathsf{burglary}).\mathbb{P}(\overline{\mathsf{fire}}). \\ &\mathbb{P}(\overline{\mathsf{johnCalls}} \mid \overline{\mathsf{alarm}}).\mathbb{P}(\overline{\mathsf{alarm}} \mid \overline{\mathsf{burglary}}, \overline{\mathsf{fire}}).\mathbb{P}(\overline{\mathsf{burglary}}).\mathbb{P}(\overline{\mathsf{fire}}). \\ &\mathbb{P}(\mathsf{johnCalls} \mid \mathsf{alarm}).\mathbb{P}(\mathsf{alarm} \mid \overline{\mathsf{burglary}}, \mathsf{fire}).\mathbb{P}(\overline{\mathsf{burglary}}).\mathbb{P}(\mathsf{fire}). \end{aligned}$$

Note that in the Bayesian network of Figure 1 the local probability distributions are expressed via flat tables and, therefore, we can easily determine the parameters that maximize the Likelihood by simply counting. Suppose we want to learn the parameter corresponding to $\mathbb{P}(\mathsf{alarm} = \mathsf{true} \mid \mathsf{burglary} = \mathsf{false}, \mathsf{fire} = \mathsf{true})$. *All we need is to count the occurrences of* $\mathsf{alarm} = \mathsf{true}$ *among all examples where* $\mathsf{burglary} = \mathsf{false}$ *and* $\mathsf{fire} = \mathsf{true}$. *We than obtain*

Table 1: Complete dataset with fully observed examples.

Example	burglary	fire	alarm	johnCalls
1	true	false	true	true
2	false	false	false	false
3	false	true	true	true

estimate:

$$\frac{N_D(\mathsf{alarm} = \mathsf{true}, \mathsf{burglary} = \mathsf{false}, \mathsf{fire} = \mathsf{true})}{N_D(\mathsf{burglary} = \mathsf{false}, \mathsf{fire} = \mathsf{true})}.$$

Expectation Maximization (EM) When examples are not fully observed, the likelihood of the model becomes a complex expression. Suppose we want to learn a parameter corresponding to $\mathbb{P}(X_i = x_{ij} \mid \text{PA}[X_i] = w_{ik})$: if the patens of X_i are not fully observed we cannot simply count the occurrences of $X_i = x_{ij}$ among all examples where $\text{PA}[X_i] = w_{ik}$. Estimates can then be produced by applying the *Expectation Maximization* (EM) learning algorithm.

Each EM iteration has 2 steps: expectation (E-step) and maximization (M-step). The E-step consists of finding the expected Log-Likelihood expression we want to maximize. This implies computing the probability of observing each of the examples one could obtain by completing the values of hidden variables in the dataset. This gives us the following expected Log-Likelihood expression:

$$ELL(\boldsymbol{\theta} \mid D, \boldsymbol{\theta}^{(k-1)}) = \sum_{i}^{N} \sum_{\mathbf{z}_i} \mathbb{P}(\mathbf{z}_i \mid \mathbf{d}_i, \boldsymbol{\theta}) \cdot LL(\boldsymbol{\theta} \mid \mathbf{d}_i, \mathbf{z}_i),$$

where $\mathbf{d_i}$ denotes all the values observed in example i and $\mathbf{z_i}$ a set of possible values that could have been observed for the hidden variables in i. $LL(\boldsymbol{\theta} \mid \mathbf{d}_i, \mathbf{z}_i)$ is the Log-Likelihood of the model for an hypothetical example where $\mathbf{d_i}$ and $\mathbf{z_i}$ would have been observed. $\boldsymbol{\theta}$ are the parameters we want to optimize and $\boldsymbol{\theta}^{(k-1)}$ refers to the values calculated for the model parameters in the previous iteration of EM. The M-step is the maximization of this expected Log-Likelihood with respect to the parameters. This cycle is repeated until the expected Log-Likelihood $ELL(\boldsymbol{\theta} \mid D, \boldsymbol{\theta}^{(k-1)})$ reaches a local maximum [8].

Example 3. *Let us consider the case where variable* burglary *in Table 1 was never observed, as suggested in Table 2. Suppose, though, we want our model to include this hidden variable, because we believe it plays an important role in predicting the whole network dynamics. Let us see step by step how the EM algorithm would be executed. Firstly, let us*

Table 2: Complete dataset with hidden variables.

Example	burglary	fire	alarm	johnCalls
1		false	true	true
2		false	false	false
3		true	true	true

begin with the E-step. The likelihood expression would be written as:

$$L(B, D) = \mathbb{P}(\overline{\mathsf{fire}}, \mathsf{alarm}, \mathsf{johnCalls}).\mathbb{P}(\overline{\mathsf{fire}}, \overline{\mathsf{alarm}}, \overline{\mathsf{johnCalls}}).\mathbb{P}(\mathsf{fire}, \mathsf{alarm}, \mathsf{johnCalls}),$$

And, due to Markov independence assumption, factorized as:

$$\begin{aligned} L(B, D) =& \mathbb{P}(\mathsf{johnCalls} \mid \mathsf{alarm}).\mathbb{P}(\mathsf{alarm} \mid \overline{\mathsf{fire}}).\mathbb{P}(\overline{\mathsf{fire}}). \\ & \mathbb{P}(\overline{\mathsf{johnCalls}} \mid \overline{\mathsf{alarm}}).\mathbb{P}(\overline{\mathsf{alarm}} \mid \overline{\mathsf{fire}}).\mathbb{P}(\overline{\mathsf{fire}}). \\ & \mathbb{P}(\mathsf{johnCalls} \mid \mathsf{alarm}).\mathbb{P}(\mathsf{alarm} \mid \mathsf{fire}).\mathbb{P}(\mathsf{fire}). \end{aligned}$$

Notice that some of the terms in this expression are not known. We can, though, calculate their expected values:

$$\begin{aligned} E[\mathbb{P}(\mathsf{alarm} \mid \overline{\mathsf{fire}})] =& \hat{\mathbb{P}}(\mathsf{alarm} \mid \overline{\mathsf{fire}}, \mathsf{burglary}).\hat{\mathbb{P}}(\mathsf{burglary} \mid \overline{\mathsf{fire}}, \mathsf{alarm}, \mathsf{johnCalls})^{(k-1)} + \\ & \hat{\mathbb{P}}(\mathsf{alarm} \mid \overline{\mathsf{fire}}, \overline{\mathsf{burglary}}).\hat{\mathbb{P}}(\overline{\mathsf{burglary}} \mid \overline{\mathsf{fire}}, \mathsf{alarm}, \mathsf{johnCalls})^{(k-1)}; \end{aligned}$$

$$\begin{aligned} E[\mathbb{P}(\overline{\mathsf{alarm}} \mid \overline{\mathsf{fire}})] =& \hat{\mathbb{P}}(\overline{\mathsf{alarm}} \mid \overline{\mathsf{fire}}, \mathsf{burglary}).\hat{\mathbb{P}}(\mathsf{burglary} \mid \overline{\mathsf{fire}}, \overline{\mathsf{alarm}}, \overline{\mathsf{johnCalls}})^{(k-1)} + \\ & \hat{\mathbb{P}}(\overline{\mathsf{alarm}} \mid \overline{\mathsf{fire}}, \overline{\mathsf{burglary}}).\hat{\mathbb{P}}(\overline{\mathsf{burglary}} \mid \overline{\mathsf{fire}}, \overline{\mathsf{alarm}}, \overline{\mathsf{johnCalls}})^{(k-1)}; \end{aligned}$$

$$\begin{aligned} E[\mathbb{P}(\mathsf{alarm} \mid \mathsf{fire})] =& \hat{\mathbb{P}}(\mathsf{alarm} \mid \mathsf{fire}, \mathsf{burglary}).\hat{\mathbb{P}}(\mathsf{burglary} \mid \mathsf{fire}, \mathsf{alarm}, \mathsf{johnCalls})^{(k-1)} + \\ & \hat{\mathbb{P}}(\mathsf{alarm} \mid \mathsf{fire}, \overline{\mathsf{burglary}}).\hat{\mathbb{P}}(\overline{\mathsf{burglary}} \mid \mathsf{fire}, \mathsf{alarm}, \mathsf{johnCalls})^{(k-1)}. \end{aligned}$$

The terms requiring an inference on the Bayesian network are calculated using the parameter values found in the previous EM iteration $(k-1)$. Actually, each example can be seen as a set of hypothetical examples, whose occurrence is weighted by the expected likelihood of the inputted values, as shown in Table 3. This will give us an expression for the Expected Likelihood. We take the logarithm of this expression and then proceed to the M-step to find the parameters that maximize the Expected Log-Likelihood value.

Table 3: Expanded examples for complete dataset with hidden variables.

Example	Weight	burglary	fire	alarm	johnCalls
1.1	$\mathbb{P}(\mathsf{burglary} = \mathsf{true} \mid \mathbf{x_1}, \boldsymbol{\theta}^{(k)})$	true	false	true	true
1.2	$\mathbb{P}(\mathsf{burglary} = \mathsf{false} \mid \mathbf{x_1}, \boldsymbol{\theta}^{(k)})$	false	false	true	true
2.1	$\mathbb{P}(\mathsf{burglary} = \mathsf{true} \mid \mathbf{x_2}, \boldsymbol{\theta}^{(k)})$	true	false	false	false
2.2	$\mathbb{P}(\mathsf{burglary} = \mathsf{false} \mid \mathbf{x_2}, \boldsymbol{\theta}^{(k)})$	false	false	false	false
3.1	$\mathbb{P}(\mathsf{burglary} = \mathsf{true} \mid \mathbf{x_3}, \boldsymbol{\theta}^{(k)})$	true	true	true	true
3.2	$\mathbb{P}(\mathsf{burglary} = \mathsf{false} \mid \mathbf{x_3}, \boldsymbol{\theta}^{(k)})$	false	true	true	true

Table 4: Complete dataset with fully observed examples.

Example	X_1	X_2	X_3
1	true	false	true
2	false	false	false
3	false	true	true
4	false	true	true

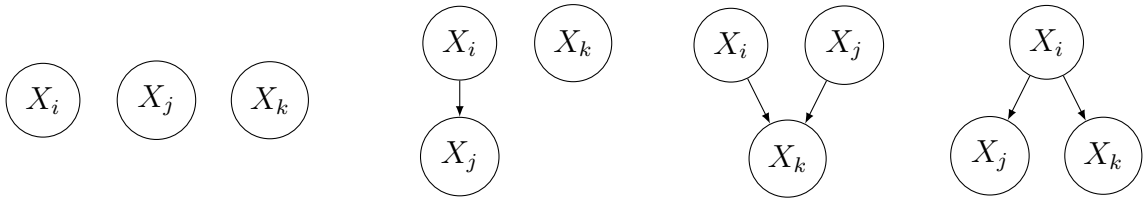
2.1.3 Structure Learning

When learning the structure of a Bayesian network we are looking for a directed acyclic graph, encoding a set of conditional independence assumptions, that will best fit the data when combined with the good parameters. There are two main approaches used for Bayesian networks structure learning: search-and-score and constraint based algorithms.

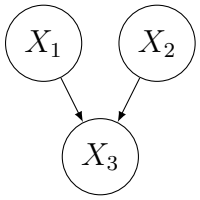
Search-and-score Algorithms In search-and-score algorithms a scoring function is used to evaluate the goodness of fit of some structure candidates. The parameters of each of these structure candidates are learned to approximate the data distribution. Once parameters for each candidate are set, all structures are scored and the one leading to highest score is selected as the final model [28]. Several scoring functions have been proposed in the literature. Most of them penalize candidates on their number of parameters, as an attempt to avoid too complex structures and data overfitting.

Example 4. *Suppose we want to learn a Bayesian network structure for the dataset in Table 4.*

Initially, we would have the following candidates to consider:



where i, j, k can be any of $1, 2, 3$. Let us calculate the score of, for instance, the third structure with $i = 1, j = 2, k = 3$. By counting on the occurrences of each configuration we obtain following parameters that maximize this candidate's local Log-Likelihood:



X_1	
T	F
0.25	*0.75*

X_2	
T	F
0.5	*0.5*

		X_3	
X_1	X_2	T	F
F	F	*0.0*	*1.0*
F	T	*1.0*	*0.0*
T	F	*0.0*	*1.0*
T	T	-	-

Notice that there is no example in Table 4 for configurations where $\{X_1 = T, X_2 = T\}$, therefore, the corresponding parameters do not appear in the Log-Likelihood. Suppose we choose the BIC score to guide this learning task. Let us calculate the local score for this specific candidate b:

$$BIC(b, D) = [\log(0.25) + 3.\log(0.75)] + [2.\log(0.5) + 2.\log(0.5)] + [\log(1.0) + 2.\log(1.0) + \log(1.0)] - \frac{1}{2} ln(4).6.$$

Notice that the minimal number of parameters $\mid \Theta \mid$ needed to represent the local conditional probabilities distribution is 6, not 12. The fact that the variables considered are all binary, implies parameters to be complementary for a given configuration and, therefore, we just need to store one parameter for each configuration.

Constraint-based Algorithms Constraint-based algorithms look for a structure satisfying as many independence constraints as possible. These constraints are obtained from independence tests on data. These try to identify pairs of variables which are independent from one another, given a set of other variables [43]. There are usually three steps to take [42]:

1. Learn the Markov blanket of each variable (optional);
2. Define the arcs composing the network *skeleton*; and
3. Define the direction of arcs.

The Markov blanket $\mathcal{B}(X)$ of a node X from a Bayesian network comprises all X's direct

parents, all X's direct successors and all direct parents of X's direct successors [33]. The Markov blankets can be found, for instance, by *feature selection* [3].

Step 2 in our list aims at finding the neighbors $\mathcal{N}(X)$ of each variable X: its direct parents and direct successors. Notice that, by firstly learning each variable's Markov blanket, we reduce the dimension of our search space, as $\mathcal{N}(X) \subset \mathcal{B}(X)$. Learning the network skeleton should then be much easier. For each pair of variables $\{X_i, X_j\}, X_j \in \mathcal{B}(X_i)$, we need to verify whether there is a set of variables $S_{X_i,X_j} \in \mathcal{B}(X_i) \setminus X_j$ such that X_i and X_j are independent given S_{X_i,X_j}. If we cannot find such a set of variables, than $X_j \in \mathcal{N}(X_i)$ and there must be an arc between X_i and X_j. Neighbors should be symetric, $X_i \in \mathcal{N}(X_j) \Leftrightarrow X_j \in \mathcal{N}(X_i)$.

To set a direction for the arcs found in step 2, one can:

- Firstly identify sets of variables $\{X_i, X_j, X_k\}$, such that $X_i \notin \mathcal{N}(X_j)$, $X_j \notin \mathcal{N}(X_i)$, $X_k \in \mathcal{N}(X_i)$ and $X_k \in \mathcal{N}(X_j)$. If $X_k \notin S_{X_i,X_j}$ than set both X_i and X_j as parents of X_k.

- If there is an undirected arc connecting a pair of variables $\{X_i, X_j\}$, which are already connected by a set of different directed arcs, set the direction of the undirected arc so as to avoid cycles.

- Finally, check remaining arcs, and if there is a set of variables $\{X_i, X_j, X_k\}$, such that X_i is a parent of X_j, $X_j \in \mathcal{N}(X_k)$, but $X_i \notin \mathcal{N}(X_k)$, then set X_j as a parent of X_k.

There are also hybrid models that try to combine both approaches, using both score-based and constraint-based methods [16, 44].

2.2 Logic Programming and Probabilistic Logic Programming

In this section, we review some basic concepts on *Logic Programming*, where concepts from logic are used to represent knowledge and reasoning processes [12, 29]. A logic program consists of a set of deterministic facts and rules. A (ground) fact is, intuitively, a statement such as: "Bill is a man", or "Bill is Ray's father". Rules consist of sentences that impose constraints amongst predicates.

Prolog is a logic programming language. In Prolog, a rule can be written as:

$$\mathsf{fatherOf}(\mathsf{A},\mathsf{B}) :- \mathsf{childOf}(\mathsf{B},\mathsf{A}),\mathsf{man}(\mathsf{A}).$$

and a fact is a rule with empty body, such as:

$$\mathsf{fatherOf}(\mathsf{bill},\mathsf{ray}).$$

The querying process consists of searching for true ground instances of an atomic clause such as:

$$\mathsf{fatherOf}(\mathsf{X},\mathsf{ray}).$$

which reads: "Is there a X such that X is father of Ray?"; or

$$\mathsf{fatherOf}(\mathsf{bill},\mathsf{ray}).$$

which reads: "Is Bill father of Ray?". In other words, a querying mechanism verifies whether, based on a knowledge base, an instance is true or not.

