Alexander Pinl

Probability Propagation Nets

Unveiling Structure and Propagations
of Bayesian Networks by means of Petri Nets

Universität Koblenz Landau
Campus Koblenz
Fachbereich 4: Informatik
Institut für Softwaretechnik

Alexander Pinl

Probability Propagation Nets

Unveiling Structure and Propagations of Bayesian Networks by means of Petri Nets

29.11.2007

vom Promotionsausschuss des Fachbereichs 4: Informatik der Universität Koblenz-Landau zur Verleihung des akademischen Grades Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation

Vorsitzender des Promotionsausschusses: Prof. Dr. Dieter Zöbel
Vorsitzender der Promotionskommission: Prof. Dr. Thomas Burkhardt
Berichterstatter:
Prof. Dr. Kurt Lautenbach, Universität Koblenz-Landau
Prof. Dr. Wolfgang Fengler, Technische Universität Ilmenau
Datum der wissenschaftlichen Aussprache: 14. April 2008

To my parents.

Abstract

This work introduces a Petri net representation for the propagation of probabilities and likelihoods, which can be applied to probabilistic Horn abduction, fault trees, and Bayesian networks. These so-called "probability propagation nets" increase the transparency of propagation processes by integrating structural and dynamical aspects into one homogeneous representation. It is shown by means of popular examples that probability propagation nets improve the understanding of propagation processes – especially with respect to the Bayesian propagation algorithms – and thus are well suited for the analysis and diagnosis of probabilistic models. Representing fault trees with probability propagation nets transfers these possibilities to the modeling of technical systems.

Keywords

Petri nets, Bayesian networks, fault trees, probability, propagation, uncertain reasoning, diagnosis

Zusammenfassung

In der vorliegenden Arbeit wird eine Petri-Netz-Repräsentation für die Propagation von Wahrscheinlichkeiten und Evidenzen (Likelihoods) vorgestellt und auf probabilistische Horn-Abduktion sowie Fehlerbäume und Bayes-Netze angewendet. Diese sogenannten Wahrscheinlichkeits-Propagations-Netze (*probability propagation nets*) machen Propagations-Prozesse transparent, indem sie strukturelle und dynamische Aspekte in einer homogenen Darstellung vereinen. Anhand populärer Beispiele wird verdeutlicht, dass Wahrscheinlichkeits-Propagations-Netze die Propagations-Prozesse – besonders im Hinblick auf die Bayes-Netz-Algorithmik – anschaulich darstellen und gut nachvollziehbar machen, so dass sie sich für die Analyse und Diagnose probabilistischer Modelle eignen. Durch die Repräsentation von Fehlerbäumen mit Wahrscheinlichkeits-Propagations-Netzen können diese Vorzüge auf die Modellierung technischer Systeme übertragen werden.

Stichworte

Petri-Netze, Bayes-Netze, Fehlerbäume, Wahrscheinlichkeit, Propagation, unsicheres Schließen, Diagnose

Preface

This thesis has been written during my employment as a scientific assistant in the research group of Prof. Dr. Lautenbach at the University of Koblenz-Landau. After finishing my studies at this university and after a short time in a company, my return to the "Campus Koblenz" indicates that I have lots of positive associations with this university.

First of all the perfect relationship with my boss and mentor Prof. Dr. Kurt Lautenbach and the excellent working atmosphere within our research group stands out. I would like to thank him in the first place for his great support, for helpful discussions and for always having a sympathetic ear for me. No term could express his engagement better than the German word "Doktorvater".

I also appreciate Prof. Dr. Wolfgang Fengler's interest in the topic of this work – which became manifest in helpful discussions – and his willingness to be the second evaluator.

Thanks to Prof. Dr. Dr. h.c. Eckehard Schnieder and to my other colleagues at the Technical University of Braunschweig, especially Dr. Jörg Müller and Dr. Roman Slovák, for the great collaboration in the DFG-funded project "ToMASEn". It was a pleasure working with you.

Special thanks go to Dr. Stephan Philippi who helped me a lot, supported me especially at the beginning of my employment by contributing his experience, and who is very committed to our research group.

I thank my colleague Katharina Hupf for her helpfulness, her sympathetic ear for questions about contents or the English language and for the excellent working atmosphere in our office.

At this place I would like to say thank you to my colleagues and friends at our local computer center (GHRKO). You are the best.

A great support for me was my brother Markus. I am very grateful for his critical comments and especially the considerable amount of time he invested in helping and

supporting me by proofreading this thesis.

I sincerely thank my wife Rica. She always has supported and motivated me and therefore she also played an important role for this thesis. Not least her proofreading helped a lot.

I do not want to miss to thank my formative teachers of mathematics, physics and computer science at the Megina Gymnasium Mayen: Wolfgang Dötsch, Josef Stopperich and Rainer Werner. They were conductive to logical thinking and systematic solving of mathematical problems and thus they have prepared me for my studies in an excellent way, which I appreciated countless times.

Last but definitely not least I thank my family, my "family-in-law", but especially my parents for always backing me up, supporting me, for their encouragement and their love.

Koblenz, November 2007 Alexander Pinl

Contents

List of Figures

List of Tables

1. Introduction

This thesis deals with the representation of Bayesian networks and the propagation of probabilities and likelihoods with Petri nets. The aim is to couple structural aspects and the propagation algorithms of Bayesian networks in one homogeneous representation, thus improving transparency and creating a clear structure of propagation flows. Additionally, it is shown how these Petri nets can be used to represent fault trees. Figure 1.1 shows the relationship between the different graph classes used in this thesis, which are now further characterized.

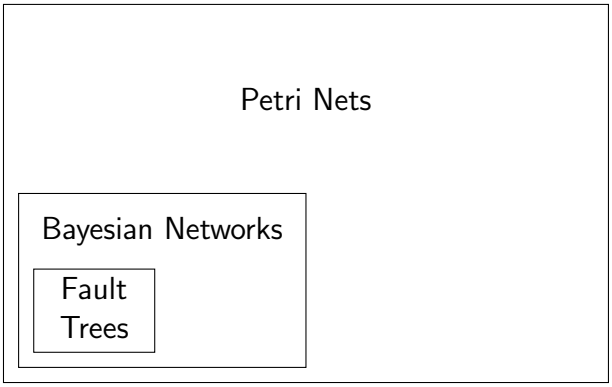


Figure 1.1.: Inclusion of Graph Classes

Petri Nets

In his PhD thesis [Petri62], Carl Adam Petri invented a new graph class which has been subject of intensive research and extension. After a short period of time, the corresponding graphs were called Petri nets. The basic idea may seem quite simple: A Petri net is a bipartite directed graph whose nodes are places and transitions. Places can hold (anonymous) objects called tokens which can be removed and created by transitions according to certain rules. Thus, transitions are responsible for representing dynamics, whereas places represent a distributed state or a situation.

It may show through that the integration of structure, situation and dynamics in one representation is a versatile approach. The basic concept of a Petri net has been enhanced for example by making the tokens distinguishable, which extends the expressiveness. Usually, the corresponding net classes are called *high-level Petri nets* (cf. [GenLau79, Jensen81, GenLau83, WagLew99, VanBil05]).

Bayesian Networks

Compared to Petri nets, Bayesian networks are more specialized. In their original form they focus on expressing and quantifying probabilistic dependencies between random variables. Bayesian networks are directed acyclic graphs (DAGs) whose nodes represent random variables. These variables have a finite set of discrete values. An edge from node A to B means, that B directly (probabilistically) depends on A. In addition, to every node a conditional probability table is attached which either quantifies the conditional probabilities of the corresponding variable given the values of the input variables or which contains the prior probabilities for the variable's elementary values if the node is a root node.

At present, Bayesian networks are used in a variety of application areas. Very popular is the so-called "Bayesian spam filter" [Sahami*98, Robi03], which is used to detect junk e-mails by a scoring algorithm relying on Bayesian techniques. Although this algorithm is implemented without the graphical representation of Bayesian networks, the underlying concepts and assumptions are identical.

In the research concerning operational risk and uncertain reasoning, Bayesian networks also play an important role. There is a lot of recent work investigating how Bayesian networks can benefit to solve problems in the context of operational risk (for example [Alex00, Yoon03, RaArGh05, PetSis06, SheWüt06]).

An outstanding theory related to Bayesian networks which is extensively used for example in biological applications – but also in the Bayesian spam filtering algorithms mentioned above – is the automated learning of network structure and parameters based on data [Heck99, Neap03]. Considering the very complex and large data which is collected in biological applications such as genomes or gene networks, automated organization and analysis of data is vital. Currently, a lot of cutting edge research is done in this area [Frie*00, Need*06, Need*07, WaChCl07].

Fault Trees

Fault trees are commonly used to model dependability in technical systems. There are different types of fault trees, for example static and the more expressive dynamic fault trees. Although the models built with static fault trees are quite simple since the functions expressing dependabilities between components are binary, this kind of modeling language is widely used and perfectly accepted in technical applications. Yet, the transformation of fault trees into Bayesian networks (cf. [Bobbio*99, PorBob99]) can grade up the modeling approach by adding certain features for example allowing easier diagnosis in the modeled systems.

By representing fault tree models with probability propagation nets via the transformation into Bayesian networks, all the advantages of the Petri net representation introduced in this work – such as transparency, clear structure, precise representation of causality, and integration of dynamics – can be transferred to the "fault tree world". Additionally, since other kinds of Petri nets can be used for modeling and controlling technical systems, a homogeneous representation with Petri nets offer opportunities to couple probabilistic models with established models in technical applications and process modeling.

Conceptual Formulation

As mentioned above, the primary objective of this thesis is to introduce a clear representation of Bayesian network propagations. Since Bayesian network models show the static probabilistic dependencies but the propagation, that means the message flow in Bayesian networks, is just defined by propagation algorithms and not represented in the network itself, the structure of Bayesian networks might be considered a bit meager. A Petri net representation of Bayesian networks helps to overcome this disadvantage, because Petri nets allow the integration of structure, situation and dynamics in one homogeneous representation.

To represent Bayesian networks together with their propagation flows, the so-called "probability propagation nets" are introduced in this thesis. As a precondition, the Bayesian networks have to be singly-connected, that means there must not be a loop contained in the Bayesian network which is to be transformed into a probability propagation net. The problem of eliminating loops in Bayesian networks is not trivial. In most tool implementations so-called "junction trees" combined with the propagation algorithms according to Shenoy and Shafer [SheSha88] or Lauritzen and Spiegelhal-

ter [LauSpi88, SpiLau90] are used. In contrast to that, the conditioning method (cf. [Pearl88]) is used in this thesis to eliminate loops, which results in singly-connected Bayesian networks that are closer to the original (multiply-connected) networks.

By transferring the complexity of eliminating loops into the Petri net structure as well as the tokens (or the marking), the linearity of propagation algorithms in probability propagation nets is preserved. Thereby, the complexity of "worst-case structures" is put into the size of the matrices being propagated.

It has to be pointed out that the Bayesian network propagation algorithms are not to be improved by the introduced Petri net representation. Yet, probability propagation nets can add transparency, define a clear structure, represent causality precisely, and integrate dynamics of the modeled system in a homogeneous representation, which may be useful for some real-life applications.

Outline

This thesis is organized as follows: In chapter 2 the basic concepts of Petri nets, probabilistic Horn abduction, Bayesian networks and fault trees are introduced. Low-level probability propagation nets, which allow for the representation of probabilistic Horn abduction, are defined in chapter 3. Foldings of these Petri nets, which are called high-level probability propagation nets, are capable of representing Bayesian networks and the message propagation as described in chapter 4. Chapter 5 shows the transformation of fault trees into probability propagation nets with the aid of the corresponding Bayesian network representations and gives some clues regarding future integration with Petri nets modeling and controlling technical systems. Conclusion and outlook finish the considerations in chapter 6.

2. Preliminaries

In this chapter, the relevant concepts of Petri nets, probabilistic Horn abduction, Bayesian networks, and fault trees will be introduced.

2.1. Petri Nets

Carl Adam Petri invented a special class of bipartite directed graphs which were later on called Petri nets. The interesting things about Petri nets are firstly, that they represent the static structure of a modeled system as well as its dynamics. Both aspects are expressed in a Petri net model via the graph structure and the so-called "token game". Secondly, and in contrast to for example finite automata, the state or situation of a Petri net is not limited to one marked node in the graph but is spread all over the net by its so-called "marking", thus representing a distributed state or situation. Thirdly, Petri nets are closely related to linear algebra and there are plenty of possibilities to use linear mathematical operations and related algorithms to analyze a model. Some of these analyses will be used exhaustively in this thesis. Fourthly, it is possible to define a hierarchical or modular refinement of Petri nets, which plays an important role for including abstraction in models with a considerable complexity — according to the principle "Divide and conquer". Fifthly, the original definition of Petri nets was extended to different classes of Petri nets. By that, concepts like time, fuzziness or stochastics can be represented and included in the analysis algorithms. The net classes can basically be devided into low-level Petri nets and high-level Petri nets. Usually, low-level nets can easily be represented by mathematical constructs and effectively be analyzed. The basic Petri net class will now be defined according to [Laut02]:

Definition 2.1 (Place/Transition Net)

1. *A* place/transition net *(*p/t-net*) is a quadruple* $\mathcal{N} = (S, T, F, W)$ *where*
   (a) S *and* T *are finite, non empty, and disjoint sets.* S *is the set of* places *(in figures represented by circles).* T *is the set of* transitions *(in figures represented by boxes).*
   (b) $F \subseteq (S \times T) \cup (T \times S)$ *is the set of directed* arcs.
   (c) $W : F \to \mathbb{N} \setminus \{0\}$ *assigns a* weight *to every arc. In case of* $W : F \to \{1\}$, $\mathcal{N} = (S, T, F)$ *is written as an abridgment.*

2. *The* preset *(*postset*) of a node* $x \in S \cup T$ *is defined as* ${}^\bullet x = \{y \in S \cup T \mid (y, x) \in F\}$ *(*$x^\bullet = \{y \in S \cup T \mid (x, y) \in F\}$*).*
   The preset *(*postset*) of a set* $H \subseteq S \cup T$ *is* ${}^\bullet H = \bigcup_{x \in H} {}^\bullet x$ *(*$H^\bullet = \bigcup_{x \in H} x^\bullet$*).*
   For all $x \in S \cup T$ *it is assumed that* $|{}^\bullet x| + |x^\bullet| \geq 1$ *holds; that means there are* no isolated nodes.