In Section 2.2.1 we define the syntax and semantics of *Normal Logic Programs*. In Section 2.2.2 we introduce *Probabilistic Logic Programs*, Sato's *Distribution Semantics* (Section 2.2.2.1) and ProbLog syntax [15, 25] (Section 2.2.2.2).

2.2.1 Normal Logic Programs

Consider a vocabulary with predicate symbols $r, s, \ldots$, logical variables $X, Y, \ldots$, and constants $a, b, \ldots$. A (lowercase) constant is a specific object, like bill or ray, and an (uppercase) logical variable, like A or B, stands for a set of constants of the same type. Each predicate symbol has an associated arity, which corresponds to the number of its arguments.

An *atom* is an expression of the form $r(t_1, t_2, \ldots, t_m)$ where r is a predicate symbol with arity m, and each t_i is a *term*, which is either a constant or a logical variable. A *literal* is an atom, such as $\mathsf{enjoys}(a, b)$, called a *positive* literal, or its negation, a *negative* literal. A *clause* consists of a single literal (atomic clause) or a disjunction of literals. A *Horn clause* contains at most one positive literal; a *definite* clause contains exactly one positive literal. A definite clause can be written as a *(deterministic) rule*:

$$h \leftarrow b_1 \wedge \cdots \wedge b_n,$$

where h denotes the rule *head* and b_is are the literals composing the rule *body*. A *fact* is

a rule with empty body. A *definite program* consists of a set of definite clauses together with a set of facts [12, 23, 30, 46].

A *normal logic program* allows for rules such as:

$$h \leftarrow b_1 \wedge \cdots \wedge b_n \cdots \wedge \overline{b}_{n+1} \wedge \ldots \overline{b}_m,$$

where $\overline{b}_i$ are negative literals. A negative literal is true when the program fails to derive the corresponding positive literal: *negation as failure*. Note this contributes to an improved representation power [2].

A substitution $\sigma = \{X_1 \to t_1, \ldots, X_m \to t_m\}$ is an injective partial function that maps a set of variables $\{X_1, \ldots, X_m\}$, called the domain of σ, to a set of terms $\{t_1, \ldots, t_m\}$, called the range of σ. The application of a substitution $\sigma = \{X_i \to t_i\}$ to an atom a, denoted $a\sigma$, maps all variables X_i in a, belonging to the domain of σ, to terms t_i. Other variables in a, not belonging to the domain of σ, are mapped to themselves. If no logical variable remains after the substitution, σ is called a grounding substitution of a and $a\sigma$ is a *ground* instance of a [12, 35]. The grounding of a rule is a *ground rule* obtained by applying to each atom the same grounding. The grounding of a program is the propositional program obtained by applying every possible grounding to each rule and fact, using only the constants in the program.

The *Herbrand base* of a logic program P is the set of all ground atoms one can obtain from the predicate symbols and constants in P. A *total interpretation* of P sets each of these atoms to true or false. A total interpretation $I = \{p, q\}$ denotes that only p and q are assigned true, other atoms in P are assigned false. If any atom remains undefined, we have a *partial interpretation* of P. A model of P is a partial interpretation where every single grounded rule in P is satisfied, meaning either its head is true or its body is false. If the number of non-negated literals in P is minimal, we have a *minimal model* of P [7].

Example 5. *Consider the following definite logic program:*

```
sick(X) :− contact(X, Y), sick(Y).
sick(X) :− fever(X).
sick(john).
sick(mary).
```

From this program, we would obtain following the Herbrand base:

{sick(mary), sick(john), contact(mary, mary), contact(mary, john)
contact(john, mary), contact(john, john), fever(mary), fever(john)}

and following grounded rules:

$$\mathsf{sick(mary)} :- \mathsf{contact(mary, mary)}, \mathsf{sick(mary)}.;$$

$$\mathsf{sick(mary)} :- \mathsf{contact(mary, john)}, \mathsf{sick(john)}.;$$

$$\mathsf{sick(john)} :- \mathsf{contact(john, mary)}, \mathsf{sick(mary)}.$$

$$\mathsf{sick(john)} :- \mathsf{contact(john, john)}, \mathsf{sick(john)}.$$

$$\mathsf{sick(john)} :- \mathsf{fever(john)}.$$

$$\mathsf{sick(mary)} :- \mathsf{fever(mary)}.$$

Consider, for instance, the following partial interpretation:

$$\Big\{\mathsf{sick(mary)} = \mathsf{false}, \mathsf{contact(john, mary)} = \mathsf{true}, \mathsf{fever(mary)} = \mathsf{true}\Big\}.$$

Note the fourth grounded rule is not satisfied in this partial interpretation of the logical program, as $\mathsf{sick(mary)} = \mathsf{false}$ *and* $\mathsf{fever(mary)} = \mathsf{true}$*. Therefore, this partial interpretation is not considered a model.*

The *dependency graph* of a grounded logic program is a directed graph, where each node represents a ground atom. If there is a ground rule with an atom A in the head and another atom B in the body, the corresponding nodes are connected throguh an arc $\langle A, B\rangle$. If B is a positive literal, arc $\langle A, B\rangle$ is a *positive* edge, else if B is a negative literal, arc $\langle A, B\rangle$ is as a *negative* edge. A logic program is *acyclic* if its grounded dependency graphs are acyclic and *stratified* if cycles in its grounded dependency graphs do not contain any negative edges.

Defining the semantics of a normal logic program is not an easy task. *Predicate completion*, introduced by Clark [5], was the first mathematically precise formalization of negation as failure. The following example shows how a first-order theory is obtained from predicate completion.

Example 6. *Consider the following normal logic program.*

$$\boxed{\begin{array}{l}\mathsf{A_0(a)}.\\ \mathsf{A_1(a,b)}.\\ \mathsf{A_2(a)} :- \mathsf{A_0(a)}.\\ \mathsf{A_2(b)} :- \overline{\mathsf{A_0}}\mathsf{(a)}, \mathsf{A_1(b,a)}.\end{array}}$$

Then replace all statements by a first-order logic representation, and combine rules with the same head:

$$\mathsf{A_0(X)} \Leftrightarrow (\mathsf{X} = \mathsf{a})$$
$$\mathsf{A_1(X,Y)} \Leftrightarrow (\mathsf{X} = \mathsf{a}) \wedge (\mathsf{Y} = \mathsf{b})$$
$$\mathsf{A_2(X)} \Leftrightarrow \Big((\mathsf{X} = \mathsf{a}) \wedge \mathsf{A_0(X)}\Big) \vee \Big((\mathsf{X} = \mathsf{b}) \wedge (\mathsf{Y} = \mathsf{a}) \wedge \overline{\mathsf{A_0}}(\mathsf{Y}), \mathsf{A_1(X,YOk)}\Big)$$

Alternatively, the semantics of a logic program can also be defined as a set of *stable models*. Formally, a total interpretation I is a stable model of P iff I coincides with the minimal model of the *reduct* P^I. The *reduct* P^I is a definite program obtained from P, so that rules that have in their body negated literals that are false according to the interpretation I are discarded and remaining rules have negated literals (if any) removed from their bodies. The next example illustrates how the stable models of a propositional logic program are obtained. The stable models for a non-propositional logic program are obtained from its grounding.

Example 7. *Consider the following normal logic program P:*

$$\mathsf{A_0}.$$
$$\mathsf{A_1} :\!- \mathsf{A_0}, \overline{\mathsf{A_2}}.$$
$$\mathsf{A_3} :\!- \mathsf{A_0}, \mathsf{A_2}.$$

and a total interpretation $I = \{\mathsf{A_0}, \mathsf{A_2}, \mathsf{A_3}\}$. As $\mathsf{A_2}$ is true *in I, the second rule is not in the reduct P^I. We obtain the following reduct P^I:*

$$\mathsf{A_0}.$$
$$\mathsf{A_3} :\!- \mathsf{A_0}, \mathsf{A_2}.$$

The minimal model of the reduct P^I is $\{\mathsf{A_0}\}$. Note the interpretation I is different from the minimal model of P^I and, therefore, I is not a stable model.

Consider now another interpretation $I' = \{\mathsf{A_0}, \mathsf{A_1}\}$. As $P^{I'}$ should be a definite program, the literal $\overline{A_1}$ must be removed from the body of the second rule and we obtain following reduct $P^{I'}$:

$$\mathsf{A_0}.$$
$$\mathsf{A_1} :\!- \mathsf{A_0}.$$
$$\mathsf{A_3} :\!- \mathsf{A_0}, \mathsf{A_2}.$$

This time I' coincides with the minimal model of $P^{I'}$. In this example, if we continue looking for stable models of P, we would not find any other stable model. Therefore, $\{\mathsf{A_0}, \mathsf{A_1}\}$ alone is the semantics of P.

2.2.2 Probabilistic Logic Programs

An old challenge, in the path to the development of intelligent agents, is the effective combination of logical formulas and probabilistic assessments [21, 32]. A promising strategy is to combine the formal language of logic programming with assessments of probability and of stochastic independence. Indeed, several proposals have tried to follow various kinds of *probabilistic logic programming* [17, 26, 48]. A particularly successful combination, often referred to as the *distribution semantics*, was originally developed by Poole [34, 36] and by Sato [40, 41]. The main point of the distribution semantics is to allow one to specify rules such as

$$\mathsf{fatherOf}(X, Y) :- \mathsf{parentOf}(X, Y), \mathsf{male}(X).$$

and additionally to specify probabilities, for example writing $\mathbb{P}(\mathsf{male}(\mathsf{a})) = 0.2$ to indicate that the probability that a is a male is 0.2. We now define the distribution semantics (Section 2.2.2.1) and ProbLog semantics (Section 2.2.2.2), which is central in this work.

2.2.2.1 Distribution Semantics

Let us define a *probabilistic logic program* (PLP) as a pair $\langle P, PF \rangle$, where P is a normal logic program and PF is a set of probabilistic facts. A probabilistic fact corresponds to an independent atomic clause (Sato's *independence assumption*), together with a probability label, which indicates the probability that this atomic clause is assigned true in a *total choice* of $\langle P, PF \rangle$. Sato's *disjointness condition* states that if a probabilistic fact in PF refers to a certain atomic clause, this atomic clause cannot be in the head of a rule in P [7].

For each total choice θ of $\langle P, PF \rangle$ we obtain a different normal logic program, denoted $\langle P, \theta \rangle$. If P is stratified, for each θ there is a unique normal logic program $\langle P, \theta \rangle$, which is also stratified. Therefore, for each total choice θ there is also a unique set of stable models. According to Satos' *distribution semantics*, a PLP can be described by the distribution over the stable models resulting from its total choices.

Example 8. *Consider the following stratified program:*

```
guilty(X) :- defraud(X).
guilty(X) :- hiding(X, Y), guilty(Y).
0.8 :: defraud(a).
0.6 :: hiding(b, a).
```

The probability that guilty(a) *is* true *is* 0.8 *and the probability that* guilty(b) *is* true *is* 0.48.

2.2.2.2 ProbLog

ProbLog follows the distribution semantics and extends the syntax of Prolog to represent probabilistic knowledge. A ProbLog program T consists of a set of independent probabilistic facts

$$F = \{\mathbb{P}(f_1) :: f_1, \ldots, \mathbb{P}(f_n) :: f_n\}$$

and a set of probabilistic rules.

$$R = \{\mathbb{P}(r_1) :: r_1, \ldots, \mathbb{P}(r_n) :: r_n\}$$

A probabilistic fact $\mathbb{P}(f_i) :: f_i$ corresponds to an atomic clause f_i, which may be grounded or not, together with a probability label $\mathbb{P}(f_i)$. A probabilistic rule $\mathbb{P}(r_j) :: r_j$ is a non-grounded definite clause, with more than a single predicate, also annotated with a probability label $\mathbb{P}(r_j)$ [27]. In fact, a probabilistic rule

$$\mathbb{P}(r_j) :: h_{r_j} \leftarrow b_{r_j}$$

is syntactically equivalent to

$$\mathbb{P}(r_j) :: f$$

$$h_{r_j} \leftarrow b_{r_j} \wedge f$$

where $\mathbb{P}(r_j) :: f$ is a probabilistic fact used to encode the probability of the body of rule r_j entailing its head.

For each probabilistic fact $\mathbb{P}(f_i) :: f_i$ in F, there is a set of ground instances $G_i = \{f_i\sigma_{i1}, \ldots, f_i\sigma_{im_i}\}$. We define G as the set of all ground instances of the probabilistic facts $\mathbb{P}(f_i) :: f_i \in F$ [12]:

$$G = \left\{ \left\{ f_1\sigma_{11} \ldots f_1\sigma_{1m_1} \right\}_{G_1}, \ldots, \left\{ f_n\sigma_{n1} \ldots f_n\sigma_{nm_n} \right\}_{G_n} \right\}$$

Example 9. *Consider, for instance, the following ProbLog program:*

```
0.5 :: man(X).
man(bill).
man(ray).
0.9 :: childOf(ray, alex).
0.9 :: childOf(ray, bill).
0.8 :: childOf(bill, jesse)).
0.9 :: childOf(alex, taylor).
1.0 :: fatherOf(A, B) :- childOf(B, A), man(A).
```

In this example, we do not know whether Alex, Jesse and Taylor are men, and this uncertainty is expressed via the probabilistic fact $0.5 :: \mathsf{man}(\mathsf{X})$. *This is a non-grounded probabilistic fact, whose ground instances can be* true*, with probability* $p = 0,5$*, or* false*, with probability* $(1-p) = 0,5$. *As the only non-grounded probabilistic fact in this program is* $0.5 :: \mathsf{man}(\mathsf{X})$*, we have:*

$$G = G_{\mathsf{man(X)}} = \{\mathsf{man(bill)}, \mathsf{man(ray)}, \mathsf{man(alex)}, \mathsf{man(jesse)}, \mathsf{man(taylor)}\}.$$

As mentioned, a ground instance $f_i\sigma_{ij}$ can be true with probability $\mathbb{P}(f_i)$, or false, with probability $1 - \mathbb{P}(f_i)$. Let us define random subsets $G' \subseteq G$:

$$G' = \Big\{ f_1\sigma_{1'}, \ldots, f_n\sigma_{n'} \Big\},$$

where $f_i\sigma_{i'} \in G_i$ are the ground instances sampled to be true, according to their probability distribution $\mathbb{P}(f_i)$, during grounding of a ProbLog program T [27].

Example 10. *Back to our example, we could have:*

$$G'^{(1)} = \Big\{ \mathsf{man(bill)}, \mathsf{man(ray)}, \mathsf{man(alex)} \Big\},$$

$$G'^{(2)} = \Big\{ \mathsf{man(bill)}, \mathsf{man(ray)}, \mathsf{man(jesse)} \Big\},$$

$$G'^{(3)} = \Big\{ \mathsf{man(ray)}, \mathsf{man(bill)}, \mathsf{man(alex)}, \mathsf{man(taylor)} \Big\},$$

etc.

The ProbLog querying process consists of calculating the probability of all ground instances $q\sigma_k$ of a query q in T. The success probability $P_S(q\sigma_k \leftarrow T)$ of a ground instance $q\sigma_k$ corresponds to the probability that this ground instance is true, given random sets of true ground facts $G' \subseteq G$ and the set of rules R [12, 27]. The expression to be computed

is:

$$P_S(q\sigma_k \leftarrow T) = \sum_{\substack{G' \subseteq G \\ q\sigma_k \leftarrow G' \cup R}} \prod_{c_i\sigma_{i'} \in G'} p_i \prod_{c_i\sigma_{i'} \in G \setminus G'} (1 - p_i).$$

Even though this expression may look daunting, its meaning is rather simple. Basically, all (grounded) probabilistic facts are independent, so we know the distribution over all of them (it is a product measure). Now for each fixed configuration of probabilistic facts, all other facts are fixed by the logical program, so the distribution over all (grounded) atoms is easily induced by the distribution over (grounded) probabilistic facts.

Example 11. *Let us calculate the probability of a query* $\mathsf{father}(\mathsf{ray}, \mathsf{alex})$*. According to the last rule of our program, if, for some* A *and some* B*,* $\mathsf{childOf}(\mathsf{A}, \mathsf{B})$ *and* $\mathsf{man}(\mathsf{A})$ *are* true*, then* $\mathsf{father}(\mathsf{A}, \mathsf{B})$ *must be* true*. If we take random sets of* true *ground facts* $G' \subseteq G$*, we should expect* $\mathsf{childOf}(\mathsf{ray}, \mathsf{alex})$ *to be* true *in 90% of cases and* $\mathsf{man}(\mathsf{alex})$ *in 50% of cases. Therefore, there is a* $0,45$ *probability that query* $\mathsf{father}(\mathsf{ray}, \mathsf{alex})$ *is* true*.*

2.2.3 Parameter Learning in ProbLog

Parameter learning consists of, given a fixed model structure, determining the probability parameters that best fit a dataset of grounded facts. In order to handle the parameter learning task, ProbLog introduces for each probabilistic rule an auxiliary fact. As mentioned in Section 2.2.2.2 a probabilistic rule $\mathbb{P}(r_j) :: h_{r_j} \leftarrow b_{r_j}$ is syntactically equivalent to a deterministic rule $h_{r_j} \leftarrow b_{r_j} \wedge f$ together with a probabilistic fact $\mathbb{P}(r_j) :: f$.

As the auxiliary variables introduced by ProbLog are not observed (latent), some sort of expectation maximization is required to implement a score-based learning algorithm. Notice these latent variables correspond to root nodes and, therefore, inference and maximization steps should be quite elementary. However, the number of iterations tends to grow with the number of latent variables considered. This is the main source of inefficiency in ProbLog's parameter learning.

Example 12. *Consider the following* A-PLP*:*

$\theta_1 :: \mathsf{bestRooms}.$
$\theta_2 :: \mathsf{bestFacilities}.$
$\theta_3 :: \mathsf{bestService}.$
$\theta_4 :: \mathsf{bestAccommodations} :- \mathsf{bestRooms}, \mathsf{bestFacilities}.$
$\theta_5 :: \mathsf{bestAccommodations} :- \overline{\mathsf{bestRooms}}, \mathsf{bestFacilities}.$
$\theta_6 :: \mathsf{bestAccommodations} :- \mathsf{bestRooms}, \overline{\mathsf{bestFacilities}}.$
$\theta_7 :: \mathsf{bestHotel} :- \mathsf{bestAccommodations}, \mathsf{bestService}.$

ProbLog, would interpret this program as follows:

θ_1 :: bestRooms.
θ_2 :: bestFacilities.
θ_3 :: bestService.
θ_4 :: aux$_4$.
bestAccommodations :− bestRooms, bestFacilities, aux$_4$.
θ_5 :: aux$_5$.
bestAccommodations :− $\overline{\text{bestRooms}}$, bestFacilities, aux$_5$.
θ_6 :: aux$_6$.
bestAccommodations :− bestRooms, $\overline{\text{bestFacilities}}$, aux$_6$.
θ_7 :: aux$_7$.
bestHotel :− bestAccommodations, bestService, aux$_7$.

and maximize the corresponding expected Log-Likelihood. To compute the probability of observing an example

$$\{\text{bestRooms}, \overline{\text{bestFacilitites}}, \text{bestAccommodations}, \text{bestService}, \overline{\text{bestHotel}}\},$$

for instance, we will need to infer the values of the auxiliary predicates aux$_4$, aux$_5$, aux$_6$ *and* aux$_7$*, which are not observed. Notice inference is very simple for these auxiliary predicates, they are* true *with probability* θ_4, θ_5, θ_6 *and* θ_7*, respectively, or* false *with probabilities* $1-\theta_4$, $1-\theta_5$, $1-\theta_6$ *and* $1-\theta_7$*. However, the more we need to infer, the more iterations may be required for the parameters to converge.*

2.2.4 Probabilistic Rule Learning

Probabilistic rule learning consists of building a predictive model for a target predicate t. The starting point is a background theory B, in the form of a ProbLog program, containing a set of grounded probabilistic facts. To build this predictive model, a set of true and false examples of the target predicate t is needed:

$$E = \Big\{(x_1, p_1)_{e_1}, \ldots, (x_n, p_n)_{e_n}\Big\},$$

where each example e_i consists of a possible ground instance $x_i = t\sigma_i$ and its probability $p_i = \mathbb{P}(t\sigma_i)$ [12].

Typically, a rule learner searches for a hypothesis H, a set of probabilistic rules entailing t, capable of, together with the background knowledge B, predicting the probability

$p_i = \mathbb{P}(t\sigma_i)$ that a ground instance $x_i = t\sigma_i$ is true [37,38]. This corresponds to minimizing a loss function [27]:

$$loss(H, B, E) = \sum_{(x_i,p_i)\in E} \mid P_S(x_i \leftarrow B \cup H) - p_i \mid .$$

ProbFOIL+ Learning Algorithm We now quickly review ProbFOIL+ algorithm, the state of the art in probabilistic rule learning. Firstly, by applying to the target predicate t all possible substitutions, mapping the terms of t to all constants of the same type in B, a set of examples is obtained:

$$E = \Big\{(x_1, p_1)_{e_1}, \ldots, (x_n, p_n)_{e_n}\Big\}.$$

An example e_i endorsed by the background theory is called positive, while examples not endorsed are called negative. The number of positive examples in E is denoted P and the number of negative examples is denoted N. Due to its probabilistic nature, each example that is a probabilistic fact is both positive and negative, contributing p_i to P and $(1 - p_i)$ to N.

The rule learning problem is treated in two loops. It starts with an empty hypothesis H, that keeps accepting new probabilistic rules learned $c(x) = (x :: c)$ until the predictive model stops improving with respect to a local scoring function:

$$accuracy_H = \frac{TP_H + TN_H}{M}$$

where TP_H corresponds to the number of positive examples correctly classified by H as positive, TN_H is the number of false examples correctly classified by H as negative, and M is the size of E. The inner loop searches for new candidates $c(x) = (x :: c)$ that maximize a local-scoring function:

$$M(x) = \frac{TP_{H\cup c(x)} + \frac{P}{N+P}}{TP_{H\cup c(x)} + FP_{H\cup c(x)+m}},$$

where $TP_{H\cup c(x)}$ corresponds to the number of true examples correctly classified by $H \cup c(x)$ as positive, $FP_{H\cup c(x)}$ is the number of negative examples incorrectly classified by $H \cup c(x)$ as positive, and m is a parameter of the algorithm.

The inner loop starts with a general rule

```
x :: target :- true.
```

which is capable of predicting all positive examples, but no negative example. All possible

refinements are then applied to this initial rule, thus, obtaining

$$\mathsf{x} :: \mathsf{target} :- \mathsf{true} \wedge \mathsf{refinements}.$$

The rules obtained are evaluated and, if not rejected by a set of heuristics, integrate a set of candidates to be extended in the next loop. For instance, there is no reason we should keep on refining a rule if the maximal score we could obtain with it is lower than the so far best rule score, or if $FP_{H \cup c(x)}$) is close to zero, which means the rule cannot be improved with the addiction of any refinement. Another heuristic states that refinements leading to rejection cannot contribute to improve the score of any candidate and, therefore, should not be reused.

Each of the remaining candidates might be extended and evaluated in the next loop, suggesting new candidates. The loop is repeated until no more candidates are suggested. The global score is then calculated considering the current hypothesis H together with the rule with best local score.

Example 13. *Consider the following deterministic dataset:*

father(bart, ben).	maleAncestor(bart, ben).
father(ben, pieter).	maleAncestor(ben, pieter).
father(jerry, jane).	maleAncestor(jerry, jane).
mother(jane, ben).	maleAncestor(bart, pieter).
mother(lucy, pieter).	maleAncestor(jerry, ben).

and the target predicate maleAncestor(A, B). *We consider a deterministic dataset to make it simpler to understand how the ProbFOIL+ algorithm works. We run the inner loop of the algorithm to search for a first rule to integrate the still empty hypothesis H. The inner loop starts with following rule candidate:*

$$\mathsf{maleAncestor}(\mathsf{A}, \mathsf{B}) : -\mathsf{true}.,$$

which we will try to refine. We try to extend the body of this candidate with all possible atoms. Allowed predicate symbols are the ones observed in the dataset: father*,* mother *and* maleAncestor*. Allowed logical variables are the ones our candidate already includes:*A *ans* B*, plus another unknown logical variable,* C *for instance. After a first refinement following rules achieve the highest local scores:*

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A}, \mathsf{B}) : -\mathsf{father}(\mathsf{A}, \mathsf{B}).$$

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A}, \mathsf{B}) : -\mathsf{mother}(\mathsf{C}, \mathsf{B}).$$

$$0.66 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{maleAncestor}(\mathsf{C},\mathsf{B}).$$

$$0.66 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{father}(\mathsf{C},\mathsf{B}).$$

We keep refining the best candidates obtained until no more improvement is observed with respect to the local score. Among all rules encountered, the one that gives the highest local score is:

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{father}(\mathsf{A},\mathsf{B}).$$

We verify it contributes to a global score improvement and add it to our hypothesis H. The loop is repeated until no more improvement is observed with respect to the global score. In the end, the following rules integrate the hypothesis H:

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{father}(\mathsf{A},\mathsf{B}).$$

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{father}(\mathsf{A},\mathsf{C}),\mathsf{father}(\mathsf{C},\mathsf{B}).$$

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{maleAncestor}(\mathsf{A},\mathsf{C}),\mathsf{maleAncestor}(\mathsf{C},\mathsf{B}).$$

$$1.0 :: \mathsf{maleAncestor}(\mathsf{A},\mathsf{B}) : -\mathsf{father}(\mathsf{A},\mathsf{C}),\mathsf{mother}(\mathsf{C},\mathsf{B}).$$

3 NOVEL ALGORITHMS FOR PARAMETER AND STRUCTURE LEARNING

In this chapter we propose novel techniques both for parameter and structure learning of A-PLPs [9, 11], respectively in Sections 3.1 and 3.2. In Section 3.1.1 we discuss how scores imported from the literature on Bayesian network learning can be computed for A-PLPs. In Section 3.1.2 we show that the score optimization problem can often be solved in closed-form. In Sections 3.1.3 and 3.1.4 we delve into our parameter learning algorithm, that is, how the parameters of A-PLPs can be learned, both from complete and incomplete data. Finally, in Section 3.2, we present an exact score-based algorithm for learning the structure of AP-PLPs.

3.1 Parameter Learning

In this section, we present an algorithm that learns the parameters of acyclic PLPs by adapting techniques from Bayesian network learning. We must, therefore, translate the language of acyclic PLPs into the language of Bayesian networks. Each ground predicate in our vocabulary is viewed as a binary random variable (0 is false and 1 is true), and the predicates that appear in the body of a rule are parents of the predicate in the head of the rule.

3.1.1 Score Computation

We must now understand how the Log-Likelihood introduced in Section 2 can be computed for a PLP (the same reasoning applies to a wide variety of decomposable scoring functions, like the BIC score). The major difference between computing the Log-Likelihood for an acyclic PLP and a Bayesian network (with conditional probability distributions explicitly encoded by flat tables) is that there may be some overlap between the bodies of rules sharing a head. This means, for example, that if an atom h is entailed by two

rules $\theta_1 :: h \leftarrow b_1$ and $\theta_2 :: h \leftarrow b_2$, and only by these two rules, we have:

$$\mathbb{P}(h) = 1 - (1 - \theta_1).(1 - \theta_2) = \theta_1 + \theta_2 - \theta_1.\theta_2.$$

Note that, if the rules are deterministic, then rules are automatically combined using disjunction, by the semantics of acyclic logic programs. When probabilities are present, the behavior of rules that share a head is akin to the noisy-or function often used in Bayesian networks. Consider two examples to understand how the noisy-or construct is used to compute the Log-Likelihood of a PLP, both in propositional and relational cases.

Example 14. *Consider the following propositional* PLP*:*

$\theta_1 :: \mathsf{burglary}.$
$\theta_2 :: \mathsf{fire}.$
$\theta_3 :: \mathsf{alarm} :- \mathsf{burglary}, \mathsf{fire}.$
$\theta_4 :: \mathsf{alarm} :- \mathsf{burglary}, \overline{\mathsf{fire}}.$
$\theta_5 :: \mathsf{alarm} :- \overline{\mathsf{burglary}}, \mathsf{fire}.$
$\theta_6 :: \mathsf{johnCalls} :- \mathsf{alarm}.$
$\theta_7 :: \mathsf{johnCalls} :- \overline{\mathsf{alarm}}.$

This PLP *is acyclic and the dependency graph encoded is similar to the graph of the Bayesian network depicted in Figure 1. For a given dataset like*

$$\left\{\{\mathsf{burglary}, \overline{\mathsf{fire}}, \mathsf{alarm}, \mathsf{johnCalls}\}_1\right\},$$

in this case with a single observation ($N = 1$), one can easily calculate the Log-Likelihood of the model. We have:

$$\begin{aligned} LL(P,T) = &\log[\mathbb{P}(\mathsf{burglary})] + \log[\mathbb{P}(\overline{\mathsf{fire}})] \\ &+ \log[\mathbb{P}(\mathsf{alarm} \mid \mathsf{burglary}, \overline{\mathsf{fire}})] + \log[\mathbb{P}(\mathsf{callsJohn} \mid \mathsf{alarm})] \end{aligned}$$

hence

$$LL(P,T) = \log\theta_1 + \log\theta_2 + \log\theta_4 + \log\theta_6.$$

Consider now a slightly different propositional PLP*:*

θ_1 :: burglary.
θ_2 :: fire.
θ_3 :: alarm :− burglary.
θ_4 :: alarm :− fire.
θ_5 :: johnCalls :− alarm.
θ_6 :: johnCalls :− $\overline{\text{alarm}}$.

The dependency graph for this new PLP *remains the same. Notice, however, that in this case the rules defining the probability distribution over variable* alarm *are not mutually exclusive anymore. From our definition of the Log-Likelihood,* $LL(B, D) = \log[\mathbb{P}(D \mid B)]$*, we would now have:*

$$LL(P,T) = \log[\mathbb{P}(\text{burglary})] + \log[\mathbb{P}(\overline{\text{fire}})] + \log[\mathbb{P}(\text{callsJohn} \mid \text{alarm})] + \log[\mathbb{P}(\text{alarm} \mid \text{burglary}) + \mathbb{P}(\text{alarm} \mid \overline{\text{fire}}) - \mathbb{P}(\text{alarm} \mid \text{burglary}).\mathbb{P}(\text{alarm} \mid \overline{\text{fire}})]$$

hence

$$LL(P,T) = \log[\theta_1] + \log[1 - \theta_2] + \log[\theta_5] + \log[\theta_3].$$

Example 15. *Consider the following relational program:*

person(john).
person(mary).
samePerson(john, john).
samePerson(mary, mary).
θ_0 :: fire(X) :− person(X).
θ_1 :: burglary(X) :− person(X).
θ_3 :: alarm(X) :− fire(X).
θ_4 :: alarm(X) :− burglary(X).
θ_5 :: cares(X, Y) :− person(X), person(Y).
θ_6 :: calls(X, Y) :− cares(X, Y), alarm(Y).

In order to calculate the Log-Likelihood of such a relational model, we need to instantiate it to all possible substitutions. Figure 3 shows the dependency graph resulting from instantiation of this relational model. For a complete dataset $D = \{\{\text{calls(john, mary)}, ...\}_1\}$*, our Log-Likelihood expression would be:*

$$LL(B, D) = \log[\mathbb{P}(\text{calls(john, mary)} \mid ...)] + ... = \log[\theta_6] + ...$$

Consider now a more complex relational program, where, for some instantiation of a

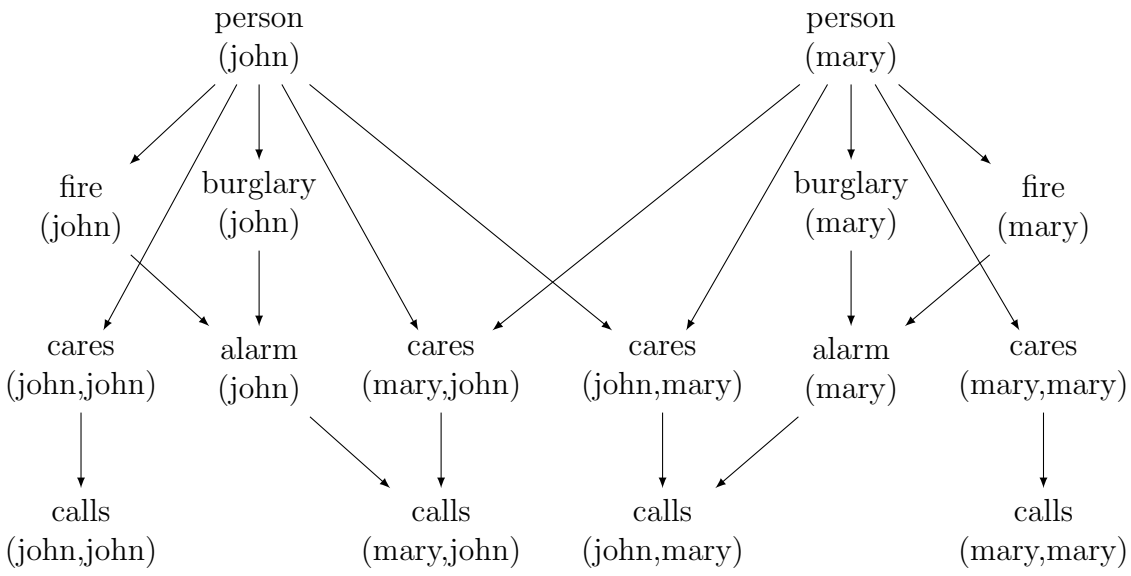


Figure 3: Dependency graph generated by instantiation of a relational logic program.

rule head, there is more than one possible instantiation of the corresponding body. For instance, let us replace the last rule of our example for:

$$\boxed{\theta_6 :: \mathsf{callsPolice(X)} :- \mathsf{cares(X, Y), alarm(Y).}}$$

In this last case, for a complete dataset

$$D = \left\{\{\mathsf{callsPolice(john), cares(john, mary), cares(john, peter), alarm(peter), alarm(mary), ...}\}_1\right\}$$

our Log-Likelihood expression would be:

$$\begin{aligned} LL(B, D) = log[&\mathbb{P}(\mathsf{callsPolice(john)} \mid \mathsf{cares(john, peter), alarm(peter)}) \\ &+ \mathbb{P}(\mathsf{callsPolice(john)} \mid \mathsf{cares(john, mary), alarm(mary)}) \\ &- \mathbb{P}(\mathsf{callsPolice(john)} \mid \mathsf{cares(john, peter), alarm(peter)}) \\ &.\mathbb{P}(\mathsf{callsPolice(john)} \mid \mathsf{cares(john, mary), alarm(mary)})] + \ldots \end{aligned}$$

Notice that the probability of observing a given head instantiation can only increase as other body instantiations are verified true.