3. *A place* p *(transition* t*) is* shared *iff* $|{}^\bullet p| \geq 2$ *or* $|p^\bullet| \geq 2$ *(*$|{}^\bullet t| \geq 2$ *or* $|t^\bullet| \geq 2$*).*

4. *A place* p *is an* input (output) boundary place *iff* ${}^\bullet p = \varnothing$ *(*$p^\bullet = \varnothing$*).*

5. *A transition* t *is an* input (output) boundary transition *iff* ${}^\bullet t = \varnothing$ *(*$t^\bullet = \varnothing$*).*

This definition represents the structural aspects of a Petri net. The dynamics are defined on behalf of a special function called "marking" and a so-called "firing rule" as given in the next definition (again according to [Laut02]):

Definition 2.2
Let $\mathcal{N} = (S, T, F, W)$ *be a p/t-net.*

1. *A* marking *of* $\mathcal{N}$ *is a mapping* $M : S \to \mathbb{N}$. $M(p)$ *which indicates the number of* tokens *on* p *under* M. $p \in S$ *is* marked by M *iff* $M(p) \geq 1$. $H \subseteq S$ *is* marked by M *iff at least one place* $p \in H$ *is marked by* M. *Otherwise* p *and* H *are* unmarked*, respectively.*

2. *A transition $t \in T$ is* enabled by M*, in symbols $M[t\rangle$, iff*

$$\forall p \in {}^\bullet t : M(p) \geq W((p,t)).$$

3. *If $M[t\rangle$, the transition t may* fire *or* occur*, thus leading to a new marking M', in symbols $M[t\rangle M'$, with*

$$M'(p) := \begin{cases} M(p) - W((p,t)) & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ M(p) + W((t,p)) & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ M(p) - W((p,t)) + W((t,p)) & \text{if } p \in {}^\bullet t \cap t^\bullet \\ M(p) & \text{otherwise} \end{cases}$$

for all $p \in S$.

4. *The set of all markings* reachable *from a marking M_0, in symbols $[M_0\rangle$, is the smallest set such that*

$$M_0 \in [M_0\rangle$$
$$M \in [M_0\rangle \wedge M[t\rangle M' \Rightarrow M' \in [M_0\rangle.$$

$[M_0\rangle$ is also called the set of follower markings *of M_0.*

5. *$\sigma = t_1 \dots t_n$ is called a* firing sequence *or* occurrence sequence *for transitions $t_1, \dots, t_n \in T$ iff there exist markings $M_0, M_1, \dots, M_n$ such that*

$$M_0[t_1\rangle M_1[t_2\rangle \dots [t_n\rangle M_n \text{ holds;}$$

in short $M_0[\sigma\rangle M_n$. $M_0[\sigma\rangle$ denotes that σ starts from *M_0. The* firing count *$\bar{\sigma}(t)$ of t in σ indicates how often t occurs in σ. The (column) vector of firing counts is denoted by $\bar{\sigma}$.*

6. *The pair $(\mathcal{N}, M_0)$ for some marking M_0 of $\mathcal{N}$ is a* p/t-system *or a* marked p/t-net*. M_0 is the* initial marking*.*

7. *A marking $M \in [M_0\rangle$ is* reproducible *iff there exists a marking $M' \in [M\rangle$, $M' \neq M$ s.t. $M \in [M'\rangle$.*

8. *Moreover, the p-column-vector* $\mathbf{0}$ *stands for the* empty marking. *A p/t-net is* $\mathbf{0}$-reproducing *iff there exists a firing sequence* φ *such that* $\mathbf{0}[\varphi\rangle\mathbf{0}$. *A transition* t *is* $\mathbf{0}$-firable *iff* t *can be enabled by some follower marking of* $\mathbf{0}$.

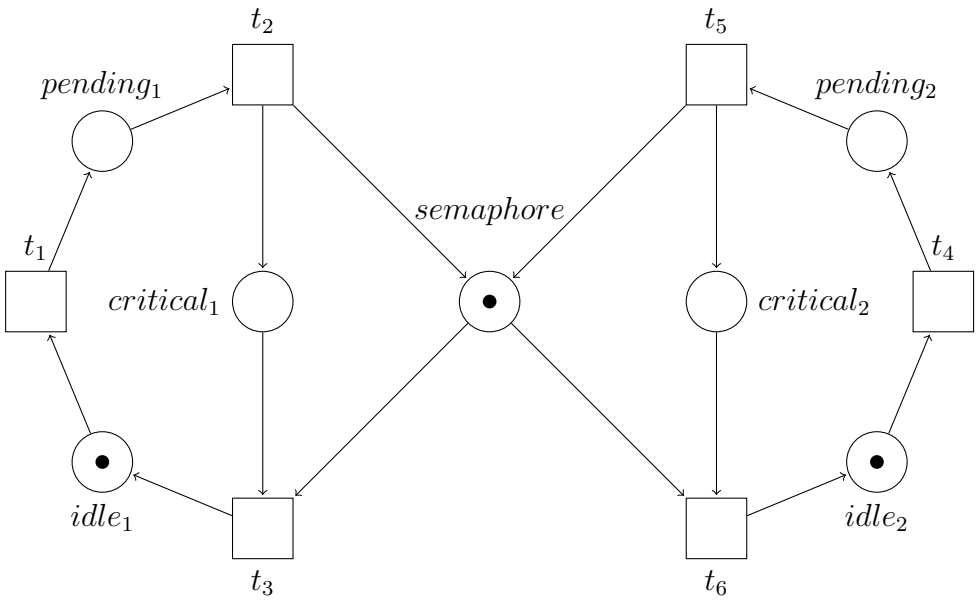


Figure 2.1.: A simple p/t-net modeling mutual exclusion for two processes

Example 2.1 (Mutual Exclusion, cf. [Kind02])
As stated in Definition 2.1, transitions are visually represented by boxes, places by circles. Figure 2.1, which is taken from [Kind02], shows a simple p/t-net model representing a mutual exclusion for two processes, that means it must be guaranteed that at most one process is running at a time.

To visualize the dynamics of the example net, tokens are usually represented by little filled circles and put on the places to indicate the current marking of a place. A certain firing sequence can now be investigated by simulating the net starting from the initial marking. One possible simulation run could be:

1. The initial marking indicates that both processes are idle. Place *semaphore* is marked, too. Transitions t_1 and t_4 are activated, either of them could fire.

2. Assume t_1 has fired. Now, $pending_1$, *semaphore* and $idle_2$ are marked with one

token. All other places are unmarked. Transitions t_2 and t_4 are both activated.

3. After that, t_4 fires. The places $pending_1$, $pending_2$ and *semaphore* are marked afterward. Transitions t_2 and t_5 are activated.

4. Now, let t_2 fire. It removes the tokens from $pending_1$ and *semaphore*, thus deactivating t_5 (because the token on *semaphore* is missing, now). It should be pointed out, that either t_2 or t_5 could have fired but no matter which transition fires, it deactivates the other one. By that, the mutual exclusion is guaranteed. After firing of t_2 only t_3 is activated. Only the places $critical_1$ and $pending_2$ are marked.

5. When t_3 fires, it removes the token from $critical_1$ and puts one token on $idle_1$ and *semaphore*, respectively. Additionally, $pending_2$ is still marked. Transitions t_1 and t_5 are both activated.

6. Analogously, after a subsequent firing of t_5 and t_6 the initial marking is reached.

Obviously, this is only one of many different possibilities to restore the initial marking or to run simulations on the net. □

Some properties and notations concerning p/t-nets remain to be defined:

Definition 2.3
Let $\mathcal{N} = (S, T, F, W)$ be a p/t-net;

1. *$\mathcal{N}$ is* pure *iff $\nexists (x, y) \in (S \times T) \cup (T \times S) : (x, y) \in F \wedge (y, x) \in F$.*

2. *A* place vector *($|S|$-vector) is a column vector $v : S \to \mathbb{Z}$ indexed by S.*

3. *A* transition vector *($|T|$-vector) is a column vector $\omega : T \to \mathbb{Z}$ indexed by T.*

4. *The* incidence matrix *of $\mathcal{N}$ is a matrix $[\mathcal{N}] : S \times T \to \mathbb{Z}$ indexed by S and T*

such that

$$[\mathcal{N}](p,t) = \begin{cases} -W((p,t)) & \text{if } p \in {}^\bullet t \setminus t^\bullet \\ W((t,p)) & \text{if } p \in t^\bullet \setminus {}^\bullet t \\ -W((p,t)) + W((t,p)) & \text{if } p \in {}^\bullet t \cap t^\bullet \\ 0 & \text{otherwise.} \end{cases}$$

v^t and A^t are the transposes *of a vector v and a matrix A, respectively. The columns of $[\mathcal{N}]$ are $|S|$-vectors, the rows of $[\mathcal{N}]$ are transposes of $|T|$-vectors. Markings are representable as $|S|$-vectors, firing count vectors as $|T|$-vectors.*

Example 2.2 (Mutual Exclusion: Incidence Matrix)
The mathematical representation of the mutual exclusion Petri net is the tuple $\mathcal{N} = (S, T, F)$ with

- $S = \{idle_1, idle_2, pending_1, pending_2, critical_1, critical_2, semaphore\}$
- $T = \{t_i \mid i = 1 \dots 6\}$
- $F = \{(idle_1, t_1), (t_1, pending_1), (pending_1, t_2), (t_2, critical_1), (critical_1, t_3), (t_3, idle_1), (t_3, semaphore), (idle_2, t_4), (t_4, pending_2), (pending_2, t_5), (t_5, critical_2), (critical_2, t_6), (t_6, idle_2), (t_6, semaphore), (semaphore, t_2), (semaphore, t_5)\}$

	t_1	t_2	t_3	t_4	t_5	t_6
$critical_1$		1	-1			
$critical_2$					1	-1
$idle_1$	-1		1			
$idle_2$				-1		1
$pending_1$	1	-1				
$pending_2$				1	-1	
$semaphore$		-1	1		-1	1

Table 2.1.: Incidence Mapping of Places and Transitions (Example 2.1)

Assuming the ordering of places and transitions according to Table 2.1, the inci-

dence matrix of the example net $\mathcal{N}$ is given by

$$[\mathcal{N}] = \begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 & -1 & 1 \end{bmatrix}$$

□

The incidence matrix is useful to analyze the net with respect to different structural properties by means of linear algebra. Amongst others, the following properties are related to the incidence matrix representations of the Petri nets (cf. [Laut02]):

Definition 2.4 (Invariants)
Let I be a place vector and J a transition vector of $\mathcal{N} = (S, T, F, W)$.

1. *I is a* place invariant *(p-invariant) iff $I \neq \mathbf{0}$ and $I^t \cdot [\mathcal{N}] = \mathbf{0^t}$*

2. *J is a* transition invariant *(t-invariant) iff $J \neq \mathbf{0}$ and $[\mathcal{N}] \cdot J = \mathbf{0}$*

3. *$\|I\| = \{p \in S \mid I(p) \neq 0\}$ and $\|J\| = \{t \in T \mid J(t) \neq 0\}$ are the* supports *of I and J, respectively.*

4. *A p-invariant I (t-invariant J) is*

   - non-negative *iff* $\forall p \in S : I(p) \geq 0$ *($\forall t \in T : J(t) \geq 0$)*
   - positive *iff* $\forall p \in S : I(p) > 0$ *($\forall t \in T : J(t) > 0$)*
   - minimal *iff I (J) is non-negative*
     and $\nexists$ p-invariant $I' : \|I'\| \subsetneq \|I\|$ ($\nexists$ t-invariant $J' : \|J'\| \subsetneq \|J\|$)
     and the greatest common divisor of all entries of I (J) is 1.

5. *The* net representation $\mathcal{N}_I = (S_I, T_I, F_I, W_I)$ *of a p-invariant* I *is defined by*

$$S_I := \|I\|$$
$$T_I := {}^\bullet S_I \cup S_I{}^\bullet$$
$$F_I := F \cap ((S_I \times T_I) \cup (T_I \times S_I))$$

W_I *is the restriction of* W *to* F_I.

6. *The* net representation $\mathcal{N}_J = (S_J, T_J, F_J, W_J)$ *of a t-invariant* J *is defined by*

$$T_J := \|J\|$$
$$S_J := {}^\bullet T_J \cup T_J{}^\bullet$$
$$F_J := F \cap ((S_J \times T_J) \cup (T_J \times S_J))$$

W_J *is the restriction of* W *to* F_J.

7. $\mathcal{N}$ *is* covered *by a p-invariant* I *(t-invariant* J*) iff* $\|I\| = S$ *(*$\|J\| = T$*).*

Proposition 2.1
Let $(\mathcal{N}, M_0)$ *be a p/t-system,* I *a p-invariant; then*

$$\forall M \in [M_0\rangle : I^t \cdot M = I^t \cdot M_0.$$

Proposition 2.2
Let $(\mathcal{N}, M_0)$ *be a p/t-system,* $M_1 \in [M_0\rangle$ *a follower marking of* M_0*, and* σ *a firing sequence that reproduces* M_1 : $M_1[\sigma\rangle M_1$*; then the firing count vector* $\bar{\sigma}$ *of* σ *is a t-invariant.*

Definition 2.5
Let $\mathcal{N} = (S, T, F, W)$ *be a p/t-net,* M_0 *a marking of* $\mathcal{N}$*, and* $r \geq \mathbf{0}$ *a* $|T|$*-vector;* r *is* realizable in $(\mathcal{N}, M_0)$ *iff there exists a firing sequence* σ *with* $M_0[\sigma\rangle$ *and* $\bar{\sigma} = r$.