Notice that, if acyclicity is not ensured, writing the likelihood expression of a logic program is certainly not obvious. In addition, it is not clear how cycles should be interpreted. That is the reason why we decided to focus on acyclic PLPs.

3.1.2 Exact Solutions

As mentioned in Section 3.1.1, differently from the case presented in Section 2.1.2, we usually cannot maximize the Log-Likelihood expression of an A-PLP by simply counting. Instead, we must write down the Log-Likelihood expression and maximize it. Remember the Log-Likelihood is decomposable and, therefore, it can be computed locally and then summed for all families of the program. Thus, we can rewrite the Log-likelihood expression of an A-PLP:

$$LL(P, D) = \sum_{d \in D} \sum_{i=1}^{n} log(\theta_{ijk}),$$

as follows:

$$LL(P, D) = \sum_{i=1}^{n} \sum_{d \in D} log(\theta_{ijk}).$$

Due to Markov independence assumption, for different values of i, the parameter values θ_{ijk} are independent from each other. The parameter values that maximize $LL(P, D)$ are then

$$\boldsymbol{\theta} = \bigcup_{i=1}^{n} \operatorname*{argmax}_{\boldsymbol{\theta}_i} \sum_{d \in D} log(\theta_{ijk}).$$

This means that, by locally maximizing the Log-Likelihood for each family, global maximization is ensured.

Consider, for instance, a family where $\text{PA}[X_0] = \{X_1, X2\}$ and the probability distribution of X_0 is given by two rules $\theta_1 :: \mathsf{X_0} :- \mathsf{X_1}, \overline{\mathsf{X_2}}$ and $\theta_2 :: \mathsf{X_0} :- \overline{\mathsf{X_1}}, \mathsf{X_2}$. The likelihood expression for this family would be:

$$L(R, D) = \theta_1^{a_0}(1 - \theta_1)^{a_1}\theta_2^{b_0}(1 - \theta_2)^{b_1}$$

If there is no observation with zero likelihood, we have following exact solution of the optimization problem:

$$\theta_1 = \frac{a_0}{a_0 + a_1}, \theta_2 = \frac{b_0}{b_0 + b_1}$$

Other combinations of rules lead to similar likelihood expressions. For instance, if the probability distribution of X_0 was given by other two rules $\theta_1 :: \mathsf{X_0} :- \mathsf{X_1}$ and $\theta_2 :: \mathsf{X_0} :- \mathsf{X_2}$ we would obtain an identical likelihood expression. Appendix A lists all likelihood expressions one could obtain under following restrictions:

- There are at most three combinations of rules entailing a same predicate, and
- Rules have only one possible body instantiation for each head instantiation

In fact, there is an exact solution for several of these likelihood expressions. In our implementations, we have covered all these cases. Notice that the Log-Likelihood expression does not depend on the size of the rules' bodies. This contributes to a considerable reduction on the optimization complexity, once only a few cases may require a numerical approach to be solved. Whenever a closed-form solution does not exist, a fast gradient-based routine can be used for a numerical approximation.

3.1.3 Complete Data

If data are complete, all variables are observed in all examples. The Log-Likelihood can then be easily calculated and finding the parameters to maximize it should be very simple. However, even when the data are complete, the auxiliary facts introduced by ProbLog to handle probabilistic rules are missing. The fact that some variables are actually not observed, implies that some sort of expectation maximization is required, as mentioned in Section 2.2.3.

Our solution is not to insert an auxiliary (latent) atom for each probabilistic rule. Instead, we must write down the Log-Likelihood and maximize it directly. For many rule patterns, this maximization can be done in closed-form, as shown in Section 3.1.2. The main insight, however, is that inference is very time-consuming and should be avoided whenever possible during learning iterations.

Example 16. *This is a comparative example to illustrate how our approach and ProbLog differ in learning the parameters of following very simple* A-PLP*:*

θ_1 :: burglary.
θ_2 :: fire.
θ_3 :: alarm :− burglary, fire.

Our approach aims at maximizing following Log-Likelihood expression:

$$LL(R, D) = a_0 log\theta_1 + a_1 log(1 - \theta_1) + b_0 log\theta_2 + b_1 log(1 - \theta_2) + c_0 log\theta_3 + c_1 log(1 - \theta_3),$$

where a_0 *is the number of examples for which* burglary *is* true *and* a_1 *is the number of examples for which* burglary *is* false. b_0 *and* b_1 *follow the same reasoning.* c_0 *denotes the number of examples for which* alarm, burglary *and* fire *are true, while* c_1 *denotes the number of examples for which* alarm *and* burglary *are* true, *but* alarm *is* false. *We suppose there are no inconsistent evidences among the examples. In other words, there are no examples where either* burglary *or* fire *is* false, *but* alarm *is* true.

ProbLog, instead, would interpret this program as follows:

θ_1 :: burglary.
θ_2 :: fire.
θ_3 :: aux.
alarm :− burglary, fire, aux.

and maximize the corresponding expected Log-Likelihood. Notice that the predicate aux *introduced by ProbLog is never observed and, therefore, a number of inferences on the hidden variable* aux *is required to maximize the expected Log-Likelihood through an iterative process. In contrast, our approach does not require any inference on latent variables.*

3.1.4 Incomplete Data

In most real-world applications, the examples are not fully observed. A variable that is never observed for any data instance is called a hidden variable, as introduced in Section 2.1.2. If all variables are potentially observed, but for some data instances, the values for some variables are not known, we say data is incomplete. *Expectation Maximization* (EM) has traditionally been used to deal with the problem of hidden variables. However, we can derive a variant of EM to deal with the missing data points in an incomplete dataset.

EM algorithm aims at maximizing the expected likelihood. This means we are interested in finding the set of parameters $\boldsymbol{\theta} = \{\theta_1, \theta_2, \dots\}$ that gives us the *Maximum Likelihood* estimate for a given set or rules $\{\theta_1 :: c_1, \theta_2 :: c_2, \dots\}$. The general form for the EM algorithm, is

$$\boldsymbol{\theta}^{(k+1)} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{\mathbf{z}} \mathbb{P}(\mathbf{Z} = \mathbf{z} \mid \mathbf{x}, \boldsymbol{\theta}^{(k)}) \cdot LL(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{Z} = \mathbf{z}) \ ,$$

where $\boldsymbol{\theta}^{(k)}$ denotes the values calculated for the model parameters in step k of the learning algorithm, $\mathbf{x}$ denotes all the values observed for non-hidden variables and $\mathbf{z}$ denotes all the values expected for the hidden-variables $\mathbf{Z}$. $LL(\boldsymbol{\theta} \mid \mathbf{x}, \mathbf{Z} = \mathbf{z})$ is the Log-Likelihood of the model for an hypothetical dataset where $\mathbf{x}$ and $\mathbf{z}$ would be observed.

In the case of an incomplete dataset, the non-observed variables are not always the same for every example. Let $\mathbf{d}_i$ represent the observed values for an example i in dataset D, $\mathbf{C_i}$ the variables for which values are missing in example i and $\mathbf{c_i}$ a possible way to complete these missing values. E-step gives then the following expression for the model's

Table 5: Examples for incomplete dataset.

Example	burglary	fire	alarm	johnCalls
1		false	true	true
2	true	true		false
3	false	true	true	true

Table 6: Expanded examples for incomplete dataset.

Example	Weight	burglary	fire	alarm	johnCalls
1.1	$\mathbb{P}(C_1 = c_{1.1} \mid d_1, \boldsymbol{\theta}^{(k)})$	true	false	true	true
1.2	$\mathbb{P}(C_1 = c_{1.2} \mid d_1, \boldsymbol{\theta}^{(k)})$	false	false	true	true
2.1	$\mathbb{P}(C_2 = c_{2.1} \mid d_1, \boldsymbol{\theta}^{(k)})$	true	true	true	false
2.2	$\mathbb{P}(C_2 = c_{2.2} \mid d_1, \boldsymbol{\theta}^{(k)})$	true	true	false	false
3	1.0	false	true	true	true

expected Log-Likelihood:

$$ELL(\boldsymbol{\theta} \mid D, \boldsymbol{\theta}^{(k)}) = \sum_i \sum_{\mathbf{c}_i} \mathbb{P}(\mathbf{C}_i = \mathbf{c}_i \mid \mathbf{d}_i, \boldsymbol{\theta}^{(k)}) \cdot LL(\boldsymbol{\theta} \mid \mathbf{d}_i, \mathbf{C}_i = \mathbf{c}_i).$$

By application of Markov condition, we can rewrite this expression as follows:

$$ELL(\boldsymbol{\theta} \mid D, \boldsymbol{\theta}^{(k)}) = \sum_x \sum_i \sum_{\mathbf{c}_{x,i}} \mathbb{P}(\mathbf{c}_{x,i} \mid \mathbf{d}_i, \boldsymbol{\theta}^{(k)}) \cdot \log \mathbb{P}(x_{\mathbf{c}_{x,i}} \mid \text{PA}[X]_{\mathbf{c}_{x,i}}, \boldsymbol{\theta}),$$

where $\sum_{\mathbf{c}_{x,i}}$ iterates over all possible configurations $\{X = x_{\mathbf{c}_{x,i}}, \text{PA}[X] = \text{PA}[X]_{\mathbf{c}_{x,i}}\}$ the ith example of the dataset can assume. Note that this expression is decomposable and can be maximized on each variable X independently.

Example 17. *Consider the following* PLP *structure. Note its dependency graph is similar to the graph of the Bayesian network depicted in Figure 1.*

θ_1 :: burglary.
θ_2 :: fire.
θ_3 :: alarm :– burglary.
θ_4 :: alarm :– fire.
θ_5 :: johnCalls :– alarm.
θ_6 :: johnCalls :– $\overline{\text{alarm}}$.

Suppose also some of the data points in Table 1 were not observed, as suggested in Table 5. Each incomplete observation can be seen as a set of complete observations, whose occurrence is weighted by a probability term, as shown in Table 6.

Suppose we want to learn the parameter θ_{ijk} that approximates $\mathbb{P}(\text{alarm} = \text{true} \mid$

$\mathsf{burglary} = \mathsf{false}, \mathsf{fire} = \mathsf{true}$). *If the conditional probabilities distributions were encoded via flat tables we could simply count the weighted occurrences of* $\mathsf{alarm} = \mathsf{true}$ *among all examples where we could hypothetically observe* $\mathsf{burglary} = \mathsf{false}$ *and* $\mathsf{fire} = \mathsf{true}$. *Instead, conditional probabilities distributions are encoded by a set of probabilistic rules. Note there are two different rules that share the same head predicate* alarm*, which are both activated by the third line in Table 5. For illustration purposes, let us write down the local log-likelihood expression for the family composed of variable* alarm *and its parents* $\{\mathsf{burglary}, \mathsf{earthquake}\}$.

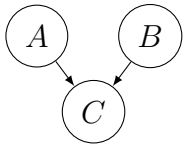
$$\begin{aligned} LL(B,D)^{(alarm|burglary,earthquake)} &= \mathbb{P}(C_1 = c_{1.1} \mid d_1, \boldsymbol{\theta}^{(k)}).log(\theta_3) \\ &+ \mathbb{P}(C_1 = c_{1.2} \mid d_1, \boldsymbol{\theta}^{(k)}).log(0) \\ &+ \mathbb{P}(C_2 = c_{2.1} \mid d_1, \boldsymbol{\theta}^{(k)}).log(\theta_3 + \theta_4 - \theta_3.\theta_4) \\ &+ \mathbb{P}(C_2 = c_{2.2} \mid d_1, \boldsymbol{\theta}^{(k)}).log(1 - \theta_3 - \theta_4 + \theta_3.\theta_4) \\ &+ log(\theta_4) \end{aligned}$$

Same procedure should be repeated to find the contribution of the other families to the expected Log-Likelihood expression. Note that these contributions are independent and, therefore, if we maximize them locally, global maximization is guaranteed. The values $\boldsymbol{\theta}^{(k)}$ *are the ones obtained in the previous iteration of our EM-like iterative algorithm.*

When data are complete, we can not avoid making some inferences. But by reducing the number of latent variables, we reduce the number of inferences required per iteration. Indeed, as we have to deal with probabilistic rules, our inferences may be much more complex than the ones implied in ProbLog's approach. However, we expect that the reduction on the number of iterations, as a consequence of the reduction on the number of inferences per iteration, may compensate for the complexity increase and improve learning performance.

Example 18. *Consider the following very simple* A-PLP*:*

$\theta_1 :: \mathsf{A}.$
$\theta_2 :: \mathsf{B}.$
$\theta_3 :: \mathsf{C} :- \mathsf{A}, \overline{\mathsf{B}}.$ $\theta_4 :: \mathsf{C} :- \mathsf{A}, \mathsf{B}.$



And a single observation:

$$\{\mathsf{A}, \overline{\mathsf{B}}\}.$$

Problog would interpret this program as:

$\theta_1 :: \mathsf{A}.$
$\theta_2 :: \mathsf{B}.$
$\theta_3 :: \mathsf{aux_3}.$
$\mathsf{C} :- \mathsf{A}, \mathsf{B}, \mathsf{aux_3}.$
$\theta_4 :: \mathsf{aux_4}.$
$\mathsf{C} :- \mathsf{A}, \mathsf{B}, \mathsf{aux_4}.$

and seak to maximize the expected log-likelihood through following expression:

$$\begin{aligned}
ELL(\boldsymbol{\theta} \mid D, \boldsymbol{\theta}^{(k)}) = {} & \mathbb{P}(\mathsf{C}, \mathsf{aux_3}, \mathsf{aux_4} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \mathsf{C}, \mathsf{aux_3}, \mathsf{aux_4}) \\
& + \mathbb{P}(\mathsf{C}, \overline{\mathsf{aux_3}}, \mathsf{aux_4} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \mathsf{C}, \overline{\mathsf{aux}}, \mathsf{aux_4}) \\
& + \mathbb{P}(\overline{\mathsf{C}}, \mathsf{aux_3}, \mathsf{aux_4} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \overline{\mathsf{C}}, \mathsf{aux_3}, \mathsf{aux_4}) \\
& + \mathbb{P}(\overline{\mathsf{C}}, \overline{\mathsf{aux_3}}, \mathsf{aux_4} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \overline{\mathsf{C}}, \overline{\mathsf{aux_3}}, \mathsf{aux_4}) \\
& + \mathbb{P}(\mathsf{C}, \mathsf{aux_3}, \overline{\mathsf{aux_4}} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \mathsf{C}, \mathsf{aux_3}, \overline{\mathsf{aux_4}}) \\
& + \mathbb{P}(\mathsf{C}, \overline{\mathsf{aux_3}}, \overline{\mathsf{aux_4}} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \mathsf{C}, \overline{\mathsf{aux}}, \overline{\mathsf{aux_4}}) \\
& + \mathbb{P}(\overline{\mathsf{C}}, \mathsf{aux_3}, \overline{\mathsf{aux_4}} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \overline{\mathsf{C}}, \mathsf{aux_3}, \overline{\mathsf{aux_4}}) \\
& + \mathbb{P}(\overline{\mathsf{C}}, \overline{\mathsf{aux_3}}, \overline{\mathsf{aux_4}} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \overline{\mathsf{C}}, \overline{\mathsf{aux_3}}, \overline{\mathsf{aux_4}}).
\end{aligned}$$

We instead try to maximize it through following expression:

$$\begin{aligned}
ELL(\boldsymbol{\theta} \mid D, \boldsymbol{\theta}^{(k)}) = {} & \mathbb{P}(\mathsf{C} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \mathsf{C}) \\
& + \mathbb{P}(\mathsf{C} \mid A, \overline{B}, \boldsymbol{\theta}^{(k)}).LL(\boldsymbol{\theta} \mid \mathsf{A}, \overline{\mathsf{B}}, \mathsf{C})
\end{aligned}$$

Our approach requires only 2 inferences per iteration, against ProbLog's 8 inferences. But notice that ProbLog's inferences are simpler, once rules are all deterministic.