Proposition 2.3
Let $\mathcal{N} = (S, T, F, W)$ be a p/t-net, M_1 and M_2 markings of $\mathcal{N}$, and σ a firing sequence s.t. $M_1[\sigma\rangle M_2$; then the linear relation

$$M_1 + [\mathcal{N}]\bar{\sigma} = M_2 \text{ holds.}$$

In the above linear relation, the state equation*, the order of transition firings is lost.*

For some Petri net classes which will be introduced later on, natural multisets and special vector products are needed:

Definition 2.6 (Natural Multiset)
Let A be a non-empty set;

- $m : A \to \mathbb{N}$ *is a* natural multiset over A*;*
- $\mathbb{M}(A)$ *is the* set of all natural multisets over A.

Definition 2.7
Let $A = (a_1, \ldots, a_m), B = (b_1, \ldots, b_n)$ be non-negative real vectors;

$$A \times B := (a_1 \cdot b_1, \ldots, a_1 \cdot b_n, a_2 \cdot b_1, \ldots, a_2 \cdot b_n, \ldots, a_m \cdot b_1, \ldots, a_m \cdot b_n).$$

If additionally $m = n$ holds,

$$A \circ B := (a_1 \cdot b_1, a_2 \cdot b_2, \ldots, a_m \cdot b_m).$$

When representing folded Bayesian networks with a special variant of high-level probability propagation nets, which is introduced in section 4.4, these products have to be generalized in order to be applied to matrices. This is defined as follows:

Definition 2.8
Let

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1k} \\ a_{21} & a_{22} & \dots & a_{2k} \\ a_{l1} & a_{l2} & \dots & a_{lk} \end{pmatrix}$$

be a $(l \times k)$*-matrix and*

$$B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ b_{l1} & b_{l2} & \dots & b_{lm} \end{pmatrix}$$

be a $(l \times m)$*-matrix.*

Then,

$$A \otimes B := \begin{pmatrix} (a_{11}, a_{12}, \dots, a_{1k}) \times (b_{11}, b_{12}, \dots, b_{1m}) \\ (a_{21}, a_{22}, \dots, a_{2k}) \times (b_{21}, b_{22}, \dots, b_{2m}) \\ (a_{l1}, a_{l2}, \dots, a_{lk}) \times (b_{l1}, b_{l2}, \dots, b_{lm}) \end{pmatrix}$$

is a $(l \times (k \cdot m))$*-matrix.*

If additionally $k = m$ *holds, then*

$$A \circ B := \begin{pmatrix} a_{11} \cdot b_{11} & a_{12} \cdot b_{12} & \dots & a_{1k} \cdot b_{1k} \\ a_{21} \cdot b_{21} & a_{22} \cdot b_{22} & \dots & a_{2k} \cdot b_{2k} \\ a_{l1} \cdot b_{l1} & a_{l2} \cdot b_{l2} & \dots & a_{lk} \cdot b_{lk} \end{pmatrix}$$

is a $(l \times k)$*-matrix.*

Remark 2.1
Obviously, $A \circ B = B \circ A$ holds for A and B being vectors of equal length as well as matrices of equal dimension. ◇

Example 2.3 (Mutual Exclusion: Invariants)
The net corresponding to the mutual exclusion example (see Figure 2.1) has two

minimal t-invariants J_1, J_2 and three minimal p-invariants I_1, I_2, I_3:

$$J_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad J_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}; \quad I_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}; \quad I_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \quad I_3 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

□

Remark 2.2
T-invariants indicate that starting from an initial marking, after complete simulation of the t-invariant this marking is reached again. P-invariants express that the weighted sum of tokens on the places included in the p-invariant is constant. The token count for each place included in the invariant's support is thereby multiplied by the corresponding entry of the invariant to calculate the weighted sum. ◇

2.2. Probabilistic Horn Abduction

Based on propositional logic, specifically Horn formulas, probabilistic Horn abduction (see [Poole93a, Poole93b, PorTor97]) assigns probabilities to assumptions and rules. It can be used for abduction and diagnosis in models governed by uncertainty. This section introduces relevant terms and gives an example (cf. [LaPhPi06]).

2.2.1. Canonical Net Representation of Horn Formulas

First, the basic concepts of Horn formulas and thus probabilistic Horn abduction have to be defined:

Definition 2.9
The alphabet *of propositional logic consists of* atoms $a, b, c, \ldots,$ operators $\neg, \vee, \wedge,$ *and* brackets (*and*).

The formulas α *are exactly the words which can be constructed by means of the*

following rules:

- *all atoms are formulas;*
- *if* α *is a formula, the* negation $\neg\alpha$ *is a formula, too;*
- *if* α *and* β *are formulas, the* conjunction $(\alpha \wedge \beta)$ *and the* disjunction $(\alpha \vee \beta)$ *are formulas, too.*

The implication $(\alpha \rightarrow \beta)$ *is an abbreviation for* $((\neg\alpha) \vee \beta)$*; the* biimplication $(\alpha \leftrightarrow \beta)$ *is an abbreviation for* $((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha))$*. For omitting brackets, the usual operator hierarchy is used.*

Definition 2.10
A literal *is an atom or the negation of an atom. A* clause *is a disjunction of literals. Usually,* $\square$ *denotes the* empty clause.

Definition 2.11
Let $\tau = \neg a_1 \vee \cdots \vee \neg a_m \vee b_1 \vee \cdots \vee b_n$ *be a clause;*
in set notation: $\tau = \neg A \cup B$ *for* $\neg A = \{\neg a_1, \ldots, \neg a_m\}$ *and* $B = \{b_1, \ldots, b_n\}$*;*

- τ *is a* fact clause *iff* $\neg A = \varnothing$,
- τ *is a* goal clause *iff* $B = \varnothing$,
- τ *is a* rule clause *iff* $\neg A \neq \varnothing \wedge B \neq \varnothing$,
- τ *is a* Horn clause *iff* $|B| \leq 1$.

Let α *be a conjunction of clauses, that means* α *is a conjunctive normal form (CNF) formula;*

- $\mathbb{A}(\alpha)$ *denotes the* set of atoms *of* α,
- $\mathbb{C}(\alpha)$ *denotes the* set of clauses *of* α,
- $\mathbb{F}(\alpha)$ *denotes the* set of fact clauses *of* α,
- $\mathbb{G}(\alpha)$ *denotes the* set of goal clauses *of* α,

- $\mathbb{R}(\alpha) := \mathbb{C}(\alpha) \setminus (\mathbb{F}(\alpha) \cup \mathbb{G}(\alpha))$ *denotes the* set of rule clauses *of* α*;*

α *is a* Horn formula *iff its clauses are Horn clauses.*

Remark 2.3
As the amount of positive literals of a Horn clause is limited by 1, Horn logic is decidable. ◇

Definition 2.12 (Complementary Atom)
Let $a \in \mathbb{A}(\alpha)$ *be an atom of a Horn formula* α*. If there is another atom* $\bar{a}$ *such that* $a \leftrightarrow \neg\bar{a}$ *and* $\neg a \leftrightarrow \bar{a}$ *holds,* $\bar{a}$ *(*a*) is called the* complementary atom of a ($\bar{a}$).

Remark 2.4
The property of an atom being the complementary atom of another one can either be expressed explicitly by the above equivalences or it can be an implicit assertion for the modeled system described for example in natural language. ◇

The following example taken from [PorTor97] illustrates the introduced terms.

Example 2.4 (Lack of Oil, cf. [PorTor97])
Let α be the Horn formula that is the conjunction of the clauses given in Table 2.2 where the atoms are *lo* (lack of oil), *nolo* (no lack of oil), *igir* (ignition irregular), *igno* (ignition normal), *owon* (oil warning lamp on), *owof* (oil warning lamp off), *acde* (acceleration delayed), and *acno* (acceleration normal).

Obviously, *lo* (*igir*, *owon*, *acde*) is the complementary atom of *nolo* (*igno*, *owof*, *acno*) and vice versa.

Clauses 1 to 12 are rule clauses as they consist of negative as well as positive literals. Clauses 13 to 16 are fact clauses as they have only one positive literal and no negative one. Clauses 17 to 20 build the set of goal clauses as they only consist of negative literals. The conjunction of these Horn clauses is a Horn formula. □

One can use p/t-nets to represent a Horn formula with Petri nets. The transformation of a logical Horn formula into the canonical net representation is described in detail in [Laut03b]. After defining the canonical net representation the example will

1	$\neg nolo \vee \neg igno \vee acno$	$\mathbb{R}(\alpha)$	13	$igno$	$\mathbb{F}(\alpha)$
2	$\neg nolo \vee \neg igir \vee acno$		14	$igir$	
3	$\neg nolo \vee \neg igno \vee acde$		15	$nolo$	
4	$\neg nolo \vee \neg igir \vee acde$		16	lo	
5	$\neg lo \vee \neg igno \vee acno$		17	$\neg acno$	$\mathbb{G}(\alpha)$
6	$\neg lo \vee \neg igir \vee acno$		18	$\neg acde$	
7	$\neg lo \vee \neg igno \vee acde$		19	$\neg owof$	
8	$\neg lo \vee \neg igir \vee acde$		20	$\neg owon$	
9	$\neg nolo \vee owof$				
10	$\neg lo \vee owof$				
11	$\neg nolo \vee owon$				
12	$\neg lo \vee owon$				

Table 2.2.: Horn Clauses (Example 2.4)

be continued by showing the corresponding p/t-net.

Definition 2.13 (Canonical Net Representation)
Let α be a CNF-formula and let $\mathcal{N}_\alpha = (S_\alpha, T_\alpha, F_\alpha)$ be a p/t-net;
$\mathcal{N}_\alpha$ is the canonical p/t-net representation *of α iff*

- *$S_\alpha = \mathbb{A}(\alpha)$ (set of atoms of α) and $T_\alpha = \mathbb{C}(\alpha)$ (set of clauses of α)*
- *for all $\tau = \neg a_1 \vee \cdots \vee \neg a_m \vee b_1 \vee \cdots \vee b_n \in \mathbb{C}(\alpha)$,*
  where $\{a_1, \ldots, a_m, b_1, \ldots, b_n\} \subseteq \mathbb{A}(\alpha)$, F_α is determined by
  ${}^\bullet\tau = \{a_1, \ldots, a_m\}$, $\tau^\bullet = \{b_1, \ldots, b_n\}$, that means the atoms $a_1, \ldots, a_m$ which are negated in the clause τ are the input places, the non-negated atoms $b_1, \ldots, b_n$ are the output places of the transition τ.

The transition τ is called fact (goal, rule) transition *iff the clause τ is a fact (goal, rule) clause.*

According to the definition, every clause is represented by a transition. Thus, for building the canonical net representation out of a Horn formula one could start with creating one transition for each clause and labeling it correspondingly. After that, for every transition its related clause is investigated. Every negated atom contained in the clause becomes an input place, every non-negated atom becomes an output place

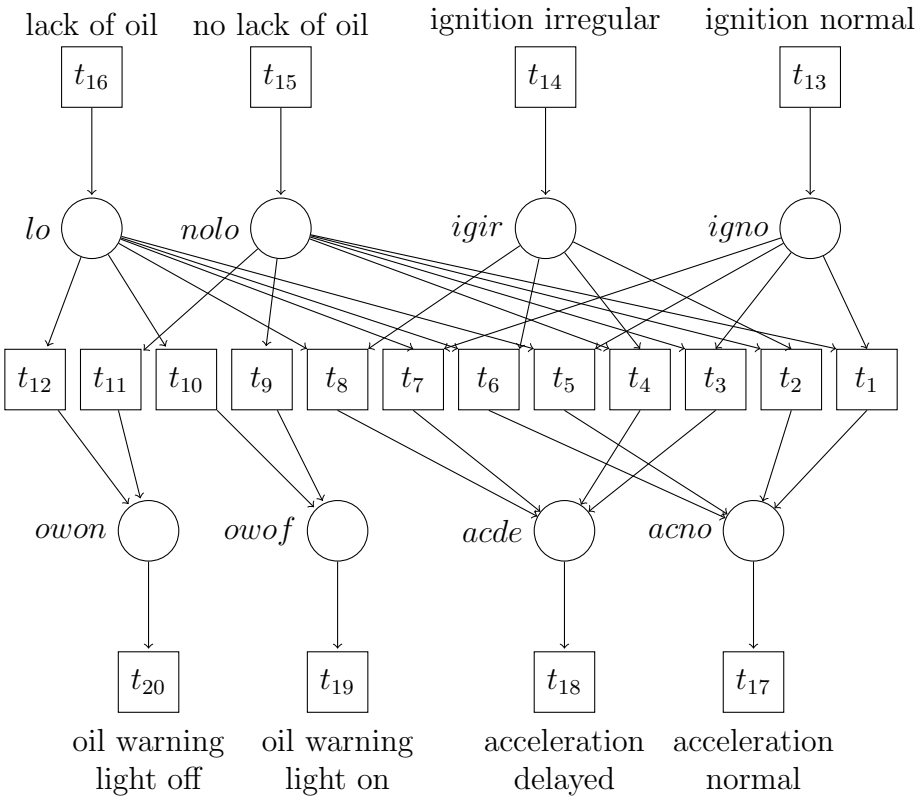


Figure 2.2.: $\mathcal{N}_\alpha$ of Example 2.4

of the transition. If the respective place already exists, just the corresponding edge has to be added.