3.2 Structure Learning

Learning a Bayesian network consists in finding a directed acyclic, composed of a set of families, and the parameters that best represent the probabilities distribution for each of these families. But there is a difference between usual Bayesian network learning and PLP learning: even within a family, we must choose a set of rules to encode the

conditional probabilities distribution. In this section, we want to learn the structure of PLPs, by maximizing a score imported from Bayesian networks literature. Note that this is a much more complex task than just learning the rules for a specific target.

We focus on the class of propositional A-PLPs with at most 2 parents per variable, AP-PLP(2). We propose a two steps learning algorithm:

1. Firstly learn the rules set that locally maximizes the likelihood of each family candidate,
2. Then choose the families that globally maximize the BIC score.

But before calculating the scores of all possible combinations of rules for each family candidate, some combinations can be pruned: (1) those whose likelihood is equal to zero, and (2) those that, compared to another existing combination with similar or higher penalization, are ensured to result in a similar maximal likelihood estimate.

Example 19. *Let us consider, for instance, a set of binary variables* $\{X_1, X_2, X_3\}$*, where* $\text{PA}[X_1] = \{X_2, X_3\}$*. Suppose we had to choose between two possible combinations of rules:*

Combination 1	*Combination 2*
$\theta_1 :: X_1 :- X_2, X_3.$	$\theta_1 :: X_1 :- X_2.$
$\theta_2 :: X_1 :- X_2, \overline{X_3}.$	$\theta_2 :: X_1 :- X_2, \overline{X_3}.$
$\theta_3 :: X_1 :- \overline{X_2}, X_3.$	$\theta_3 :: X_1 :- \overline{X_2}, X_3.$

These combinations look very similar, but let us take a look at the corresponding flat tables.

Combination 1

X_2	X_3	$\mathbb{P}(X_1 = \mathsf{true} \mid X_2, X_3)$
false	false	0
false	true	θ_3
true	false	θ_2
true	true	θ_1

Combination 2

X_2	X_3	$\mathbb{P}(X_1 = \mathsf{true} \mid X_2, X_3)$
false	false	0
false	true	θ_3
true	false	$\theta_{1;2} = \theta_1 + \theta_2 - \theta_1.\theta_2$
true	true	θ_1

Each line corresponds to the base of an exponential factor in the likelihood expression we want to maximize.

$$LL(\boldsymbol{\theta} \mid D)^{(1)} = (0)^{N_{100}}(1)^{N_{000}}{\theta_3}^{N_{101}}(1-\theta_3)^{N_{001}}{\theta_2}^{N_{110}}(1-\theta_2)^{N_{110}}{\theta_1}^{N_{111}}(1-\theta_1)^{N_{111}}$$

$$LL(\boldsymbol{\theta} \mid D)^{(2)} = (0)^{N_{100}}(1)^{N_{000}}\theta_3^{N_{101}}(1-\theta_3)^{N_{001}}{\theta_{1;2}}^{N_{110}}(1-\theta_{1;2})^{N_{110}}\theta_1^{N_{111}}(1-\theta_1)^{N_{111}}$$

The likelihood expression of Combination 2 contains a nonlinear function of the parameters, a fact that complicates the maximization step. Notice, however, if we optimize the likelihood expression for Combination 1, we can directly infer the optimal parameters for Combination 2 by solving following possible determinate system:

$$\theta_1^{(1)} = \theta_1^{(2)}$$

$$\theta_1^{(1)} = \theta_1^{(2)} + \theta_2^2 - \theta_1^1.\theta_2^2$$

$$\theta_3^{(1)} = \theta_3^{(2)}$$

Therefore, there is no reason to keep both combinations among the possible candidates. Thus, we can prune either Combination 1 or Combination 2, without any loss in score.

In addition, notice that for the class AP-PLP*(2) we do not need to consider combinations of more than four rules neither. There would not be a combination of four rules with better likelihood than the ones equivalent to a flat table.*

$\theta_1 :: X_1 :- \overline{X_2}, \overline{X_3}.$
$\theta_2 :: X_1 :- \overline{X_2}, X_3.$
$\theta_3 :: X_1 :- X_2, \overline{X_3}.$
$\theta_4 :: X_1 :- X_2, X_3.$

X_2	X_3	$\mathbb{P}(X_1 = \mathsf{true} \mid X_2, X_3)$
false	false	θ_1
false	true	θ_2
true	false	θ_3
true	true	θ_4

Any other combination of four rules or more would lead at best to the same maximal likelihood estimate, though with a higher penalization if there are more than four rules to consider.

Obviously, the number of possible rule sets grows fast with the number of atoms that may appear in bodies. This explains why, in order to limit the number of possible cases to consider, we impose in this work the restriction that resulting PLPs must belong to the class AP-PLP(2).

Appendix C lists all possible combinations of rules for a family such that PA$[X_1] = \{X_2, X_3\}$: there are 4 sets consisting each of one rule, 30 sets consisting of two rules each, and 82 sets consisting of three rules each, and one set of 4 rules. Each of the tables in Appendix D shows a combination of rules that remains after pruning and the ones pruned for satisfying condition (2): rules that are ensured to result in a lower likelihood value and similar or higher penalization.

After pruning, we need to find the optimal parameters for each of the remaining combinations of rules. Remember that if we have at most three rules to consider for a

same head predicate, several optimization steps can be solved exactly, as discussed in Section 3.1.2. Appendix B lists all the possible likelihood expressions one could obtain for class AP-PLP(2), which is a subset of the expressions listed in Appendix A together with a combination of four rules encoding the equivalent of a flat table. Actually, also for this combination of four rules there is a closed-form solution, the one we presented in Section 2.1.2. Once the optimal parameters are found for each combination of rules, each family candidate is associated with a highest score combination.

Finally, we need to find the families that globally maximize the BIC score. Van Beek and Hoffmann proposed a constraint-programming algorithm (CPBayes) [45] to find the families that maximize the global score of a Bayesian network. The CPBayes algorithm implements a depth-first BnB search restricted to a set of constraints: dominance constraints, symmetry-breaking constraints, cost-based pruning rules, and a global acyclicity constraint. When trying to expand a node in the search tree, two conditions are verified: (1) whether constraints are satisfied, and (2) whether a lower bound estimate of the cost (a negative score) does not exceed the current upper bound [9, 45]. We use CPBayes algorithm to run our global optimization step, but other approaches could have been used, and our contribution is certainly not due to our use of CPBayes.

The procedure we have developed is summarized by Algorithm 1 [9].

Algorithm 1 Learning algorithm for class AP-PLP(2).

```
collect variables from dataset
for each family of variables in dataset do
    build all possible rules
    build all possible combinations of rules
    gather rule sets into patterns
    for each pattern do
        prune combinations with ensured lower score
        prune combinations with zero likelihood
        for each combination left do
            if there is an exact solution to the likelihood maximization problem then
                calculate parameters
            else
                run numeric (exact or approximate) likelihood maximization
        calculate local scores
    for each family do
        associate best rule set with family
call CPBayes algorithm to maximize global score
```

4 EXPERIMENTS

We first investigate parameter learning in Section 3.1. We compare our results with the ProbLog implementation, in order to measure the improvements in computation speed. Next we dive into structure learning and verify whether our algorithm produces AP-PLP(2)s with less parameters and higher score than Bayesian Networks learned under similar constraints. This should be an indicative of a better generalization capacity.

The algorithms were implemented in Python and experiments were initially performed on a Unix Machine with Intel core i5 (2.7 GHz) processor and 8 GB 1867 MHz DDR3 SDRAM. Larger tests were run in identical processors at Amazon Web Services. Source code for parameter learning is freely available in Github at
https://github.com/arthurcgusmao/asteroidea/tree/master/asteroidea,
and source code for structure learning is freely available in Github at
https://github.com/franciscodefaria/AP-PLP—Structure-Learning.git.

4.1 Parameter Learning

Complete Data Figure 6 shows how the original ProbLog parameter learning algorithm and the algorithm we propose compare in terms of computation effort for complete datasets, both in propositional and relational cases, as described bellow [11].

Consider first an acyclic propositional program that encodes the energy plant of a ship. We have derived the model from a simplified energy plant diagram, which is obtained from the description available in the site http://www.machineryspaces.com/emergency-power-supply.html. Our simplified diagram is depicted in Figure 4, where a states for alternator, ll for lighting loads and pl for power loads, which are on if true or off, otherwise. The probabilistic representation of such an energy plant allows reasoning over its availability, a major concern when planning maintenance procedures and equipment replacement.

0.95 :: lowSupply :– a1.	0.95 :: lowLoad :– highLoad.	0.75 :: a2 :– a4.
1.0 :: lowSupply :– highSupply.	0.8 :: lowLoad :– ll1, pl1.	0.85 :: a1.
0.95 :: failure :– highLoad, $\overline{\text{highSupply}}$.	0.8 :: lowLoad :– ll2, pl2.	0.95 :: a3.
0.95 :: failure :– lowLoad, $\overline{\text{lowSupply}}$.	0.8 :: lowLoad :– ll3, pl3.	0.95 :: a4.
0.98 :: emergency :– $\overline{\text{a3}}$, $\overline{\text{a4}}$.	0.95 :: highSupply :– a2, a3.	0.8 :: ll2.
0.7 :: ll1 :– emergency.	0.95 :: highSupply :– a2, a4.	0.8 :: pl2.
0.7 :: pl1 :– emergency.	0.95 :: highSupply :– a3, a4.	0.8 :: ll3.
0.6 :: highLoad :– ll2, ll3, pl2, pl3.	0.75 :: a2 :– a3.	0.8 :: pl3.

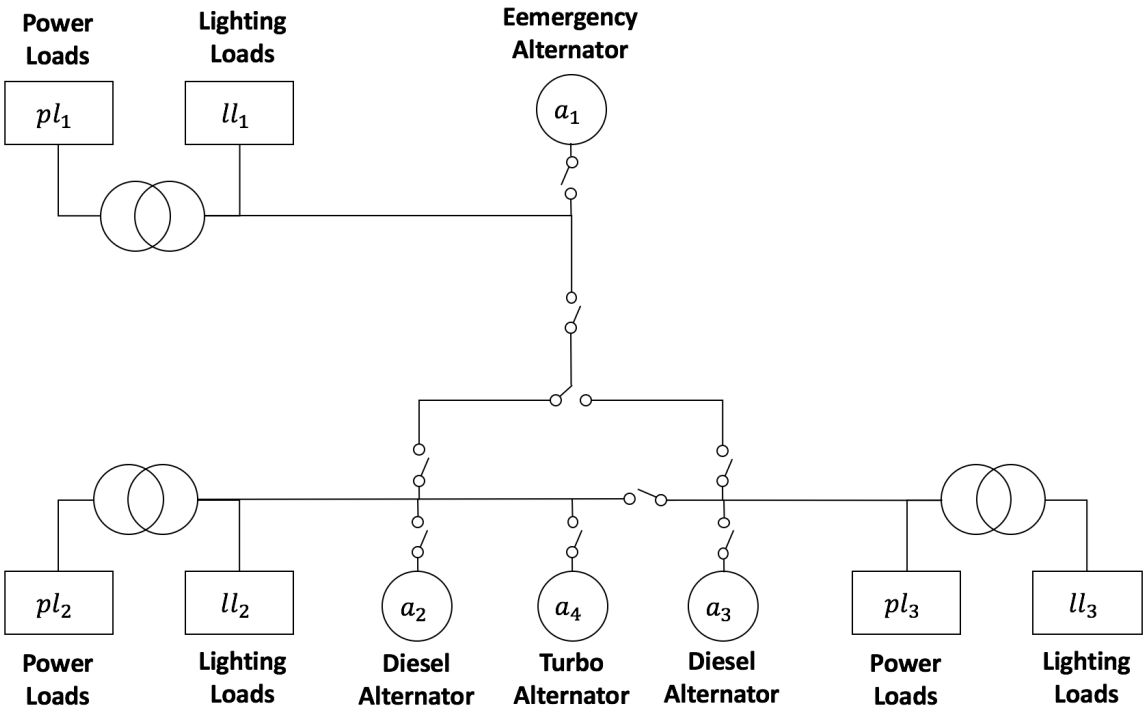


Figure 4: Simplified energy plant diagram used in the first complete data experiment.

The second experiment encodes following relational A-PLP.

0.3 :: fire(X).
0.4 :: burglary(X).
0.2 :: neighbor(X, Y).
0.7 :: alarm(X) :– fire(X).
0.9 :: alarm(X) :– burglary(X).
0.8 :: calls(X, Y) :– neighbor(X, Y), alarm(Y).

For a relational dataset with only two constants ($N = 2$), again mary and john for instance, the PLP instantiation would result in the dependency graph of Figure 5.

In the propositional case, size of dataset is the number of observations for all atoms; in the relational case, size of dataset is the number of constants in the program. The

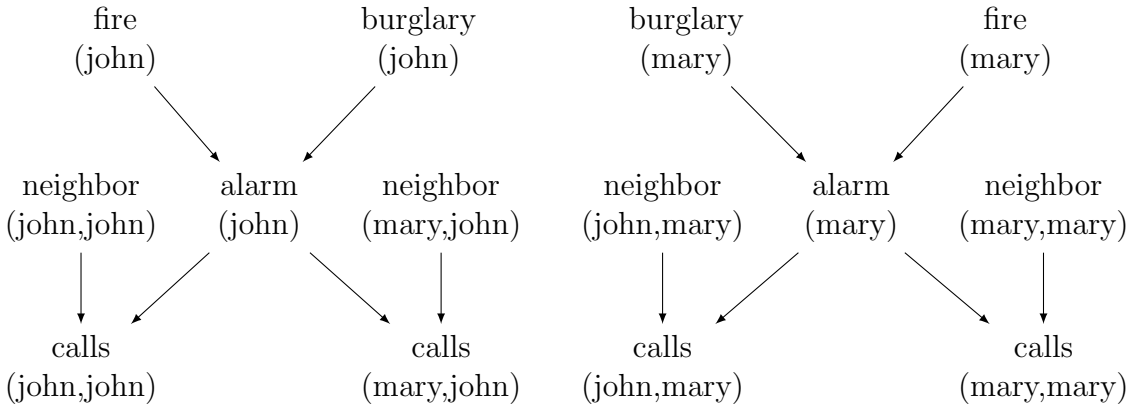


Figure 5: Dependency graph generated by instantiation of the relational program used in the second complete data experiment.

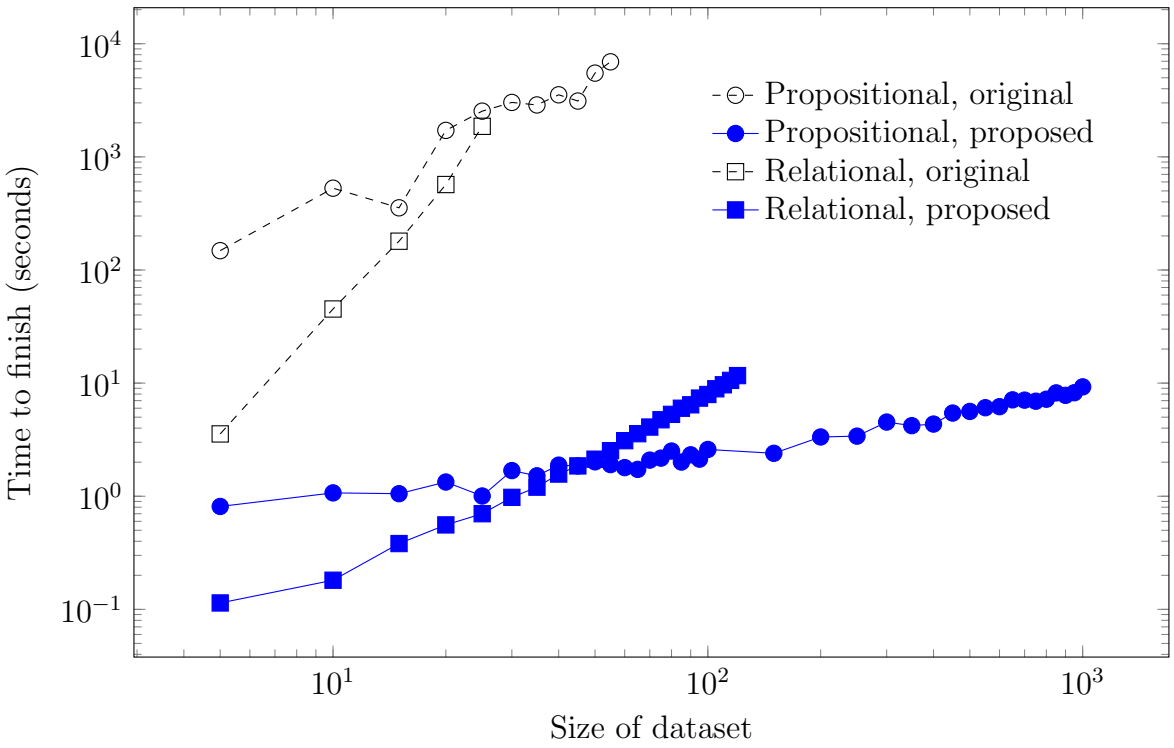


Figure 6: Time to learn parameters from complete data.

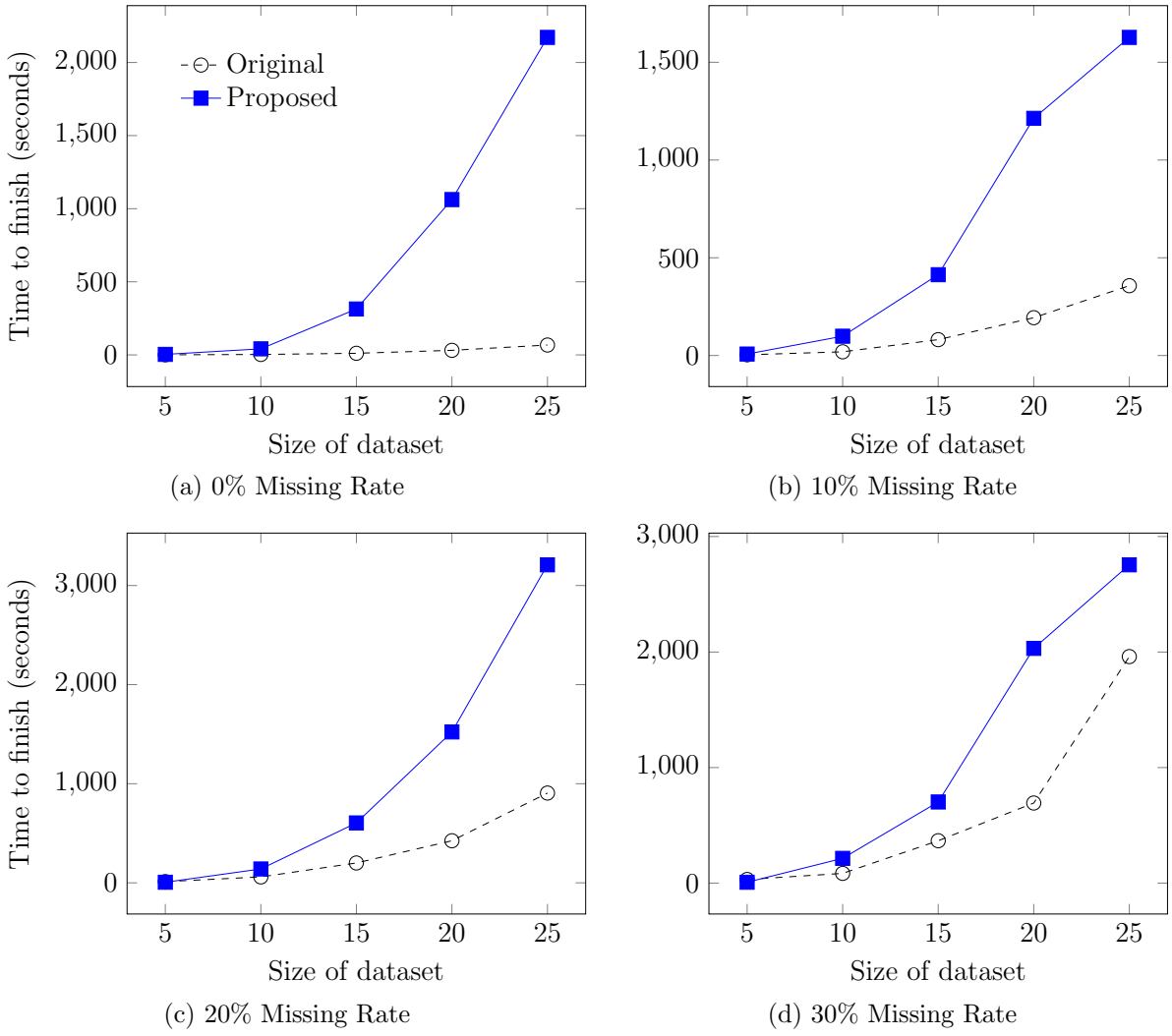
size of examples is limited by ProbLog's inability to handle larger models in reasonable time. In both experiments, the two learning algorithms reach similar Log-Likelihood values, but notice our implementations can contribute with a significant reduction on computation time (note log-scale!). In the first experiment, for a propositional dataset with 50 observations, ProbLog reaches a Log-Likelihood of -176.65 in 5498.06 seconds, while our algorithm reaches a Log-Likelihood of -176.60 in 2.02 seconds. In the second experiment, for a relational dataset with 25 constants, ProbLog reaches a Log-Likelihood of -523.11 in 1863.03 seconds, while our algorithm reaches a Log-Likelihood of -523.11 in 0.69 seconds. The gain in computation time is mainly due to the fact that no iterative process is required to learn the parameters from complete data.

Incomplete Data Table 7 shows how the original ProbLog parameter learning algorithm (blue) and the algorithm we propose (white) compare in terms of computation time for an incomplete relational dataset sampled from following structure:

$$
\begin{array}{|l|}
\hline
0.3 :: \mathsf{fire}(\mathsf{X}) :- \mathsf{person}(\mathsf{X}). \\
0.4 :: \mathsf{burglary}(\mathsf{X}) :- \mathsf{person}(\mathsf{X}). \\
0.7 :: \mathsf{alarm}(\mathsf{X}) :- \mathsf{fire}(\mathsf{X}). \\
0.9 :: \mathsf{alarm}(\mathsf{X}) :- \mathsf{burglary}(\mathsf{X}). \\
0.8 :: \mathsf{cares}(\mathsf{X},\mathsf{Y}) :- \mathsf{person}(\mathsf{X}), \mathsf{person}(\mathsf{Y}). \\
0.8 :: \mathsf{calls}(\mathsf{X},\mathsf{Y}) :- \mathsf{cares}(\mathsf{X},\mathsf{Y}), \mathsf{alarm}(\mathsf{Y}), \overline{\mathsf{samePerson}(\mathsf{X},\mathsf{Y})}. \\
\hline
\end{array}
$$

ProbLog's stopping criteria is defined over the convergence of the Log-Likelihood values observed. Notice, however, as ProbLog's iterative process differs substantially from our algorithm's, using the same stopping criteria does not guarantee similar Log-Likelihood values are reached [10]. In order to ensure a valid comparison basis, we set ProbLog's stopping criteria to 1×10^{-3}, which means it stops when the Log-likelihood variation between subsequent iterations is smaller than 1×10^{-3}, and set our algorithm to stop whenever is overcomes the Log-Likelihood values obtained with ProbLog. Each data point in Table 7 corresponds to the average computation time for learning from four different datasets we generated using the target parameters. Again, size of dataset is the number of constants in the program. Solid squares were generated by ProbLog; empty circles were generated by our algorithm.

Our algorithm outperformed ProbLog in the vast majority of cases, losing only for datasets with 5 constants and 20% or more of missing data. From the curve, it is easy to notice, that when the size of datasets increases, the ratio between ProbLog's and our



(a) 0% Missing Rate

(b) 10% Missing Rate

(c) 20% Missing Rate

(d) 30% Missing Rate

Figure 7: Time to learn parameters from incomplete relational data using ProbLog learning algorithm (blue) and our algorithm (white)

approach's computation time tends to decrease. This was already expected, as the number of iterations tends to grow with the number of latent variables.

4.2 Structure Learning

We have implemented the structure learning algorithm described in Section 3.2, and tested it with a number of complete propositional datasets. We compare two different approaches to learn the local structure of the family candidates: (1) allowing only for combinations of rules encoding complete probability tables and (2) allowing for any combination of rules.

There are a few combinations of rules for which the local likelihood maximization problem has no closed-form solution. For numerical maximization, two different algorithms were considered: (1) Limited-memory BFGS (L-BFGS) and (2) the Basin-hopping algorithm [47]. The L-BFGS algorithm approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [22], which is an iterative method for solving unconstrained nonlinear optimization problems. L-BFGS allows for a significant reduction on memory consumption, once it uses a few vectors to approximate the inverse Hessian matrix. Nevertheless, it has a quite strong dependence on the initial guess. Conversely, Basin-hopping repeats the optimization process for a number of different initial guesses, usually providing a better approximation of the global maximum. This algorithm is however much more time-consuming [9]. Both methods are implemented in the Python library scipy.optimize.

We tested our implementations for three different applications:

Test 1 - Binary Adder This first test aims at learning an AP-PLP(2) to represent a (simulated) faulty Boolean circuit, which is designed to add two 4-bit numbers in the binary numeral system. In order to generate binary datasets for training and testing, we took a number of pairs of randomly selected 4-bit numbers. Each pair was processed by a circuit of 18 logic gates (XOR, OR, AND) that simulates the binary adder. Each gate was associated with a 1% probability of failure. This probability of failure indicates how often gates should output random values, instead of the values expected. Training datasets contain 30, 60, 90, 120, 250, 500 and 1000 examples and the testing dataset contains 10000 examples. Examples contain 24 observations: 8 entry bits and 18 output values, one for each gate. For the optimization steps, we used L-BFGS algorithm. A few tests were repeated using Basin-hopping optimization algorithm, which is much more time-

Table 7: Binary adder experiment results.

L-BFGS					
# Instances		Complete CPT		Any combination of rules	
Training set	Testing set	Log-Likelihood	Parameters	Log-Likelihood	Parameters
30	10000	-317590.82	61	-190592.57	48
60	10000	-282860.54	63	-211535.40	51
90	10000	-231096.56	61	-200086.83	48
120	10000	-281571.16	61	-197029.87	51
250	10000	-244887.02	65	-251489.19	51
500	10000	-228617.04	66	-217608.84	52
1000	10000	-188236.10	80	-177049.65	62
Basin-hopping					
# Instances		Complete CPT		Any combination of rules	
Training set	Testing set	Log-Likelihood	Parameters	Log-Likelihood	Parameters
30	10000	-344580.91	61	-190652.02	48
1000	10000	-188236.10	80	-177049.52	62

consuming, but no improvements were observed in terms of Log-Likelihood and number of parameters. Results obtained are listed in Table 7 [9].

For smaller datasets, the algorithm we proposed scores better on testing and requires fewer parameters. For larger datasets, both approaches tend to converge in terms of Log-Likelihood, but there is still a significant reduction on the number of parameters. Notice this is a nearly-deterministic application and, therefore, it was already expected that logic rules would be capable of encoding the local probability distributions much more compactly than complete CPT's. In addition, this is the typical example for which the restriction of having at most two parents per variable does not jeopardize the prediction capability, once we know each logical gate has only two input terminals.

Test 2 - Heart Diagnosis This second test aims at learning an AP-PLP(2) to reason over diagnosis made from observation of cardiac Single Proton Emission Computed Tomography (SPECT) images [1]. The training and testing datasets contain 80 and 187 instances, respectively. 23 binary attributes are observed for each example, there is no missing data. The learning algorithm is tested with both L-BFGS and Basin-Hopping optimization methods. Results obtained are listed in Table 8. Notice that, again, both approaches, the one allowing only for Complete CPTs and the other allowing for any combination of rules, tend to converge in terms of Log-Likelihood, but there is still a significant reduction on the number of parameters [9]. In addition, results obtained with both optimization methods, L-BFGS and Basin-Hopping, are exactly the same.

Table 8: Heart diagnosis experiment results.

L-BFGS			
Complete CPT		Any combination of rules	
Log-Likelihood	Parameters	Log-Likelihood	Parameters
-1281.78	63	-1263.85	55
Basin-hopping			
Complete CPT		Any combination of rules	
Log-Likelihood	Parameters	Log-Likelihood	Parameters
-1281.78	63	-1263.85	55

Table 9: Movie genres experiments results.

Complete CPT		Any combination of rules	
Log-Likelihood	Parameters	Log-Likelihood	Parameters
-2032.38	62	-2031.49	42

Test 3 - Movie genres This third test aims at learning an AP-PLP(2) to reason over movie genres. Labels result from previous multi-classification of movies according to the genres they represent. Information is extracted from MovieLens datasets[1]. Training and testing datasets are obtained from a random 70/30 split of the original dataset, which contains 3449 movies. The learning algorithm is tested with L-BFGS optimization method. Results obtained are listed in Table 9. The resulting AP-PLP(2) and the corresponding dependency graph are shown below:

[1]Available at http://grouplens.org/datasets/movielens/

0.04 :: war.	0.10 :: adventure :– $\overline{\text{drama}}$.
0.00 :: children.	0.03 :: adventure :– $\overline{\text{action}}$, drama.
0.14 :: comedy.	0.02 :: romance.
0.31 :: comedy :– $\overline{\text{thriller}}$, $\overline{\text{war}}$.	0.16 :: romance :– $\overline{\text{documentary}}$, $\overline{\text{horror}}$.
0.00 :: thriller :– war.	0.09 :: animation :– adventure.
0.17 :: thriller :– $\overline{\text{war}}$.	0.00 :: animation :– drama.
0.13 :: action :– $\overline{\text{comedy}}$.	0.04 :: animation :– $\overline{\text{drama}}$.
0.24 :: action :– thriller.	0.02 :: crime :– horror.
0.06 :: action :– comedy, $\overline{\text{thriller}}$.	0.27 :: crime :– $\overline{\text{horror}}$, thriller.
0.27 :: drama.	0.06 :: crime :– $\overline{\text{horror}}$, $\overline{\text{thriller}}$.
0.53 :: drama :– $\overline{\text{action}}$, $\overline{\text{comedy}}$.	0.17 :: fantasy :– adventure.
0.01 :: western.	0.23 :: fantasy :– animation.
0.03 :: western :– $\overline{\text{comedy}}$, $\overline{\text{drama}}$.	0.03 :: fantasy :– $\overline{\text{adventure}}$, $\overline{\text{animation}}$.
0.03 :: horror.	0.00 :: musical :– thriller.
0.23 :: horror :– $\overline{\text{comedy}}$, $\overline{\text{drama}}$.	0.33 :: musical :– animation, $\overline{\text{thriller}}$.
0.00 :: documentary.	0.03 :: musical :– $\overline{\text{animation}}$, $\overline{\text{thriller}}$.
0.08 :: documentary :– $\overline{\text{comedy}}$, $\overline{\text{drama}}$.	0.01 :: filmNoir.
0.03 :: sciFi :– drama.	0.13 :: filmNoir :– $\overline{\text{action}}$, crime.
0.27 :: sciFi :– action, $\overline{\text{drama}}$.	0.22 :: mystery :– filmNoir.
0.07 :: sciFi :– $\overline{\text{action}}$, $\overline{\text{drama}}$.	0.13 :: mystery :– thriller.
0.27 :: adventure :– action.	0.02 :: mystery :– $\overline{\text{filmNoir}}$, $\overline{\text{thriller}}$.

In this experiment we do not observe a significant improvement in terms of Log-Likelihood, but the number of parameters was considerably reduced. Notice that the parameter values found are quite small. The only label that could be predicted true, woth probability higher than 0.5, is drama, iff action and comedy are both false with a probability of at least 0.95. Actually, in this context, the restriction of having at most two parents per variable may have been too strong. However, the dependency graph still gives us an idea of which labels are more correlated.