Example 2.5 (Lack of Oil: Canonical Net Representation)
Applied to Example 2.4, the resulting p/t-net is shown in Figure 2.2. The transitions are labeled with a 't' indexed by the number of the related clause according to Table 2.2. □

Remark 2.5
In non-canonical p/t-net representations, the set of places S_α contains negated atoms (see [Laut03b]). ◇

For reasoning, abduction and diagnosis with Horn formulas, explanations for some clauses play an important role. Informally, an explanation for an outcome consists of

facts that – by applying the appropriate rules – lead to the outcome. Formally, this term is defined as follows:

Definition 2.14
Let α be a Horn formula,
let $H \subseteq \mathbb{F}(\alpha)$ be a set of fact clauses called the "assumable hypotheses",
let be $E \subseteq H$ the focused potential explanations *and $R \subseteq \mathbb{R}(\alpha) \cup (\mathbb{F}(\alpha) \setminus E)$,*
let $\varepsilon = \bigwedge_{\varphi \in E} \varphi$, $\varrho = \bigwedge_{\kappa \in R} \kappa$ be the corresponding Horn formulas,
let $\gamma = \neg g_1 \vee \cdots \vee \neg g_m$, $\gamma \in \mathbb{G}(\alpha)$ be a goal clause.
Then ε is an explanation (diagnosis) *of $\neg\gamma = g_1 \wedge \cdots \wedge g_m$ iff*

- *$\neg\gamma = g_1 \wedge \cdots \wedge g_m$ is a logical consequence of $\varepsilon \wedge \varrho$ and*
- *$\varepsilon \wedge \varrho$ is not contradictory.*

The representation of logical Horn formulas with Petri nets allows to find explanations or reasons of clauses by using t-invariants in the net representation. This is associated with an important theorem about the reproducibility of the empty marking in p/t-nets given and proved in [Laut03b]:

Theorem 2.1 Let α be a Horn formula and let $\mathcal{N}_\alpha = (S_\alpha, T_\alpha, F_\alpha)$ be its canonical p/t-net representation; then the following statements are equivalent:

(1) α is contradictory.

(2) $\mathcal{N}_\alpha$ is **0**-reproducing.

(3) $\mathcal{N}_\alpha$ has a t-invariant $R \geq 0$ with $R(g) > 0$ for some goal transition g.

(4) In $\mathcal{N}_\alpha$ a goal transition g is **0**-firable.

(5) In $\mathcal{N}_\alpha$ there exists a set Y of reverse paths from a goal transition to fact transitions such that with any transition t of a path of Y its incidenting places $p \in {}^\bullet t \cup t^\bullet$ are nodes of a path of Y, too.

Proof *See [Laut03b].* ■

Explanations for a goal clause can be found by collecting all t-invariants which have the corresponding transition in their support. This is similar to a proof by contradiction. Assume there is a t-invariant "going through" the goal transition. According to Theorem 2.1 the Horn formula is contradictory. As the goal transition represents the negated atoms, this contradiction is associated with the given facts, rules and the negated goals. If the facts are not contradictory, the only reason for the contradiction can be the negated goal atoms. Thus, the non-negated goals would not lead to that contradiction. An explanation for the non-negated goal atoms in this case would be the support of the investigated t-invariant intersected with the set of facts.

Remark 2.6
A minimal t-invariant of $\mathcal{N}_\alpha$ has only one goal transition g, because α is a Horn formula. $\diamond$

Example 2.6 (Lack of Oil: T-Invariants)
The t-invariants of $\mathcal{N}_\alpha$ (see Figure 2.2 and Table 2.3) are $\mathbf{0}$-reproducing which can easily be verified by simulation. There are four t-invariants passing through $t_{18} = \neg acde$ (for which $I(t_{18}) > 0$ holds), namely I_5, I_6, I_7, I_8.

According to Theorem 2.1 $((3),(2),(1))$ in the net representation of all these t-invariants (regarded as single canonical net representations) $acde$ is a logical consequence of the other clauses. For example I_7:

$$\gamma_7 = t_{18} = \neg acde$$
$$\varepsilon_7 = \{t_{16}, t_{14}\} = \{lo, igir\} = lo \wedge igir$$
$$\varrho_7 = t_8 = \neg lo \vee \neg igir \vee acde = lo \wedge igir \rightarrow acde$$

so, $\neg\gamma_7 = acde$ is a logical consequence of $\varepsilon_7 \wedge \varrho_7$.

Moreover, $\varepsilon_7 \wedge \varrho_7$ is not contradictory since in its canonical net representation t_{18} is missing such that the empty marking $\mathbf{0}$ is not reproducible (see Theorem 2.1). So ε_7 is an explanation of $\neg\gamma_7 = acde$.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
I_1					1							
I_2	1											
I_3						1						
I_4		1										
I_5							1					
I_6			1									
I_7								1				
I_8				1								
I_9										1		
I_{10}									1			
I_{11}												1
I_{12}											1	

	t_{13} *igno*	t_{14} *igir*	t_{15} *nolo*	t_{16} *lo*	t_{17} $\neg$*acno*	t_{18} $\neg$*acde*	t_{19} $\neg$*owof*	t_{20} $\neg$*owon*
I_1	1			1	1			
I_2	1		1		1			
I_3		1		1	1			
I_4		1	1		1			
I_5	1			1		1		
I_6	1		1			1		
I_7		1		1		1		
I_8		1	1			1		
I_9				1			1	
I_{10}			1				1	
I_{11}				1				1
I_{12}			1					1

Table 2.3.: T-invariants of $\mathcal{N}_\alpha$ (Example 2.5)

Alltogether,

$$\varepsilon_5 = \{lo, igno\} = lo \wedge igno$$
$$\varepsilon_6 = \{nolo, igno\} = nolo \wedge igno$$
$$\varepsilon_7 = \{lo, igir\} = lo \wedge igir$$
$$\varepsilon_8 = \{nolo, igir\} = nolo \wedge igir$$

are the explanations of *acde*. □

Remark 2.7
It may seem strange that $\varepsilon_6 = nolo \wedge igno$ is an explanation for a delayed acceleration, because there is no lack of oil nor is there an irregular ignition. When adding probabilities as described in the next section, the probability of this explanation will be 0.0 which is consequential. ◇

2.2.2. Extending Horn Formulas by Probabilities

By assigning probabilities to fact and rule clauses, a logical Horn formula can be enriched with aspects of uncertainty. Together with some other extensions and restrictions, this leads to a basis for probabilistic Horn abduction [Poole93a, Poole93b, PorTor97]:

Definition 2.15
Let α be a Horn formula and $P_\alpha : \mathbb{C}(\alpha) \to [0,1]$ a real function, called a probability function of α*;*

let $H \subseteq \mathbb{F}(\alpha)$ be a set of fact clauses; let $\{D_1, \ldots, D_n\}$ be a partition of H (that means $D_i \cap D_j = \varnothing$ for $i \neq j, \bigcup_{i=1}^{n} D_i = H$) where for all $D_i, 1 \leq i \leq n, \sum_{\varphi \in D_i} P_\alpha(\varphi) = 1$;

then the sets $D_1, \ldots, D_n$ are called disjoint classes*;*

let be $P_\alpha(\gamma) := 1$ for all goal clauses $\gamma \in \mathbb{G}(\alpha)$, let be $E \subseteq H$, $R \subseteq \mathbb{R}(\alpha) \cup \mathbb{F}(\alpha)$ and let $\varepsilon = \bigwedge_{\varphi \in E} \varphi$, $\varrho = \bigwedge_{\kappa \in R} \kappa$ be the corresponding Horn formulas, where ε is an explanation (diagnosis) of $\neg\gamma$.

The probability of ε *is given by $P_\alpha(\varepsilon \wedge \varrho)$. The problem to find explanations is the* probabilistic Horn abduction (PHA).

The 'Lack of Oil'-example is continued by assigning appropriate probabilities to each clause and defining the disjoint classes, thus making the formula analyzable by means of probabilistic Horn abduction.

Example 2.7 (Lack of Oil: Additional Probabilities)
According to [PorTor97], the probability for a lack of oil equals 0.4 ($P(lo) = 0.4$). The complementary probability is taken for 'no lack of oil' ($P(nolo) = 0.6$). The probabilities for *igir* and *igno* are 0.1 and 0.9, respectively.

The oil warning light deterministically depends on a lack of oil, that means the oil warning light is on, if and only if there is a lack of oil. If there is no lack of oil, the warning light is always off. This behavior is modeled by appropriate probability assignments to the corresponding rules. The probability assigned to the rule $lo \rightarrow owon$ equals 1, whereas the rule $lo \rightarrow owof$ has a probability of 0.

To express some kind of vagueness or uncertainty, rules may have probabilities between 0 and 1. For instance the probability of no lack of oil and irregular ignition leading to a delayed acceleration is set to 0.6 ($P(nolo \wedge igir \rightarrow acde) = 0.6$).

	Clause	Probability	Disjoint Class
1	$\neg nolo \vee \neg igno \vee acno$	1.0	
2	$\neg nolo \vee \neg igir \vee acno$	0.4	
3	$\neg nolo \vee \neg igno \vee acde$	0.0	
4	$\neg nolo \vee \neg igir \vee acde$	0.6	
5	$\neg lo \vee \neg igno \vee acno$	0.2	
6	$\neg lo \vee \neg igir \vee acno$	0.0	
7	$\neg lo \vee \neg igno \vee acde$	0.8	
8	$\neg lo \vee \neg igir \vee acde$	1.0	
9	$\neg nolo \vee owof$	1.0	
10	$\neg lo \vee owof$	0.0	
11	$\neg nolo \vee owon$	0.0	
12	$\neg lo \vee owon$	1.0	
13	$igno$	0.9	D_1
14	$igir$	0.1	D_1
15	$nolo$	0.6	D_2
16	lo	0.4	D_2
17	$\neg acno$	1.0	
18	$\neg acde$	1.0	
19	$\neg owof$	1.0	
20	$\neg owon$	1.0	

Table 2.4.: Probability Assignment and Disjoint Classes (Example 2.4)

The complete probability function is shown in Table 2.4. □

For probabilistic Horn formulas, the probability of explanations found by Probabilistic Horn Abduction correlates with the probability of explanations induced by **0**-reproducing t-invariants in the canonical net representation, as stated in the following definition:

Definition 2.16 (continuing Definition 2.15)
Let furthermore I be a t-invariant of the canonical net representation $\mathcal{N}_\alpha$ of α such that I performs the **0***-reproduction, induced by $\varepsilon \wedge \varrho \wedge \gamma$ being contradictory; then $\prod_{t \in \|I\| \setminus \{\gamma\}} P_\alpha(t)$ equals the* probabilities of ε and of $\neg\gamma$ *with respect to I.*

Thus, t-invariants are well-suited not only to find explanations for clauses of Horn formulas but also to calculate the corresponding probabilities in a quite simple way.

Remark 2.8
The atoms of α are now to be interpreted as random variables. The atoms of the fact clauses in a disjoint class D together with P_α form a finite probability space. ◇

Remark 2.9
For interpreting the probability function P_α, let $\tau = \neg a_1 \vee \cdots \vee \neg a_m \vee b$ be a Horn clause of α where $\neg A = \{\neg a_1, \ldots, \neg a_m\}, \quad B = \{b\}$:

- if τ is a fact clause ($\tau \in \mathbb{F}(\alpha), \neg A = \varnothing, B \neq \varnothing$), $P_\alpha(\tau)$ is the prior probability $P(b)$ of b,
- if τ is a rule clause ($\tau \in \mathbb{R}(\alpha), \neg A \neq \varnothing, B \neq \varnothing$), $P_\alpha(\tau)$ is the conditional probability $P(b \mid a_1, \ldots, a_m)$ of b given $a_1, \ldots, a_m$ (see Definition 2.17),
- if τ is a goal clause ($\tau \in \mathbb{G}(\alpha), \neg A \neq \varnothing, B = \varnothing$), the value of $P_\alpha(\tau)$ is not relevant for any calculation;
  from a logical point of view, the value 0 is justified because every **0**-reproduction is an indirect proof and results in a contradiction; the value 1 (see Definition 2.15) is a very handy compromise.

◇

The use of t-invariants to calculate probabilities will be demonstrated by continuing the example.

Example 2.8 (Lack of Oil: Calculating Probabilities)
The probability function P_α with two disjoint classes is shown in Table 2.4. Now, the probabilities of a delayed acceleration ($acde$) and its explanations are to be calculated

by means of the canonical Petri net representation in conjunction with the given probability function P_α according to Definition 2.16.

In simple cases like this one or when it is not necessary to watch the simulation of the (net representations of the) t-invariants, results can be calculated immediately:

$$P(\varepsilon_i) = \prod_{t \in \|I_i\|} P_\alpha(t).$$

(Please note that for the goal transitions $P_\alpha(t) = 1.0$ holds.)

$$P(\varepsilon_5) = 0.9 \cdot 0.4 \cdot 0.8 \cdot 1.0 = 0.288 \text{ (max.)}$$
$$P(\varepsilon_6) = 0.9 \cdot 0.6 \cdot 0.0 \cdot 1.0 = 0.0$$
$$P(\varepsilon_7) = 0.1 \cdot 0.4 \cdot 1 \cdot 1.0 = 0.04$$
$$P(\varepsilon_8) = 0.1 \cdot 0.6 \cdot 0.6 \cdot 1.0 = 0.036$$

$P(acde)$ sums up to 0.364. In case of simulating the four t-invariants, transition t_{18} (acceleration delayed) would fire for $ad = 0.288, 0, 0.04,$ and 0.036.

Not very surprisingly, all of these values coincide with the values calculated by probabilistic Horn abduction in [PorTor97]. □

Until now, these probabilities are not represented in the Petri net itself. In chapter 3 an approach to directly integrate probabilities into the Petri net representation will be presented.

2.3. Bayesian Networks

In probability theory so-called "Bayesian networks" are commonly used to describe probabilistic dependencies between random variables qualitatively and quantitatively. Basically, Bayesian networks are directed acyclic graphs whose nodes represent probabilistic variables each one having distinct elementary events and probability tables attached. The notion of conditional probability thereby plays a very important role. Thus, the introduction of Bayesian networks given in the next section starts with the definition of conditional probability and probabilistic dependencies. Afterwards, Bayesian networks are defined and their functionality is described. The problem of

loopiness is addressed to in section 2.3.3.

2.3.1. Static Structure

Thomas Bayes (1702–1761) introduced a mathematical expression in [Bayes63][1] which nowadays is commonly used to describe conditional probabilities. In fact the Bayesian formula introduces a measure for the probability of an event provided that another event is given. This is expressed via the following ratio formula:

Definition 2.17 (Bayesian Conditional Probability, cf. [Pearl88])
Let A and B be some (stochastic) events.

$$P(A \mid B) := \frac{P(A \wedge B)}{P(B)}$$

is called the conditional probability of A given B.