It is also important to highlight that if the number of attributes to consider becomes much larger, some sort of feature selection may be recommended to reduce the search space and consequently computation time. Also, if the number of instances observed increases considerably, sampling might be required. Special attention should be given to the quality of the sample used, which should ensure a good approximation of the original dataset parameters.

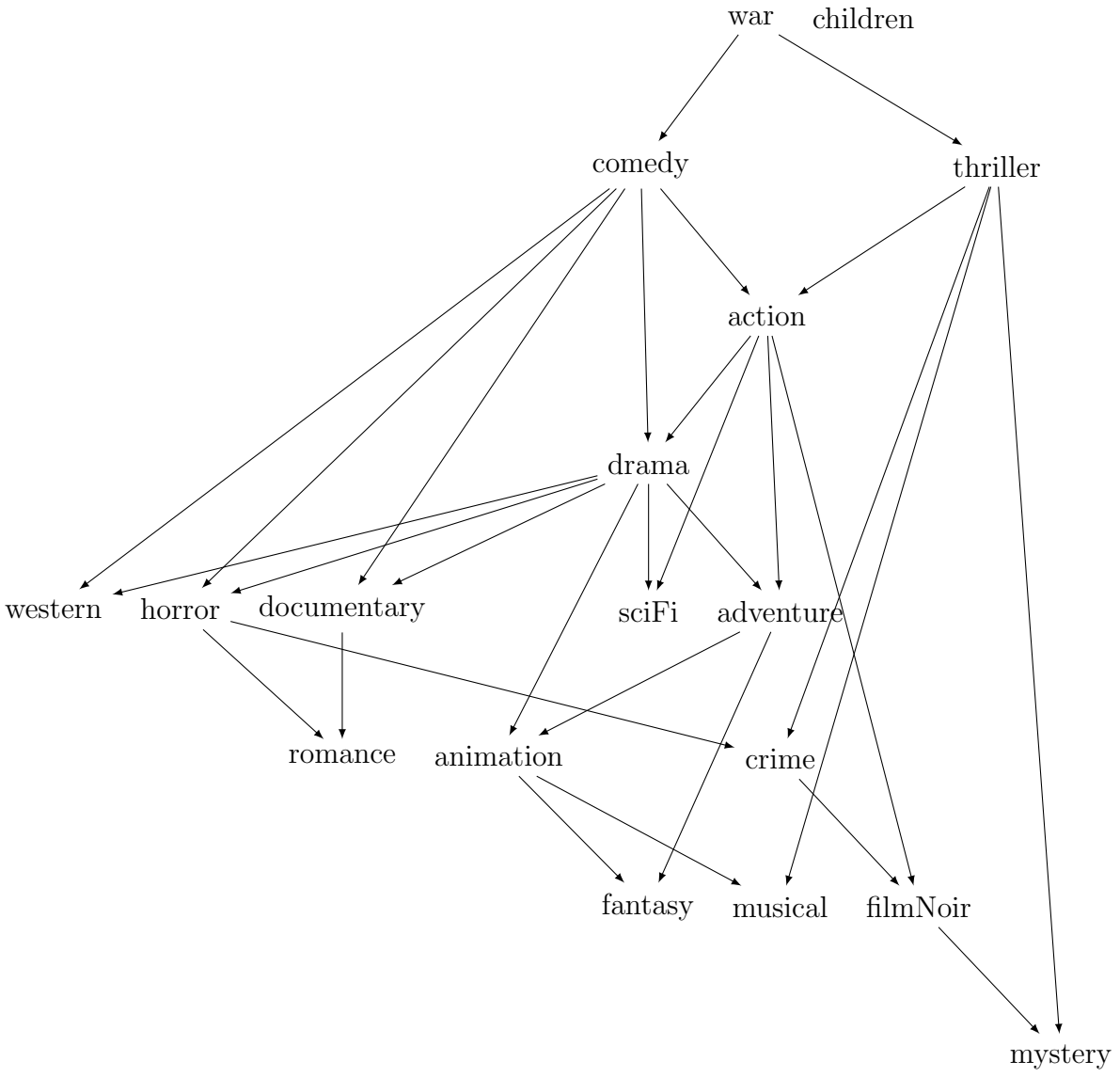


Figure 8: Dependency graph of the AP-PLP(2) learned in the movie genres experiment.

5 CONCLUSIONS

The goal of this work was to develop algorithms that learn parameters and structure of PLPs from data. We focused on acyclic PLPs, which are closely related to Bayesian networks. We proposed, therefore, to adapt techniques from Bayesian network learning to deal with the challenging task of learning probabilistic programs.

We first introduced novel techniques that lead to significant improvements in ProbLog's parameter learning algorithm. We exploited two main insights. First, we avoided introducing unnecessary latent atoms, as they require an iterative optimization process. Second, we locally maximized the Log-Likelihood for each family in the most efficient manner (closed-form or gradient-based). Due to decomposability of the Log-Likelihood, local optimization ensures global optimization.

We also proposed techniques for exact score-based structure learning. We focused on the class of AP-PLP(2) and complete data. Experiments have shown that learned PLPs contain less parameters than the corresponding CPT-based Bayesian networks, as intuitively expected, and better accuracy over testing sets, indicating an improvement in generalization performance. However, depending on the context, the restriction of having at most two parents per variable may be too strong.

As mentioned in the Introduction, there seems to be a trade-off between prediction accuracy and interpretability. But the challenge is: when should we prioritize accuracy and when should we prioritize interpretability instead? This question inspires a final reflection on why we should be interested in learning PLPs. Logical reasoning may not contribute to identifying objects, but it may help a human user to understand how these objects relate to each other. There is certainly much space for further investigation in this direction.

REFERENCES

[1] Kevin Bache and Moshe Lichman. UCI machine learning repository. 2013.

[2] F Bergadano, D Gunetti, M Nicosia, G Ruffo, et al. Learning logic programs with negation as failure. *Advances in inductive logic programming*, pages 107–123, 1996.

[3] Anh Tuan Bui and Chi-Hyuck Jun. Learning bayesian network structure using markov blanket decomposition. *Pattern Recognition Letters*, 33(16):2134–2140, 2012.

[4] Jorge Casillas, Oscar Cordón, Francisco Herrera Triguero, and Luis Magdalena. *Interpretability issues in fuzzy modeling*, volume 128. Springer, 2013.

[5] Keith L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322. Springer, 1978.

[6] Mark G Core, H Chad Lane, Michael Van Lent, Dave Gomboc, Steve Solomon, and Milton Rosenberg. Building explainable artificial intelligence systems. In *AAAI*, pages 1766–1773, 2006.

[7] Fabio Gagliardi Cozman and Denis Deratani Mauá. On the semantics and complexity of probabilistic logic programs. *arXiv preprint arXiv:1701.09000*, 2017.

[8] Adnan Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[9] Francisco H. O. V. de Faria, Fabio G. Cozman, and Denis D. Mauá. Learning probabilistic logic programs by score maximization: some tractable cases. In *11th International Conference on Scalable Uncertainty Management (SUM-17)*, in press.

[10] Francisco Henrique Otte Vieira de Faria, Arthur Colombini Gusmão, Glauber De Bona, Fabio Gagliardi Cozman, and Denis Deratani Mauá. Parameter learning in problog with probabilistic rules. In *5th Symposium on Knowledge Discovery, Mining and Learning (KDMiLe-2017)*, in press.

[11] Francisco Henrique Otte Vieira de Faria, Arthur Colombini Gusmão, Fabio Gagliardi Cozman, and Denis Deratani Mauá. Speeding-up ProbLog's parameter learning. In *7th International Workshop on Statistical Relational AI (StaRAI-17)*, in press.

[12] Luc De Raedt, Anton Dries, Ingo Thon, Guy Van den Broeck, and Mathias Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *International Joint Conference on Artificial Intelligence*, 2015.

[13] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A probabilistic prolog and its application in link discovery. 2007.

[14] Amit Dhurandhar, Vijay Iyengar, Ronny Luss, and Karthikeyan Shanmugam. A formal framework to characterize interpretability of procedures. 2017.

[15] Daan Fierens, Guy Van den Broeck, Joris Renkens, Dimitar Shrerionov, Bernd Gutmann, Gerda Janssens, and Luc de Raedt. Inference and learning in probabilistic logic programs using weighted Boolean formulas. *Theory and Practice of Logic Programming*, 15(3):358–401, 2014.

[16] Nir Friedman, Iftach Nachman, and Dana Peér. Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.

[17] Norbert Fuhr. Probabilistic Datalog — a logic for powerful retrieval methods. In *Conference on Research and Development in Information Retrieval*, pages 282–290, Seattle, Washington, 1995.

[18] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[19] Marian B Gorzałczany and Filip Rudziński. Accuracy vs. interpretability of fuzzy rule-based classifiers: an evolutionary approach. In *Swarm and Evolutionary Computation*, pages 222–230. Springer, 2012.

[20] Bernd Gutmann, Ingo Thon, and Luc De Raedt. Learning the parameters of probabilistic logic programs from interpretations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 581–596. Springer, 2011.

[21] Joseph Y. Halpern. *Reasoning about Uncertainty*. MIT Press, Cambridge, Massachusetts, 2003.

[22] John D Head and Michael C Zerner. A Broyden—Fletcher—Goldfarb—Shanno optimization procedure for molecular geometries. *Chemical physics letters*, 122(3):264–270, 1985.

[23] Christopher John Hogger. Essentials of logic programming. 1990.

[24] Hisao Ishibuchi and Yusuke Nojima. Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. *International Journal of Approximate Reasoning*, 44(1):4–31, 2007.

[25] Manfred Jaeger. Model-theoretic expressivity analysis. In L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton, editors, *Probabilistic Inductive Logic Programming*. Springer, 2008.

[26] Kristian Kersting and Luc De Raedt. 1 Bayesian logic programming: Theory and tool. *Statistical Relational Learning*, page 291, 2007.

[27] Angelika Kimmig, Bart Demoen, Luc De Raedt, Vitor Santos Costa, and Ricardo Rocha. On the implementation of the probabilistic logic programming language ProbLog. *Theory and Practice of Logic Programming*, 11(2-3):235–262, 2011.

[28] Timo JT Koski and John Noble. A review of Bayesian networks and structure learning. *Mathematica Applicanda*, 40(1):51–103, 2012.

[29] Robert Kowalski and Fariba Sadri. From logic programming towards multi-agent systems. *Annals of Mathematics and Artificial Intelligence*, 25(3):391–419, 1999.

[30] John W Lloyd. *Foundations of logic programming*. Springer Science & Business Media, 2012.

[31] Tom Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[32] Nils J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.

[33] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, California, 1988.

[34] David Poole. Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

[35] David Poole. Abducing through negation as failure: stable models within the independent choice logic. *Journal of Logic Programming*, 44:5–35, 2000.

[36] David Poole. The Independent Choice Logic and beyond. In L. De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton, editors, *Probabilistic Inductive Logic Programming*, volume 4911 of *Lecture Notes in Computer Science*, pages 222–243. Springer, 2008.

[37] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[38] Luc De Raedt, Paolo Frasconi, Kristian Kersting, and Stephen Muggleton. *Probabilistic Inductive Logic Programming*. Springer, 2010.

[39] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd ed.)*. Upper Saddle River, Prentice Hall, New Jersey, 1995.

[40] Taisuke Sato. A statistical learning method for logic programs with distribution semantics. In *Int. Conference on Logic Programming*, pages 715–729, 1995.

[41] Taisuke Sato and Yoshitaka Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[42] Marco Scutari. Bayesian network constraint-based structure learning algorithms: Parallel and optimised implementations in the bnlearn r package. *arXiv preprint arXiv:1406.7648*, 2014.

[43] P. Spirtes, C. Glymour, and R. Scheines. *Causality, Prediction, and Search*. Springer-Verlag, New York, 1993.

[44] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[45] Peter van Beek and Hella-Franziska Hoffmann. Machine learning of Bayesian networks using constraint programming. In *International Conference on Principles and Practice of Constraint Programming*, pages 429–445. Springer, 2015.

[46] Maarten H Van Emden and Robert A Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM)*, 23(4):733–742, 1976.

[47] David J Wales and Jonathan PK Doye. Global optimization by Basin-Hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116, 1997.

[48] Yiwei Zhang, Joseph P Mcgeehan, Edward M Regan, Stephen Kelly, and Jose Luis Nunez-Yanez. Biophysically accurate floating point neuroprocessors for reconfigurable logic. *IEEE Transactions on Computers*, 62(3):599–608, 2013.

APPENDIX A – EXACT SOLUTIONS FOR CLASS A-PLPS

This appendix lists all possible likelihood expressions one could obtain for a combination of at most three rules entailing the same predicate. It also indicates which expressions can be maximized in closed form (exact solution), and which cannot (numeric solution).

Table 10: likelihood expressions for combinations of at most three rules entailing the same predicate [Part 1].

Likelihood Expression	Solution
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}$	Exact
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}$	Exact
$L(R,D) = \theta_{1,2,3}^{a_0}.(1-\theta_{1,2,3})^{a_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2,3}^{b_0}.(1-\theta_{1,2,3})^{b_1}$	Exact
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{1,3}^{b_0}.(1-\theta_{1,3})^{b_1}$	Exact
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}$	Exact
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{1,2,3}^{b_0}.(1-\theta_{1,2,3})^{b_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2,3}^{c_0}.(1-\theta_{1,2,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{1,2,3}^{c_0}.(1-\theta_{1,2,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,2,3}^{c_0}.(1-\theta_{1,2,3})^{c_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}$	Exact
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}$	Exact
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{1,3}^{b_0}.(1-\theta_{1,3})^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}$	Numeric
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{1,3}^{b_0}.(1-\theta_{1,3})^{b_1}.\theta_{1,2,3}^{c_0}.(1-\theta_{1,2,3})^{c_1}$	Numeric
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,2,3}^{c_0}.(1-\theta_{1,2,3})^{c_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_{1,2}^{a_0}.(1-\theta_{1,2})^{a_1}.\theta_{1,3}^{b_0}.(1-\theta_{1,3})^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,2,3}^{d_0}.(1-\theta_{1,2,3})^{d_1}$	Numeric

Table 11: likelihood expressions for combinations of at most three rules entailing the same predicate [Part 2].

Likelihood Expression	Solution
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,3}^{e_0}.(1-\theta_{1,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}.\theta_{1,2}^{e_0}.(1-\theta_{1,2})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{1,2}^{b_0}.(1-\theta_{1,2})^{b_1}.\theta_{2,3}^{c_0}.(1-\theta_{2,3})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_{2,3}^{b_0}.(1-\theta_{2,3})^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{1,2,3}^{e_0}.(1-\theta_{1,2,3})^{e_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,3}^{e_0}.(1-\theta_{1,3})^{e_1}.\theta_{2,3}^{f_0}.(1-\theta_{2,3})^{f_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,3}^{e_0}.(1-\theta_{1,3})^{e_1}.\theta_{1,2,3}^{f_0}.(1-\theta_{1,2,3})^{f_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}.\theta_{1,2,3}^{f_0}.(1-\theta_{1,2,3})^{f_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}.\theta_{1,2,3}^{f_0}.(1-\theta_{1,2,3})^{f_1}$	
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,2}^{c_0}.(1-\theta_{1,2})^{c_1}.\theta_{1,3}^{d_0}.(1-\theta_{1,3})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}.\theta_{1,2,3}^{f_0}.(1-\theta_{1,2,3})^{f_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{2,3}^{e_0}.(1-\theta_{2,3})^{e_1}.\theta_{1,2,3}^{f_0}.(1-\theta_{1,2,3})^{f_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_{1,3}^{c_0}.(1-\theta_{1,3})^{c_1}.\theta_{2,3}^{d_0}.(1-\theta_{2,3})^{d_1}.\theta_{1,2}^{e_0}.(1-\theta_{1,2})^{e_1}.\theta_{1,2,3}^{f_0}.(1-\theta_{1,2,3})^{f_1}$	Numeric
$L(R,D) = \theta_1^{a_0}.(1-\theta_1)^{a_1}.\theta_2^{b_0}.(1-\theta_2)^{b_1}.\theta_3^{c_0}.(1-\theta_3)^{c_1}.\theta_{1,2}^{d_0}.(1-\theta_{1,2})^{d_1}.\theta_{1,3}^{e_0}.(1-\theta_{1,3})^{e_1}.\theta_{2,3}^{f_0}.(1-\theta_{2,3})^{f_1}.\theta_{1,2,3}^{g_0}.(1-\theta_{1,2,3})^{g_1}$	Numeric

APPENDIX B – EXACT SOLUTIONS FOR CLASS A-PLP(2)

This appendix lists all likelihood expressions and corresponding solutions for combinations of rules entailing a variable X_0 under the restriction of having at most 2 parents, X_1 and X_2.