- *If $P(A \mid B) = P(A)$ holds, A and B are called* independent.
- *If $P(A \mid B \wedge C) = P(A \mid C)$, A and B are called* conditionally independent given C.

Bayes' formula is a kind of 'diagnostic' probability. It answers the question, "Given a symptom B, how probable is the explanation (or diagnosis) A?" If there are different conceivable diagnoses, the other direction is interesting, too: "Given the diagnosis A_i, how probable is it that A_i causes B?" These two notions of conditional probabilities can be combined by the Bayesian rule:

$$\begin{aligned} & P(A \mid B) = \frac{P(A \wedge B)}{P(B)} \\ \Leftrightarrow \quad & P(A \wedge B) = P(A \mid B) \cdot P(B) \end{aligned}$$

[1] This essay written by Bayes was published two years after his death.

$$P(B \mid A) = \frac{P(B \wedge A)}{P(A)} = \frac{P(A \wedge B)}{P(A)}$$
$$\Leftrightarrow \quad P(A \wedge B) = P(B \mid A) \cdot P(A)$$

Therefore,

$$P(A \mid B) \cdot P(B) = P(B \mid A) \cdot P(A)$$
$$\Leftrightarrow \quad P(B \mid A) = \frac{P(A \mid B) \cdot P(B)}{P(A)}$$

As mentioned before, this 'causal' probability is a degree of confirmation that A is the cause for B. This is also called *likelihood*:

Definition 2.18 (Likelihood)
Let A be dependent on B, that means $P(A \mid B) \neq P(A)$.

Then,

$$L(A \mid B) := P(B \mid A)$$

is called the likelihood of A given B.

Based on the concepts of probabilities, conditional probabilities and likelihoods, *Bayesian networks*, a special class of directed acyclic graphs (DAGs), have been developed to model conditional dependencies between random variables. Basically, a variable which directly depends on other variables is represented as a child node of these variables. Conditionally independent variables have no parents. Probabilities are annotated in so-called "conditional probability tables (CPTs)". For root elements, the prior probabilities are taken, whereas for child nodes, the conditional probabilities determine the nodes' conditional probability tables. Formally, Bayesian networks are defined as follows (cf. [Pearl88, Jensen96, Neap90, Neap03]):

Definition 2.19 (Bayesian Network)
Let $\mathcal{B} = (R, E)$ be a directed acyclic graph with the set R of nodes and the set E of edges; let for every $r \in R \quad par(r)$ be the set of parent nodes of r;

$\mathcal{B}$ is a Bayesian network *(*BN*) iff R equals a set of random variables and to every*

$r \in R$ the table $P(r \mid par(r))$ of conditional probabilities is assigned. $P(r \mid par(r))$ indicates the prior probabilities of r if $par(r) = \varnothing$.

Remark 2.10
The following notations are commonly used for random variables A, B, C and stochastic events d, e, f with respect to a probability function P:

$$P(A \mid BC) := P(A \mid B \cap C)$$
$$P(d \mid e, f) := P(d \mid e \wedge f).$$

$\diamond$

Example 2.9 (Lack of Oil: Bayesian Network)
The directed acyclic graph in combination with the probabilities assigned to the nodes in Figure 2.3 is a Bayesian network $\mathcal{B}$. Furthermore, it is noted that messages π (probabilities) and λ (likelihoods) flow in both directions via the edges from node to node. Table 2.5 shows the conditional probability tables assigned to the nodes of the Bayesian network. $\square$

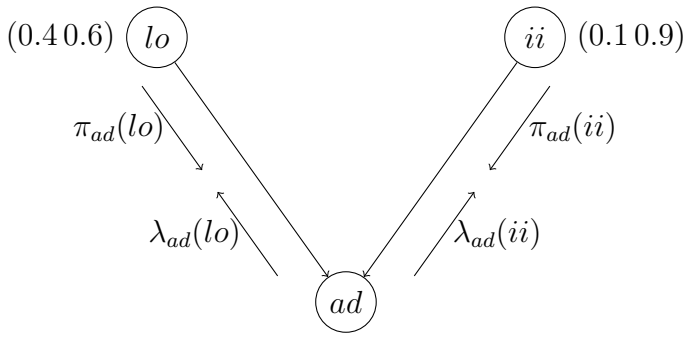


Figure 2.3.: Bayesian Network $\mathcal{B}$ of Example 2.9

2.3.2. Message Propagation

Obviously, a Bayesian network models static probabilistic dependencies between some random variables or random events. Given a Bayesian network consisting of different variable nodes and some edges, the question is how the prior probabilities of the input

$$P(lo) = \begin{array}{c|cc} lo & 1 & 0 \\ \hline & 0.4 & 0.6 \end{array} \qquad P(ii) = \begin{array}{c|cc} lo & 1 & 0 \\ \hline & 0.1 & 0.9 \end{array}$$

$$P(ad \mid lo, ii) = \begin{array}{cc|cc} & & \multicolumn{2}{c}{ad} \\ lo & ii & 1 & 0 \\ \hline 1 & 1 & 1.0 & 0.0 \\ 1 & 0 & 0.8 & 0.2 \\ 0 & 1 & 0.6 & 0.4 \\ 0 & 0 & 0.0 & 1.0 \end{array}$$

Table 2.5.: Conditional Probability Tables of $\mathcal{B}$

variables influence the other variables' probabilities. This is done by so-called "message propagations" according to special algorithms defined for Bayesian networks. The messages are divided into λ- and π-messages. Basically, λ-messages represent likelihood propagations and π-messages propagate probabilities.

The algorithms mentioned before are not visually represented in the Bayesian network graph. As they play a very important role, they will be completely quoted according to [Neap90] below. Before that, the two message types are defined. Neapolitan defines message propagation for causal networks which are more generalized than Bayesian networks. Yet, the algorithms apply to Bayesian networks, too, because they build a subclass of causal networks.

Remark 2.11
In the following definitions and rules $s(X)$ means the set of child nodes of a Bayesian network node X. Furthermore, $\lambda(x_i)$ ($\pi(x_i)$) is the current likelihood (probability) assigned to the elementary event x_i of X (cf. [Pearl88, Neap90]) with respect to the current situation of the Bayesian network. ◇

Definition 2.20 (λ Message, cf. [Neap90])
Let $C = (V, E, P)$ be a causal network in which the graph is a tree, W a subset of instantiated variables, $B \in V$ a variable with k possible values, and $C \in s(B)$ a child of B with m possible values. Then for $1 \leq i \leq k$, we define

$$\lambda_C(b_i) = \sum_{j=1}^{m} P(c_j|b_i)\lambda(c_j).$$

The entire vector of values $\lambda_C(b_i)$ *for* $1 \leq i \leq k$*, is called the* λ message from C to B *and is denoted* $\lambda_C(B)$.

Definition 2.21 (π Message, cf. [Neap90])
Let $C = (V, E, P)$ *be a causal network in which the graph is a tree,* W *a subset of instantiated variables,* $B \in V$ *a variable which is not the root, and* $A \in V$ *the father of* B*. Suppose* A *has* m *possible values. Then we define for* $1 \leq j \leq m$

$$\pi_B(a_j) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_j \\ 0 & \text{if } A \text{ is instantiated, but not for } a_j \\ \pi(a_j) \prod_{\substack{C \in s(A) \\ C \neq B}} \lambda_C(a_j) & \text{if } A \text{ is not instantiated,} \end{cases}$$

where $\lambda_C(a_j)$ *is defined in definition 2.20. Again, if there are no terms in the product, it is meant to represent the value* 1*. The entire vector of values,* $\pi_B(a_j)$ *for* $1 \leq j \leq m$*, is called the* π message from A to B *and is denoted* $\pi_B(A)$.

One term for Bayesian networks still has to be introduced: The so-called "belief" is the current feeling that a probabilistic variable has the corresponding values. Simply spoken, π-values ("probabilities") represent statistical data or experience, whereas λ-values ("likelihoods") represent evidences or observations. Beliefs combine these two notions. Together with the component product defined in Definition 2.7, belief can be defined as follows:

Definition 2.22 (Belief, cf. [Pearl88])
Let $\mathcal{B} = (R, E)$ *be a Bayesian network,* $X \in R$ *be a node of* $\mathcal{B}$ *and thus a random variable, and let* $\lambda(X)$ *and* $\pi(X)$ *be its current* λ*- and* π*-values.*

Then, with α *being a normalizing constant,*

$$BEL(X) := \alpha \left(\lambda(X) \circ \pi(X)\right)$$

is called the current belief of X.

Remark 2.12
Often, instead of $BEL(X)$ the notation $P(X)$ is used. This must not be mixed up

with the (prior) probability of X which usually is denoted $\pi(X)$ in this case. $\diamond$

Now, the different propagation rules are quoted from [Neap90]:

Operative Formulas

1. If B is a child of A, B has k possible values, A has m possible values, and B has one other parent D, with n possible values, then for $1 \leq j \leq m$ the λ message from B to A is given by

$$\lambda_B(a_j) = \sum_{p=1}^{n} \pi_B(d_p) \left(\sum_{i=1}^{k} P(b_i \mid a_j, d_p) \lambda(b_i) \right).$$

2. If B is a child of A and A has m possible values, then for $1 \leq j \leq m$ the π message from A to B is given by

$$\pi_B(a_j) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_j \\ 0 & \text{if } A \text{ is instantiated, but not for } a_j \\ \frac{P'(a_j)}{\lambda_B(a_j)} & \text{if } A \text{ is not instantiated,} \end{cases}$$

   where $P'(a_j)$ is defined to be the current conditional probability of a_j based on the variables thus far instantiated.

3. If B is a variable with k possible values, $s(B)$ is the set of B's children, then for $1 \leq i \leq k$ the λ value of B is given by

$$\lambda(b_i) = \begin{cases} \prod_{C \in s(B)} \lambda_C(b_i) & \text{if } B \text{ is not instantiated} \\ 1 & \text{if } B \text{ is instantiated for } b_i \\ 0 & \text{if } B \text{ is instantiated, but not for } b_i. \end{cases}$$

4. If B is a variable with k possible values and exactly two parents, A and D, A has m possible values, and D has n possible values, then for $1 \leq i \leq k$ the π value of B is given by

$$\pi(b_i) = \sum_{j=1}^{m} \sum_{p=1}^{n} P(b_i \mid a_j, d_p) \pi_B(a_j) \pi_B(d_p).$$

5. If B is a variable with k possible values, then for $1 \leq i \leq k$, $P'(B_i)$, the conditional probability of b_i based on the variables thus far instantiated, is given by

$$P'(b_i) = \alpha\lambda(b_i)\pi(b_i).$$

Initialization

A. Set all λ values, λ messages and π messages to 1.

B. For all roots A, if A has m possible values, then for $1 \leq j \leq m$, set $\pi(a_j) = P(a_j)$.

C. For all roots A for all children B of A do
Post a new π message to B using operative formula 2. (A propagation flow will then begin due to updating procedure C.)

Updating

When a variable is instantiated, or a λ or π message is received by a variable, one of the following updating procedures is used:

A. If a variable B is instantiated for b_j, then

1. Set $P'(b_j) = 1$ and for $i \neq j$, set $P'(b_i) = 0$;
2. Compute $\lambda(B)$ using operative formula 3;
3. Post new λ messages to all B's parents using operative formula 1;
4. Post new π messages to all B's children using operative formula 2.

B. If a variable B receives a new λ message from one of its children, then if B is not already instantiated,

1. Compute the new value of $\lambda(B)$ using operative formula 3;
2. Compute the new value of $P'(B)$ using operative formula 5;
3. Post λ messages to all B's parents using operative formula 1;
4. Post new π messages to B's other children using operative formula 2.

C. If a variable B receives a new π message from a parent, then if B is not already instantiated,

   1. Compute the new value of $\pi(B)$ using operative formula 4;
   2. Compute the new value of $P'(B)$ using operative formula 5;
   3. Post new π messages to all B's children using operative formula 2;

   else if $\lambda(B) \neq (1, 1, \ldots, 1)$, then

   4. Post new λ messages to B's other parents using operative formula 1.

The application of these propagation rules is exemplified in [Neap90]. Even for small Bayesian networks, the propagation may become considerably complex which is induced by the recursion formulas. For bigger networks a supporting tool is vital. Due to that, the propagation process of Bayesian networks is not shown in detail in this thesis but the result for Example 2.9 after the initialization is given as follows:

Example 2.10 (Lack of Oil: Message Propagation Results)
Applying these propagation rules to Example 2.9 yields the same results after initialization as calculated by probabilistic Horn abduction. The probability of a delayed acceleration is: $P(ad) = (0.364, 0.636)$. □

The message propagation described above works for singly connected Bayesian networks. If a network has at least one loop, applying the algorithms can lead to incorrect values, for example because a fact may be used several times which means that the respective probability may be considered multiply in the operating formulas' products. Pearl dedicates a whole chapter to this problem and calls it, "Coping with Loops" [Pearl88]. The method used in this thesis will be introduced in the following section.

2.3.3. Eliminating Loops: Conditioning

In Bayesian networks which have one or more loops and thus are multiply connected, the message propagation according to the given algorithms generally leads to false values because values of a variable are multiply considered in the propagation. In [Pearl88], different ways to eliminate loops in Bayesian networks are described, namely

clustering, conditioning, and stochastic simulation. In this thesis, the conditioning method is used.

Generally speaking, since probabilities and likelihoods are always multiplied when being propagated (see propagation rules in section 2.3.2) and the product $a \cdot a$ of a variable a basically is not idempotent (except for two special values: $a = 0$ and $a = 1$), the double effect of a probabilistic variable often results in too small values. To avoid this double effect for critical values ($a \in]0, 1[$), the variable a can be instantiated with 1, thus taking advantage of the fact that the product $a \cdot a$ is idempotent for $a = 1$. After propagation the results have to be adjusted by the original value of a.

Transferred to Bayesian networks, the basic idea of conditioning is to cut the network for example at the root node of a loop and to duplicate this node, such that every outgoing edge of the original root node now is connected to its own duplicated instance, respectively. For message propagation, this node has to be instantiated for all of its elementary events. The messages are afterwards propagated separately for each instantiated event and the propagation results are at first weighted by the original probabilities of the corresponding elementary events and then summed up.

This will be demonstrated by an example which was introduced in [Cooper84] and referred to for example in [Spie85, PenReg86, Pearl88, Neap90]. The illustration in this thesis is according to [Pearl88].

Example 2.11 (Metastatic Cancer, cf. [Cooper84, Pearl88])
The Bayesian network of Figure 2.4 indicates that "metastatic cancer" (A) has two consequences: "increased total serum calcium" (B) and "brain tumor" (C). The probabilities of A and $\neg A$ are 0.2 and 0.8, respectively. The relevant probabilities of $B, \neg B, C, \neg C$ are the conditional probabilities given A. Similarly, B and C have the consequence "coma" (D). Moreover, C has the consequence "severe headaches" (E). Again, the relevant probabilities are the conditional probabilities given the predecessors (parents). They are defined in the corresponding conditional probability tables in Figure 2.4. □

By definition, Bayesian networks have no directed cycles but – as mentioned above – they can have loops with (at least) two undesirable effects. The Bayesian network in Figure 2.4 has the loop $ABDCA$. First of all, there is a double influence of A on

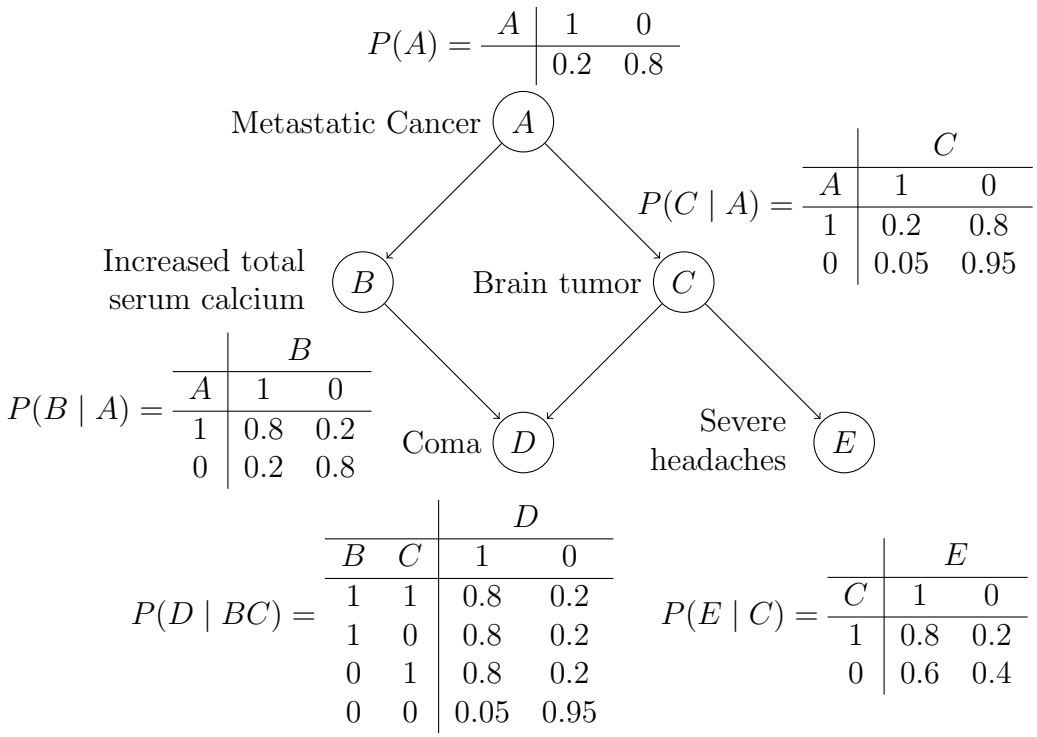
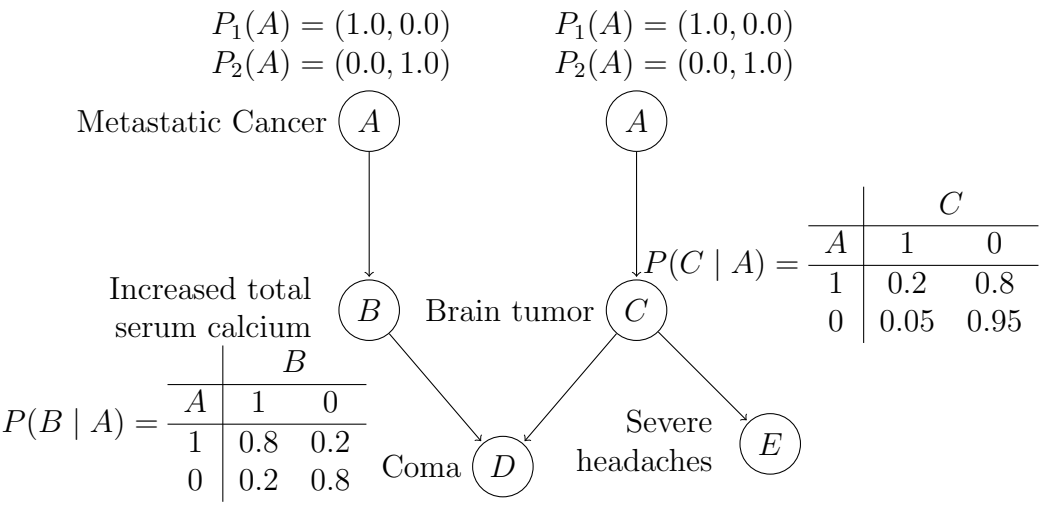


Figure 2.4.: Metastatic Cancer Bayesian Network $\mathcal{B}$

D. On the one hand, from a logical point of view this does not matter because of $A \wedge A = A$. However, from a probabilistic point of view this double influence leads to wrong values. Since the probability $P(A)$ of A is a factor in $P(B)$ and in $P(C)$, $(P(A))^2$ instead of $P(A)$ becomes a factor in $P(D)$. With the exception of $P(A) = 1$ and $P(A) = 0$, $(P(A))^2 \neq P(A)$ causes wrong values. On the other hand, this is the reason for the specific part $P(A) = 1$ and $P(A) = 0$ play in the "conditioning approach" [Pearl88] to cope with loops.

Example 2.12 (Metastatic Cancer: Conditioned Bayesian Network)
The Bayesian network $\mathcal{B}_{cut}$ of Figure 2.5 is a modification of the Bayesian network $\mathcal{B}$ of Figure 2.4 where the loop has been cut off at the root node A. Moreover, the actual probabilities of A and $\neg A$ $(0.2, 0.8)$ have been replaced by two pairs $P_1(A) = (1.0, 0.0)$ and $P_2(A) = (0.0, 1.0)$. Strictly speaking, Figure 2.5 shows two Bayesian networks, where the above mentioned values 0.0 and 1.0 are the initial probabilities for A and $\neg A$. After evaluating both Bayesian networks, the probabilities of all nodes



$P(B \mid A) =$

A	$B=1$	$B=0$
1	0.8	0.2
0	0.2	0.8

$P(C \mid A) =$

A	$C=1$	$C=0$
1	0.2	0.8
0	0.05	0.95

$P(D \mid BC) =$

B	C	$D=1$	$D=0$
1	1	0.8	0.2
1	0	0.8	0.2
0	1	0.8	0.2
0	0	0.05	0.95

$P(E \mid C) =$

C	$E=1$	$E=0$
1	0.8	0.2
0	0.6	0.4

Figure 2.5.: Conditioned Metastatic Cancer Bayesian Network $\mathcal{B}_{cut}$

are determined by calculating the weighted sum of the respective probabilities in both networks, where the weights are the actual probabilities of A and $\neg A$; initially $P(A) = 0.2, P(\neg A) = 0.8$.

Applied to the conditioned Bayesian network shown in Figure 2.5, the following π-values are calculated when initializing the Bayesian network:

$$\pi(A) = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix}$$

$$\pi(B) = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}$$

$$\pi(C) = \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix}$$

$$\pi(D) = \begin{pmatrix} 0.68 & 0.32 \\ 0.23 & 0.77 \end{pmatrix}$$
$$\pi(E) = \begin{pmatrix} 0.64 & 0.36 \\ 0.61 & 0.39 \end{pmatrix}$$

Since no evidence or observations are given, all likelihood values remain neutral:

$$\lambda(A) = \lambda(B) = \lambda(C) = \lambda(D) = \lambda(E) = \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix}$$

Thus, the beliefs for B, C, D and E with A being instantiated for 1 are:

$$BEL_1(B) = (0.8, 0.2) \circ (1.0, 1.0) = (0.8, 0.2)$$
$$BEL_1(C) = (0.2, 0.8) \circ (1.0, 1.0) = (0.2, 0.8)$$
$$BEL_1(D) = (0.68, 0.32) \circ (1.0, 1.0) = (0.68, 0.32)$$
$$BEL_1(E) = (0.64, 0.36) \circ (1.0, 1.0) = (0.64, 0.36)$$

and with A being instantiated for 0:

$$BEL_0(B) = (0.2, 0.8) \circ (1.0, 1.0) = (0.2, 0.8)$$
$$BEL_0(C) = (0.05, 0.95) \circ (1.0, 1.0) = (0.05, 0.95)$$
$$BEL_0(D) = (0.23, 0.77) \circ (1.0, 1.0) = (0.23, 0.77)$$
$$BEL_0(E) = (0.61, 0.39) \circ (1.0, 1.0) = (0.61, 0.39).$$

The actual probability $P(A) = (0.2, 0.8)$ provides the weights for

$$P(B) = BEL_1(B) \cdot 0.2 + BEL_0(B) \cdot 0.8 = (0.32, 0.68)$$
$$P(C) = BEL_1(C) \cdot 0.2 + BEL_0(C) \cdot 0.8 = (0.08, 0.92)$$
$$P(D) = BEL_1(D) \cdot 0.2 + BEL_0(D) \cdot 0.8 = (0.32, 0.68)$$
$$P(E) = BEL_1(E) \cdot 0.2 + BEL_0(E) \cdot 0.8 = (0.616, 0.384).$$

That means, given the probability for metastatic cancer $P(A) = (0.2, 0.8)$, the probabilities for increased total serum calcium and brain tumor are $P(B) = (0.32, 0.68)$ and $P(C) = (0.08, 0.92)$, respectively, while the probabilities for coma and severe

headaches are $P(D) = (0.32, 0.68)$ and $P(E) = (0.616, 0.384)$, respectively. □

Even though Bayesian networks are more expressive than the so-called "fault trees", the latter are usually used for the failure analysis of technical systems. Fault trees model the dependencies of system components regarding their functionality. The next section gives a short summary of relevant concepts and describes how fault trees can be transformed into Bayesian networks to take advantage of the higher expressiveness of Bayesian networks.

2.4. Fault Trees

Fault trees are commonly used in engineering science for dependability analysis. This section briefly and informally introduces fault trees and describes how fault trees can be represented with Bayesian networks. Formal definitions are given in [Vesely*81, Ireson88, DugTri89, RolMor90, Comm06], for example.

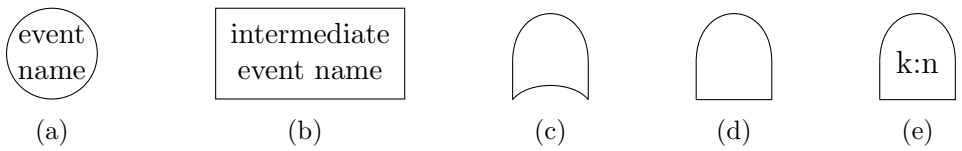


Figure 2.6.: Basic Symbols of a Fault Tree

In principle, a (static) fault tree is a graphical representation of a boolean logical formula consisting of basic events (Figure 2.6(a)), intermediate events (Figure 2.6(b)), OR-gates (Figure 2.6(c)), AND-gates(Figure 2.6(d)), and "k out of n"-gates (Figure 2.6(e)), which build a kind of a tree.

According to [PorBob99], fault trees have the following properties: The so-called "top event", which is the root node, represents a failure of the complete modeled system or of a subsystem. Usually, the top event is represented as an intermediate event. Intermediate events always have exactly one child which can be a gate or a basic event. Gates have at least two children which must be intermediate or basic events. A "k out of n"-gate always has exactly n children. The logical formula is expressed by the nesting of the gates and events. The leaves of the tree always are basic events, which represent certain components of the (sub)system. Basic events

are binary events, either valued `true`, meaning the component is faulty, or `false`, meaning the component works properly.

AND-gates return `true` if every input event is `true`. Else they return `false`. So, for an AND-gate to be faulty all input components/gates have to be faulty, too. OR-gates return `true` if at least one input event is `true`, that means only one faulty input component/gate suffices to cause an OR-gate to be faulty. Only if all input components/gates are functional the OR-gate is functional, too. The result of a "k out of n"-gate is `true` if at least k input events are `true`. Otherwise it returns `false`.

Remark 2.13
As mentioned above, fault trees consisting of the symbols shown in Figure 2.6 are called *static fault trees*. There are also so-called "dynamic fault trees" which are more expressive than static ones, as additional gates and dependencies are defined. But in this thesis, only static fault trees are of interest. ◇

Example 2.13 (Multiprocessor System, cf. [MalTri95])
A very popular example is the fault tree model of a fault tolerant multiprocessor system given in [MalTri95]. Figure 2.7 shows the corresponding fault tree.

The modeled multiprocessor system consists of two redundant subsystems, each consisting of a processor $P_1(P_2)$, a local memory $M_1(M_2)$ and a (mirrored) disk array D_{11}, D_{12} (D_{21}, D_{22}). The subsystems are linked by a bus N and connected to a global memory M_3. The whole system is operational if at least one subsystem and the bus works. A subsystem is operational if the processor works and at least one disk is functional and the local or the global memory works. In the fault tree, this is expressed as follows:

G_1**:** If the bus N **or** G_2 fails, then the whole system fails.