Table 12: Exact solutions for combinations of at most three rules entailing the same predicate.

Likelihood Expression	Solution
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}$	$\theta_1 = \frac{a_0}{a_0+a_1}$
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_2^{b_0}(1-\theta_2)^{b_1}$	$\theta_1 = \frac{a_0}{a_0+a_1}, \theta_2 = \frac{b_0}{b_0+b_1}$
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_{1,2}^{b_0}(1-\theta_{1,2})^{b_1}$	$\theta_1 = \frac{a_0}{a_0+a_1}, \theta_2 = \frac{a_1 b_0 - a_0 b_1}{a_1 b_0 + a_1 b_1}$
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_2^{b_0}(1-\theta_2)^{b_1}\theta_{1,2}^{c_0}(1-\theta_{1,2})^{c_1}$	Numeric
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_2^{b_0}(1-\theta_2)^{b_1}\theta_3^{c_0}(1-\theta_3)^{c_1}$	$\theta_1 = \frac{a_0}{a_0+a_1}, \theta_2 = \frac{b_0}{b_0+b_1}, \theta_3 = \frac{c_0}{c_0+c_1}$
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_{1,2}^{b_0}(1-\theta_{1,2})^{b_1}\theta_{1,3}^{c_0}(1-\theta_{1,3})^{c_1}$	$\theta_1 = \frac{a_0}{a_0+a_1}, \theta_2 = \frac{a_1 b_0 - a_0 b_1}{a_1 b_0 + a_1 b_1}, \theta_3 = \frac{a_1 c_0 - a_0 c_1}{a_1 c_0 + a_1 c_1}$
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_2^{b_0}(1-\theta_2)^{b_1}\theta_{1,2}^{c_0}(1-\theta_{1,2})^{c_1}\theta_3^{d_0}(1-\theta_3)^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_{1,2}^{b_0}(1-\theta_{1,2})^{b_1}\theta_{1,3}^{c_0}(1-\theta_{1,3})^{c_1}\theta_{1,2,3}^{d_0}(1-\theta_{1,2,3})^{d_1}$	Numeric
$L(R,D) = \theta_1^{a_0}(1-\theta_1)^{a_1}\theta_2^{b_0}(1-\theta_2)^{b_1}\theta_{1,3}^{c_0}(1-\theta_{1,3})^{c_1}\theta_{2,3}^{d_0}(1-\theta_{2,3})^{d_1}$	Numeric

APPENDIX C – POSSIBLE COMBINATIONS OF RULES FOR CLASS AP-PLP(2)

This appendix lists all possible combinations of rules entailing a variable X_0 under the restriction of having at most 2 parents, X_1 and X_2.

Table 13: Possible combinations of rules for family $\langle X_0, \text{PA}[X_0] = \{X_1, X_2\}\rangle$ [Part 1].

$\text{PA}[X_0]$	Combination of rules
$\{\}$	$\theta_0 :: \mathsf{X_0}$
$\{\mathsf{X_1}\}$	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_1 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_1 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$
	$\theta_1 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_1 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_1 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_2 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$
$\{\mathsf{X_1}, \mathsf{X_2}\}$	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_2}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_2}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_2}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_2}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \mathsf{X_1}, \overline{\mathsf{X_2}}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$
	$\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \mathsf{X_2}$ $\quad$ $\theta_0 :: \mathsf{X_0} :\!\!- \overline{\mathsf{X_1}}, \overline{\mathsf{X_2}}$

Table 14: Possible combinations of rules for family $\langle X_0, \text{PA}[X_0] = \{X_1, X_2\}\rangle$ [Part 2].

$\text{PA}[X_0]$	Combination of rules		
$\{\mathsf{X}_1, \mathsf{X}_2\}$	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$

Table 15: Possible combinations of rules for family $\langle X_0, \text{PA}[X_0] = \{X_1, X_2\}\rangle$ [Part 3].

$\text{PA}[X_0]$	Combination of rules			
$\{\mathsf{X}_1, \mathsf{X}_2\}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$	
	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \mathsf{X}_1, \overline{\mathsf{X}_2}$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \mathsf{X}_2$	$\theta_0 :: \mathsf{X}_0 :- \overline{\mathsf{X}_1}, \overline{\mathsf{X}_2}$

APPENDIX D – PRUNING RESULTS FOR CLASS A-PLP(2)

This appendix shows which likelihood expressions can be pruned and which remain after pruning.

Table 16: Combinations of rules before and after Pruning [Part 1].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$	$1-\theta_{1,3}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	$\theta_{1,3}$	$\theta_{1,3}$	$\theta_{1,2}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A2}$	$1-\theta_1$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	θ_1	θ_1	$\theta_{1,2}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A2}}$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_1$	$\theta_{1,2}$	$\theta_{1,2}$	θ_1	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$	$1-\theta_2$	$1-\theta_2$	$1-\theta_1$	$1-\theta_1$	θ_2	θ_2	θ_1	θ_1

Table 17: Combinations of rules before and after Pruning [Part 2].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$\theta_{1,3}$	$\theta_{1,2}$	$\theta_{1,3}0$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,2}$	θ_1	$\theta_{1,2}$	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_3}}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_1$	$\theta_{1,2}$	θ_1	$\theta_{1,2}$	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_3}}$	$1-\theta_2$	$1-\theta_1$	$1-\theta_2$	$1-\theta_1$	θ_2	θ_1	θ_2	θ_1

Table 18: Combinations of rules before and after Pruning [Part 3].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	1	1	$1-\theta_{1,3}$	$1-\theta_{1,2}$	0	0	$\theta_{1,3}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$	1	1	$1-\theta_1$	$1-\theta_{1,2}$	0	0	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	1	1	$1-\theta_{1,2}$	$1-\theta_1$	0	0	$\theta_{1,2}$	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	1	1	$1-\theta_2$	$1-\theta_1$	0	0	θ_2	θ_1

Table 19: Combinations of rules before and after Pruning [Part 4].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	1	1	$\theta_{1,3}$	$\theta_{1,2}$	0	0
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,2}$	1	1	θ_1	$\theta_{1,2}$	0	0
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2}$	$1-\theta_1$	1	1	$\theta_{1,2}$	θ_1	0	0
Combination left								
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_2$	$1-\theta_1$	1	1	θ_2	θ_1	0	0

Table 20: Combinations of rules before and after Pruning [Part 5].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	1	$1-\theta_{1,3}$	1	$1-\theta_{1,2}$	0	$\theta_{1,3}$	0	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$	1	$1-\theta_1$	1	$1-\theta_{1,2}$	0	θ_1	0	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_{1,2}$	1	$1-\theta_1$	0	$\theta_{1,2}$	0	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_2$	1	$1-\theta_1$	0	θ_2	0	θ_1

Table 21: Combinations of rules before and after Pruning [Part 6].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	1	$1-\theta_{1,2}$	1	$\theta_{1,3}$	0	$\theta_{1,2}$	0
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_1$	1	$1-\theta_{1,2}$	1	θ_1	0	$\theta_{1,2}$	0
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2}$	1	$1-\theta_1$	1	$\theta_{1,2}$	0	θ_1	0
Combination left								
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_2$	1	$1-\theta_1$	1	θ_2	0	θ_1	0

Table 22: Combinations of rules before and after Pruning [Part 7].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_{1,2,3}$	θ_1	θ_1	$\theta_{1,2}$	$\theta_{1,2,3}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_1$	$1-\theta_1$	$1-\theta_{1,2,3}$	$1-\theta_{1,2}$	θ_1	θ_1	$\theta_{1,2,3}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,3}$	$\theta_{1,2}$	$\theta_{1,2}$	θ_1	$\theta_{1,3}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	$1-\theta_1$	$\theta_{1,2}$	$\theta_{1,2}$	$\theta_{1,3}$	θ_1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_1$	$1-\theta_1$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	θ_1	θ_1	$\theta_{1,3}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$	$1-\theta_2$	$1-\theta_2$	$1-\theta_1$	$1-\theta_{1,3}$	θ_2	θ_2	θ_1	$\theta_{1,3}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_2$	$1-\theta_2$	$1-\theta_{1,3}$	$1-\theta_1$	θ_2	θ_2	$\theta_{1,3}$	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_1$	$1-\theta_1$	$1-\theta_3$	$1-\theta_2$	θ_1	θ_1	θ_3	θ_2

Table 23: Combinations of rules before and after Pruning [Part 8].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	θ_1	$\theta_{1,3}$	$\theta_{1,2}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	θ_1	$\theta_{1,2}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_{1,2}$	$1-\theta_{1,2,3}$	$1-\theta_1$	$1-\theta_1$	$\theta_{1,2}$	$\theta_{1,2,3}$	θ_1	θ_1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2,3}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_1$	$\theta_{1,2,3}$	$\theta_{1,2}$	θ_1	θ_1
$\theta_1 :: \mathsf{A_1} :- \mathsf{True}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_1$	$\theta_{1,3}$	$\theta_{1,2}$	θ_1	θ_1
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_2$	$1-\theta_{2,3}$	$1-\theta_1$	$1-\theta_1$	θ_2	$\theta_{2,3}$	θ_1	θ_1
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{2,3}$	$1-\theta_2$	$1-\theta_1$	$1-\theta_1$	$\theta_{2,3}$	θ_2	θ_1	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} :- \mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} :- \overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	$1-\theta_2$	$1-\theta_1$	$1-\theta_1$	θ_3	θ_2	θ_1	θ_1

Table 24: Combinations of rules before and after Pruning [Part 9].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,2,3}$	θ_1	$\theta_{1,2}$	θ_1	$\theta_{1,2,3}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,2,3}$	$1-\theta_1$	$1-\theta_{1,2}$	θ_1	$\theta_{1,2,3}$	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	$\theta_{1,2}$	θ_1	$\theta_{1,2}$	$\theta_{1,3}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$1-\theta_1$	$\theta_{1,2}$	$\theta_{1,3}$	$\theta_{1,2}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,3}$	$1-\theta_1$	$1-\theta_{1,2}$	θ_1	$\theta_{1,3}$	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$	$1-\theta_2$	$1-\theta_1$	$1-\theta_2$	$1-\theta_{1,3}$	θ_2	θ_1	θ_2	$\theta_{1,3}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_2$	$1-\theta_{1,3}$	$1-\theta_2$	$1-\theta_1$	θ_2	$\theta_{1,3}$	θ_2	θ_1
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_2$	$1-\theta_3$	$1-\theta_1$	$1-\theta_2$	θ_1	θ_3	θ_1	θ_2
Combination left								
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_2$	$1-\theta_3$	$1-\theta_1$	$1-\theta_2$	θ_1	θ_3	θ_1	θ_2

Table 25: Combinations of rules before and after Pruning [Part 10].

Combinations pruned	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	θ_1	$\theta_{1,2}$	$\theta_{1,3}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,2}$	$\theta_{1,3}$	$\theta_{1,2}$	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{1,2,3}$	$1-\theta_1$	$\theta_{1,2}$	θ_1	$\theta_{1,2,3}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2,3}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_1$	$\theta_{1,2,3}$	θ_1	$\theta_{1,2}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{True}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_1$	$1-\theta_{1,2}$	$1-\theta_1$	$\theta_{1,3}$	θ_1	$\theta_{1,2}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_2)$	$1-\theta_1$	$1-\theta_{2,3}$	$1-\theta_1$	θ_2	θ_1	$\theta_{2,3}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{2,3}$	$1-\theta_1$	$1-\theta_2$	$1-\theta_1$	$\theta_{2,3}$	θ_1	θ_2	θ_1
Combination left								
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	$1-\theta_1$	$1-\theta_2$	$1-\theta_1$	θ_3	θ_1	θ_2	θ_1

Table 26: Combinations of rules before and after Pruning [Part 11].

Combination of rules	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$	1	$1-\theta_2$	$1-\theta_1$	$1-\theta_{1,2,3}$	0	θ_2	θ_1	$\theta_{1,2,3}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	1	$1-\theta_2$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	0	θ_2	$\theta_{1,3}$	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_{2,3}$	$1-\theta_1$	$1-\theta_{1,2}$	0	$\theta_{2,3}$	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_3$	$1-\theta_1$	$1-\theta_{1,2}$	0	θ_3	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_3$	$1-\theta_{1,2}$	$1-\theta_1$	0	θ_3	$\theta_{1,2}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	1	$1-\theta_1$	$1-\theta_3$	$1-\theta_{1,2}$	0	θ_1	θ_3	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_{1,3}$	$1-\theta_2$	$1-\theta_1$	0	$\theta_{1,3}$	θ_2	θ_1
Combination left after Pruning								
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	1	$1-\theta_3$	$1-\theta_2$	$1-\theta_1$	0	θ_3	θ_2	θ_1

Table 27: Combinations of rules before and after Pruning [Part 12].

Combination of rules	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$	$1-\theta_2$	1	$1-\theta_{1,2}$	$1-\theta_{1,3}$	θ_2	0	$\theta_{1,2}$	$\theta_{1,3}$
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_2$	1	$1-\theta_{1,2,3}$	$1-\theta_1$	θ_2	0	$\theta_{1,2,3}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{2,3}$	1	$1-\theta_{1,2}$	$1-\theta_1$	$\theta_{2,3}$	0	$\theta_{1,2}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	1	$1-\theta_1$	$1-\theta_{1,2}$	θ_3	0	θ_1	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	1	$1-\theta_{1,2}$	$1-\theta_1$	θ_3	0	$\theta_{1,2}$	θ_1
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_1$	1	$1-\theta_{1,3}$	$1-\theta_2$	θ_1	0	$\theta_{1,3}$	θ_2
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	1	$1-\theta_1$	$1-\theta_2$	$\theta_{1,3}$	0	θ_1	θ_2
Combination left								
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	1	$1-\theta_2$	$1-\theta_1$	θ_3	0	θ_2	θ_1

Table 28: Combinations of rules before and after Pruning [Part 13].

Combination of rules	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,2}$	1	$1-\theta_{2,3}$	θ_1	$\theta_{1,2}$	0	$\theta_{2,3}$
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,2,3}$	1	$1-\theta_2$	θ_1	$\theta_{1,2,3}$	0	θ_2
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_{1,2}$	1	$1-\theta_2$	$\theta_{1,3}$	$\theta_{1,2}$	0	θ_2
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,3}$	1	$1-\theta_2$	θ_1	$\theta_{1,3}$	0	θ_2
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_1$	1	$1-\theta_2$	$\theta_{1,3}$	θ_1	0	θ_2
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	$1-\theta_1$	1	$1-\theta_{1,2}$	θ_3	θ_1	0	$\theta_{1,2}$
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	$1-\theta_{1,2}$	1	$1-\theta_1$	θ_3	$\theta_{1,2}$	0	θ_1
Combination left after Pruning								
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}, \mathsf{A_3}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	$1-\theta_2$	1	$1-\theta_1$	θ_3	θ_2	0	θ_1

Table 29: Combinations of rules before and after Pruning [Part 14].

Combination of rules	Likelihood of observations A_1, A_2, A_3							
	0,0,0	0,0,1	0,1,0	0,1,1	1,0,0	1,0,1	1,1,0	1,1,1
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2}$	$1-\theta_1$	$1-\theta_{2,3}$	1	$\theta_{1,2}$	θ_1	$\theta_{2,3}$	0
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_{1,2}$	$1-\theta_{1,3}$	$1-\theta_2$	1	$\theta_{1,2}$	$\theta_{1,3}$	θ_2	0
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,2,3}$	$1-\theta_1$	$1-\theta_2$	1	$\theta_{1,2,3}$	θ_1	θ_2	0
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_{1,3}$	$1-\theta_2$	1	θ_1	$\theta_{1,3}$	θ_2	0
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_1$	$1-\theta_2$	1	$\theta_{1,3}$	θ_1	θ_2	0
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$	$1-\theta_1$	$1-\theta_3$	$1-\theta_{1,2}$	1	θ_1	θ_3	$\theta_{1,2}$	0
$\theta_1 :: \mathsf{A_1} : -\overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_{1,3}$	$1-\theta_2$	$1-\theta_1$	1	$\theta_{1,3}$	θ_2	θ_1	0
Combination left after Pruning								
$\theta_1 :: \mathsf{A_1} : -\mathsf{A_2}, \overline{\mathsf{A_3}}$ $\theta_2 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \mathsf{A_3}$ $\theta_3 :: \mathsf{A_1} : -\overline{\mathsf{A_2}}, \overline{\mathsf{A_3}}$	$1-\theta_3$	$1-\theta_2$	$1-\theta_1$	1	θ_3	θ_2	θ_1	0