G_2**:** If the first subsystem (G_3) **and** the second subsystem (G_4) fail, then G_2 is faulty, too.

G_3**:** If the second local disk array (G_5) **or** the memory for the second subsystem (G_6) **or** the second processor fails, then G_3 is faulty, too.

G_4**:** If the first local disk array (G_7) **or** the memory for the first subsystem (G_8) **or** the first processor fails, then G_4 is faulty, too.

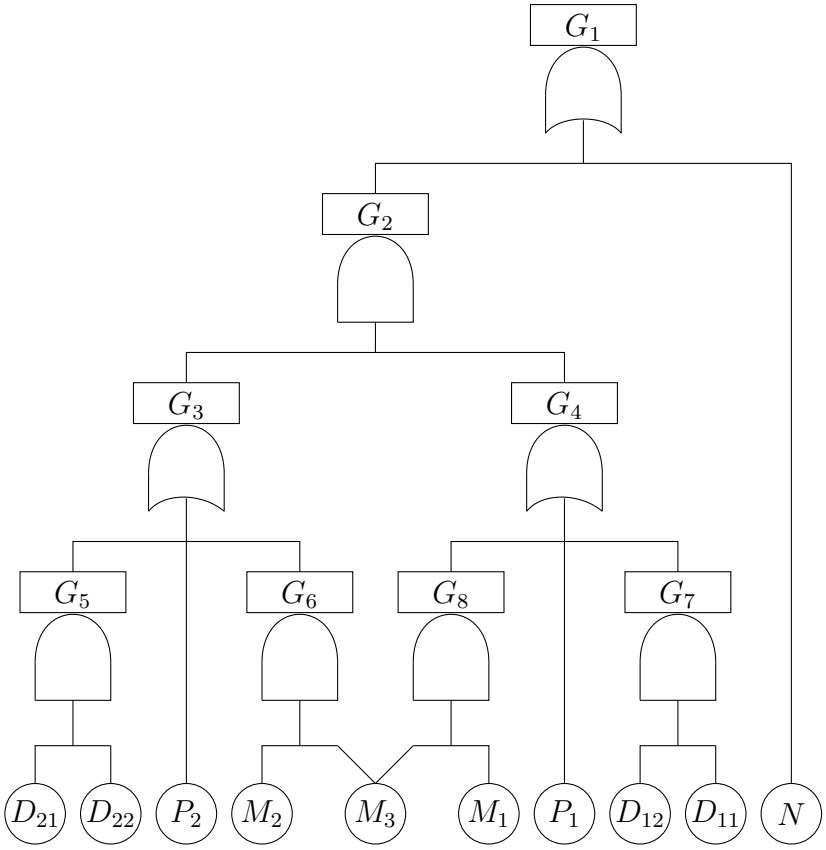


Figure 2.7.: Multiprocessor System Fault Tree

G_5**:** If disk D_{21} **and** disk D_{22} fail, then the second disk array (G_5) is faulty, too.

G_6**:** If the local memory M_2 **and** the global memory M_3 fail, then the memory for the second subsystem is faulty, too.

G_7**:** If disk D_{11} **and** disk D_{12} fail, then the first disk array (G_7) is faulty, too.

G_8**:** If the local memory M_1 **and** the global memory M_3 fail, then the memory for the first subsystem is faulty, too.

The corresponding boolean logical formula can be derived straight forward:

$$G_1 = \Big(\big((D_{21} \wedge D_{22}) \vee P_2 \vee (M_2 \wedge M_3)\big) \wedge \big((M_1 \wedge M_3) \vee P_1 \vee (D_{11} \wedge D_{12})\big)\Big) \vee N.$$

Note that M_3 appears twice which will be of interest later on. □

Until now, the given fault tree represents causal dependencies of different events with respect to the functionality of the corresponding components. By adding probabilistic data and making some basic assumptions on the independence of basic events, the fault tree model can be used to calculate the probability of the top event, that means the probability that the whole system fails. This is described in the next section.

2.4.1. Calculating Probabilities

By assigning probabilities of a component's failure to the corresponding basic events, a fault tree representation can be used to calculate the failure probability of the top event, that means the probability of a failure of the complete (sub)system. Thereby, basic events are always assumed to be stochastically independent. Basically, the probabilities for intermediate events are calculated as follows:

$$P(A \wedge B) = P(A) \cdot P(B) \text{ (AND-gate)}$$
$$P(A \vee B) = P(A) + P(B) - P(A) \cdot P(B) \text{ (OR-gate)}$$

Unfortunately, considering OR-gates with more than two inputs, this formula gets more complex. In addition, similar to Bayesian networks, problems with loops can occur, because the failure probability of one basic event (in Example 2.13 M_3) affects the top event multiply (in Example 2.13 via $G_6 \rightarrow G_3 \rightarrow G_2$ and $G_8 \rightarrow G_4 \rightarrow G_2$). Thus, the boolean formula as the logical representation of the fault tree has to be made stochastically independent. This can be done by so-called "cutset generation" or by *binary decision diagrams (BDDs)* (see [DoyDug95, Dugan01]). The latter is now demonstrated by continuing Example 2.13:

Example 2.14 (Multiprocessor System: Binary Decision Diagram)
A binary decision diagram is a tree which represents a logical formula. Its intermediate nodes are binary events – in case of a fault tree representation basic events –, its leaves are usually two nodes 0 (representing a functional system) and 1 (representing a faulty system). As the events are binary, every event E has exactly two outcomes: `false` (functional, denoted $\neg E$) and `true` (faulty, denoted E). The outgoing edges of a node are related to its outcomes. Usually, the left outgoing edge means `false` and

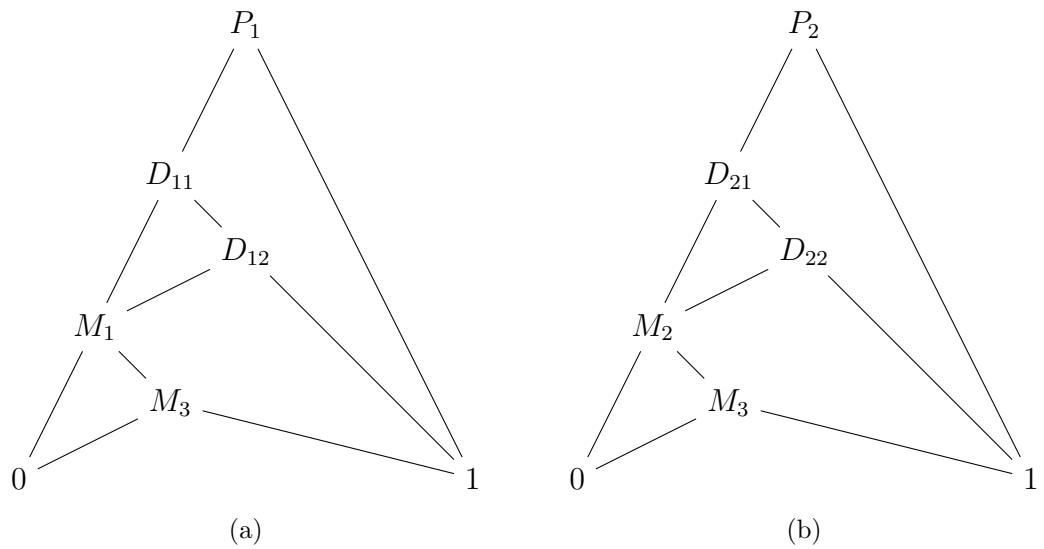


Figure 2.8.: Binary Decision Diagrams for the two Subsystems

the right outgoing edge means `true`.

The binary decision diagram of a (sub)system is generated bottom up by recursively connecting partial diagrams according to the function of the gate connecting these parts. This is demonstrated for example in [Dugan01].

Figure 2.8 shows the binary decision diagrams corresponding to the two subsystems. To generate stochastically independent components of the boolean formula (which is needed to calculate probabilities), every possible path starting at the root node and leading to the fault event 1 is traced. In Figure 2.8(a), for example, there are four distinct paths (see Figure 2.9):

- $P_1 \rightarrow 1$ (see Figure 2.9(a)),
- $\neg P_1 \rightarrow D_{11} \rightarrow D_{12} \rightarrow 1$ (see Figure 2.9(b)),
- $\neg P_1 \rightarrow D_{11} \rightarrow \neg D_{12} \rightarrow M_1 \rightarrow M_3 \rightarrow 1$ (see Figure 2.9(c)),
- $\neg P_1 \rightarrow \neg D_{11} \rightarrow M_1 \rightarrow M_3 \rightarrow 1$ (see Figure 2.9(d)).

The probability $P(G_4)$ that the first subsystem fails can then be determined as

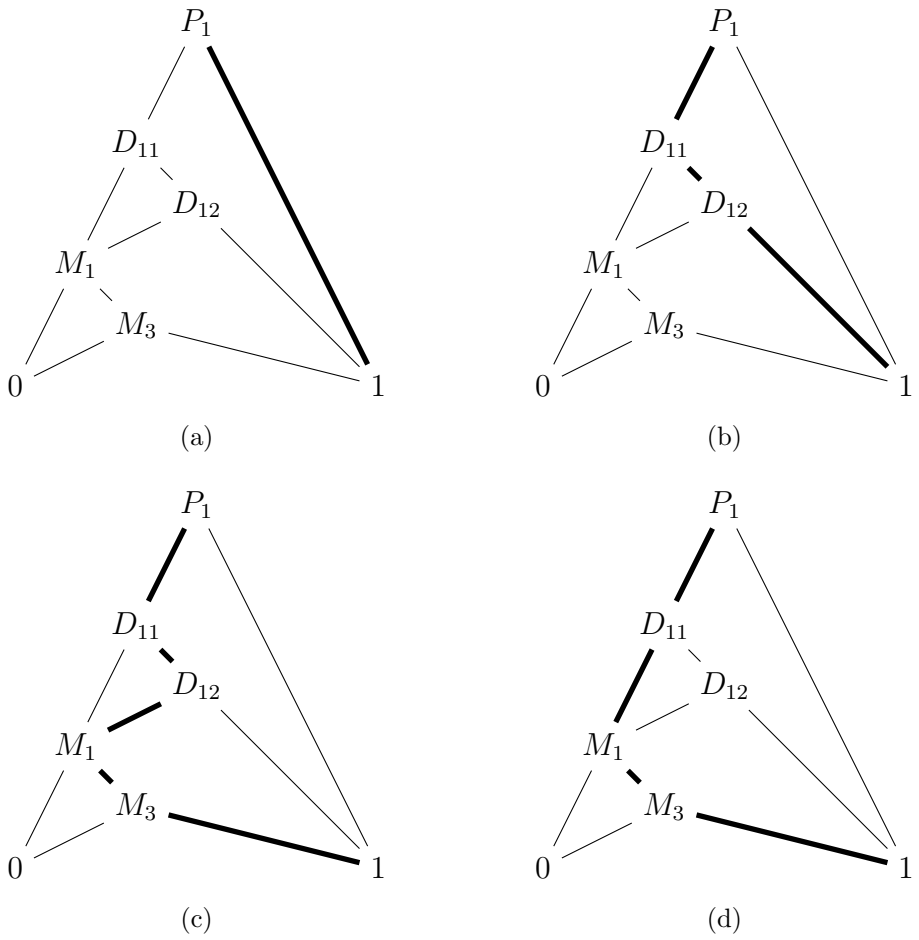


Figure 2.9.: Paths to the Fault Node of Subsystem 1

follows:

$$\begin{aligned} P(G_4) = {} & P(P_1) \\ & + P(\neg P_1) \cdot P(D_{11}) \cdot P(D_{12}) \\ & + P(\neg P_1) \cdot P(D_{11}) \cdot P(\neg D_{12}) \cdot P(M_1) \cdot P(M_3) \\ & + P(\neg P_1) \cdot P(\neg D_{11}) \cdot P(M_1) \cdot P(M_3). \end{aligned}$$

Figure 2.10 shows the complete binary decision diagram for the multiprocessor system, which was built by applying the rules described in [Dugan01]. The loop contained in the fault tree causes M_3 to appear twice in the binary decision diagram. Hence, it is possible to find contradictory paths through the diagram. The sequence

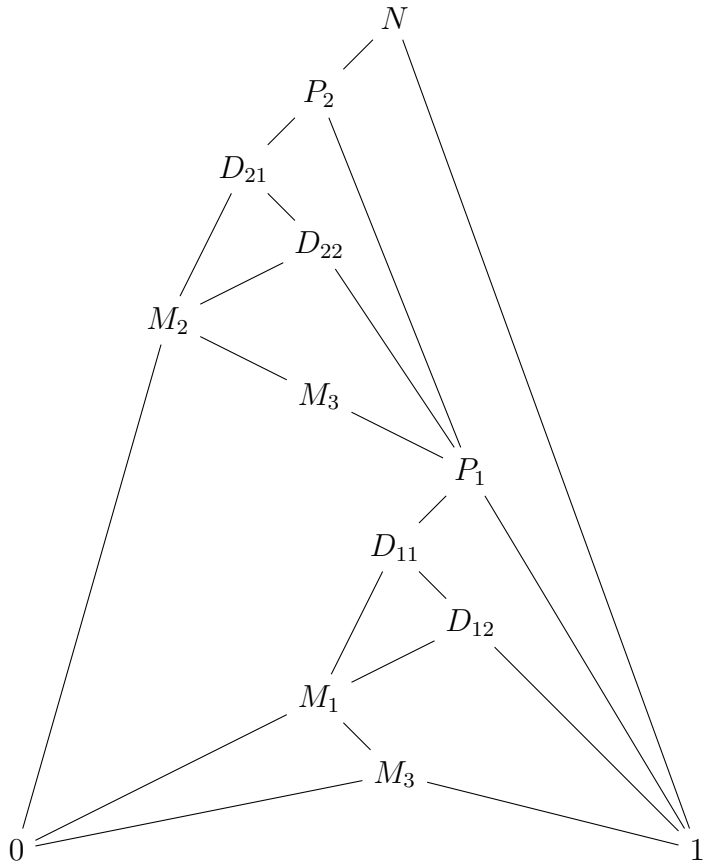


Figure 2.10.: Binary Decision Diagram for the complete Multiprocessor System

$\neg N \rightarrow \neg P_2 \rightarrow \neg D_{21} \rightarrow M_2 \rightarrow M_3 \rightarrow \neg P_1 \rightarrow \neg D_{11} \rightarrow M_1 \rightarrow \neg M_3$, for example, can be located as a path in the diagram but is not valid because it contains M_3 as well as $\neg M_3$. On the one hand, this could be regarded as a disadvantage of the binary decision diagram representation. On the other hand, the primary purpose of the diagram is to find paths leading to the fault node 1 and therefore it is essential that those paths are consistent — which they are in this case. There is no contradictory path from the root node to the fault node.

Obviously, the more nodes are contained in the diagram, the more distinct paths from the root node to the fault node 1 can be found. In this case, there are 17 distinct paths, which lead to the following formula to calculate the probability for a

non-operational system $P(G_1)$:

$$
\begin{aligned}
P(G_1) &= P(N) \\
&+ P(\neg N) \cdot P(P_2) \cdot P(P_1) \\
&+ P(\neg N) \cdot P(P_2) \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(D_{12}) \\
&+ P(\neg N) \cdot P(P_2) \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(\neg D_{12}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(P_2) \cdot P(\neg P_1) \cdot P(\neg D_{11}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(D_{22}) \cdot P(P_1) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(D_{22}) \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(D_{12}) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(D_{22}) \\
&\quad \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(\neg D_{12}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(D_{22}) \\
&\quad \cdot P(\neg P_1) \cdot P(\neg D_{11}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(\neg D_{22}) \cdot P(M_2) \cdot P(M_3) \cdot P(P_1) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(\neg D_{22}) \cdot P(M_2) \cdot P(M_3) \\
&\quad \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(D_{12}) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(\neg D_{22}) \cdot P(M_2) \cdot P(M_3) \\
&\quad \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(\neg D_{12}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(D_{21}) \cdot P(\neg D_{22}) \cdot P(M_2) \cdot P(M_3) \\
&\quad \cdot P(\neg P_1) \cdot P(\neg D_{11}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(\neg D_{21}) \cdot P(M_2) \cdot P(M_3) \cdot P(P_1) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(\neg D_{21}) \cdot P(M_2) \cdot P(M_3) \\
&\quad \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(D_{12}) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(\neg D_{21}) \cdot P(M_2) \cdot P(M_3) \\
&\quad \cdot P(\neg P_1) \cdot P(D_{11}) \cdot P(\neg D_{12}) \cdot P(M_1) \cdot P(M_3) \\
&+ P(\neg N) \cdot P(\neg P_2) \cdot P(\neg D_{21}) \cdot P(M_2) \cdot P(M_3) \\
&\quad \cdot P(\neg P_1) \cdot P(\neg D_{11}) \cdot P(M_1) \cdot P(M_3).
\end{aligned}
$$

Considering that the modeled multiprocessor system is not that complex, it is obvious that real-life system models require adequate tool support.

Variable	Failure Probability
N	0.00001
P_1, P_2	0.0025
M_1, M_2, M_3	0.00015
$D_{11}, D_{12}, D_{21}, D_{22}$	0.32968

Table 2.6.: Failure Probabilities of the Multiprocessor System Components

Assume the prior probabilities of the basic events as given in Table 2.14. The values are chosen according to [Bobbio*99]. Assigning these values to the variables in the above formula leads to the probability

$$\underline{P(G_1)} = 0.0123125023 \approx \underline{0.01231}$$

for a faulty system. □

The next section describes how to translate fault trees into Bayesian networks. By that the higher expressiveness of Bayesian networks can be used to evaluate more complex scenarios than the fault tree approach sketched above could. Additionally, for example the conditioning method can be used to eliminate loops in the Bayesian network representation of the fault tree which makes binary decision diagrams needless.

2.4.2. Translating Fault Trees into Bayesian Networks

Fault trees can be translated into Bayesian networks straightforward. The mapping rules are described according to [PorBob99, Bobbio*99]:

- Each basic event is represented by a root node in the Bayesian network. The corresponding conditional probability table contains the prior probabilities of the basic event.
- Each gate is represented by an (intermediate) node in the Bayesian network.
- The nodes in the Bayesian network are connected like the corresponding basic events and gates are connected in the fault tree.

- The conditional probability tables of the Bayesian network nodes corresponding to gates are set as follows:
  - If the node corresponds to an AND-gate, its conditional probability table returns `true` with probability 1.0, if and only if all input variables are `true`, else it returns `false` with probability 1.0 (and `true` with probability 0.0).
  - If the node corresponds to an OR-gate, its conditional probability table returns `false` if and only if all input variables are `false`, else it returns `true`.
  - If the node corresponds to a "k out of n"-gate, its conditional probability table returns `true`, if at least k input variables are `true`, else it returns `false`.
- The node corresponding to the top event of the fault tree is labeled as the $Fault$ node.

The translation is now demonstrated by continuing the Multiprocessor System example.

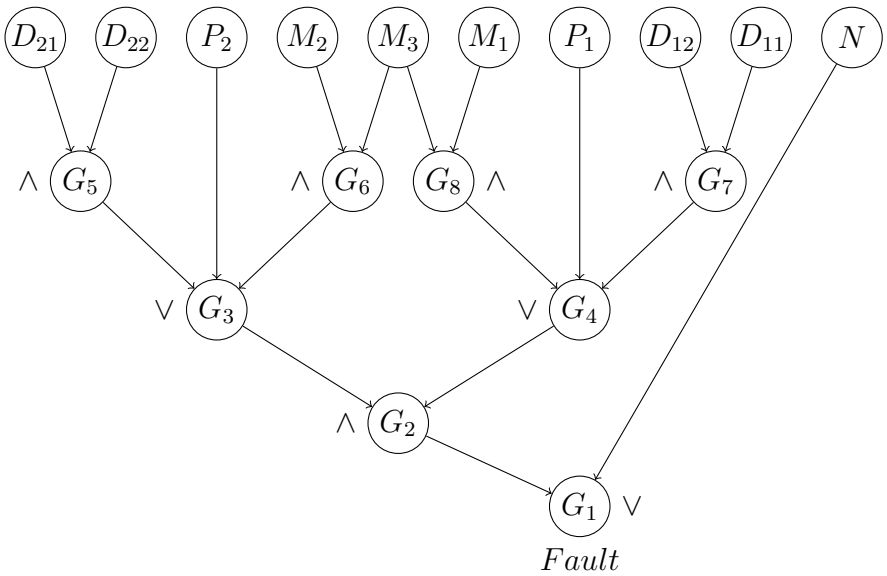


Figure 2.11.: Bayesian Network for the Multiprocessor System

Example 2.15 (Multiprocessor System: Bayesian Network)
Figure 2.11 (cf. [Bobbio*99]) shows the Bayesian network representing the fault tree of Example 2.13. Due to lack of space, the conditional probability tables are not annotated in the figure but they are listed in Table 2.7. Obviously, the entries in the conditional probability tables are trivial (either 1.0 or 0.0). As additionally every variable is binary, this kind of Bayesian network is called a *binary Bayesian network*.

By initializing the Bayesian network, the system fault probability can be calculated. This is best done with the help of a Bayesian network tool. For this thesis, the free available tool "BNJ (Bayesian Network tools in Java)" [Hsu04] was used.[2] It computes $P(G_1) = 0.012$, which – considering the accuracy of the displayed results – is equal to the result determined by the binary decision diagram method. □

Remark 2.14
Since the Bayesian network shown in Figure 2.11 is binary, the corresponding interpretation of the conditional probability tables is annotated at the gate nodes. These annotations are not part of the Bayesian network syntax; it is just to ease the understanding of the Bayesian network and to show the analogy to the fault tree. ◇

[2]The XML-code of the network for this example is listed in appendix A. Since it is coded in the so-called "Bayesian Networks Interchange Format (BIF)", it should be compatible to all tools supporting this standard format.

D_{ij}	1	0
	0.32968	0.67032

P_i	1	0
	0.0025	0.9975

M_k	1	0
	0.00015	0.99985

N	1	0
	0.00001	0.99999

$i, j \in \{1, 2\}$, $k \in \{1, 2, 3\}$

		G_1	
G_2	N	1	0
1	1	1.0	0.0
1	0	1.0	0.0
0	1	1.0	0.0
0	0	0.0	1.0

		G_2	
G_3	G_4	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

			G_3	
P_2	G_5	G_6	1	0
1	1	1	1.0	0.0
1	1	0	1.0	0.0
1	0	1	1.0	0.0
1	0	0	1.0	0.0
0	1	1	1.0	0.0
0	1	0	1.0	0.0
0	0	1	1.0	0.0
0	0	0	0.0	1.0

			G_4	
P_1	G_7	G_8	1	0
1	1	1	1.0	0.0
1	1	0	1.0	0.0
1	0	1	1.0	0.0
1	0	0	1.0	0.0
0	1	1	1.0	0.0
0	1	0	1.0	0.0
0	0	1	1.0	0.0
0	0	0	0.0	1.0

		G_5	
D_{21}	D_{22}	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

		G_6	
M_2	M_3	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

		G_7	
D_{11}	D_{12}	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

		G_8	
M_1	M_3	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

Table 2.7.: Conditional Probability Tables of the Bayesian Network

3. Low-Level Probability Propagation Nets

This chapter introduces the first class of probability propagation nets which are suited to represent probabilistic Horn abduction. Their structure is defined in section 3.1. Section 3.2 explains how the probabilities for goal variables can be calculated. A first folding of these low-level probability propagation nets is shown in section 3.3.

3.1. Petri Net Representation of Probabilistic Horn Abduction

A considerable limitation of the canonical Petri net representation of Horn formulas with respect to probabilistic Horn abduction is that the probabilities are not directly integrated in the Petri net representation. If they were contained in the Petri net appropriately, it would be possible to calculate probabilities of explanations by simulating the corresponding t-invariants. In the lack of oil example, this can be achieved as follows:

Example 3.1 (Lack of Oil: Motivation of Net Extensions)
Consider explanation ε_5 for acceleration delayed ($acde$). The Petri net representation of the corresponding invariant I_5 is shown in Figure 3.1(a). For the calculation of $P(\varepsilon_5) = P(lo \wedge igno)$ it would be convenient to have the following sequence of markings:

1. M with $M(lo) = M(igno) = M(acde) = \varnothing$ (empty marking)

2. $M'(lo) = (P(lo)) = (P_\alpha(t_{16})) = (0.4)$;
   $M'(igno) = (P(igno)) = (P_\alpha(t_{13})) = (0.9)$;

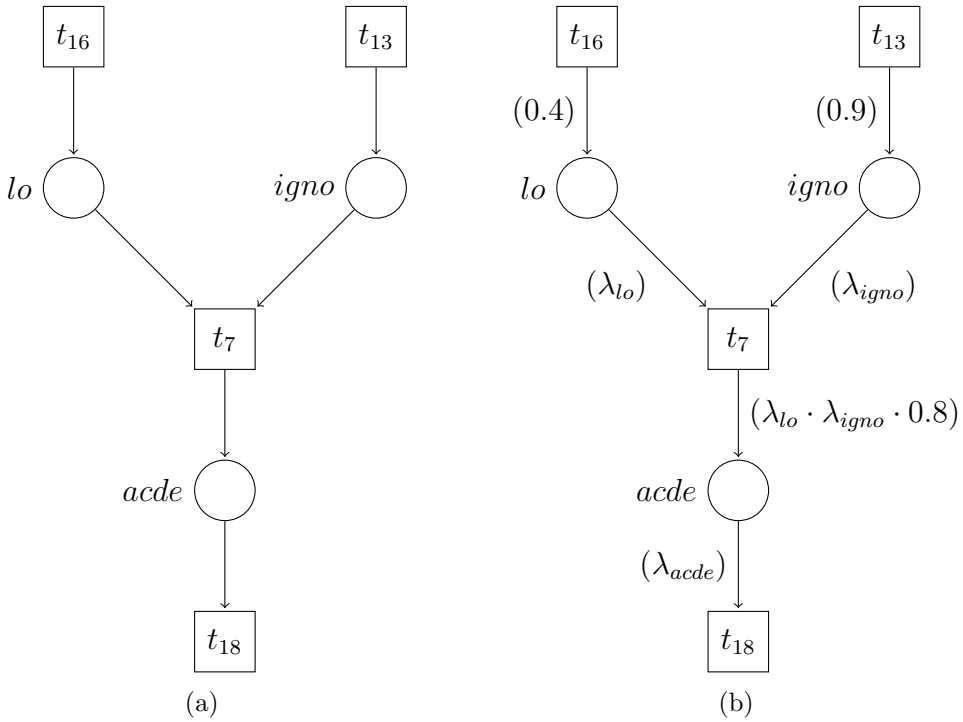


Figure 3.1.: Invariant I_5 of Example 2.8 as Part of the Canonical Net Representation and the corresponding Probability Propagation Net

$M'(acde) = \varnothing$ after one subsequent firing of t_{16} and t_{13}

3. $M''(lo) = M''(igno) = \varnothing$
   $M''(acde) = (P(\varepsilon_5)) = (P(lo) \cdot P(igno) \cdot P(acde \mid lo \wedge igno)) = (0.4 \cdot 0.9 \cdot P_\alpha(t_7)) =$
   $(0.4 \cdot 0.9 \cdot 0.8) = (0.288)$ after one subsequent firing of t_7

4. $M''' = M$ (empty marking) after one subsequent firing of t_{18}.

To get all that in accordance with the notation of a suitable higher level Petri net (predicate/transition net notation in this case, cf. [GenLau79, GenLau83]) the net must be completed as shown in Figure 3.1(b). □

The following definition (cf. [LauPin05, LaPhPi06]) describes this transformation and thus the representation of probabilistic Horn formulas with a new class of Petri nets.

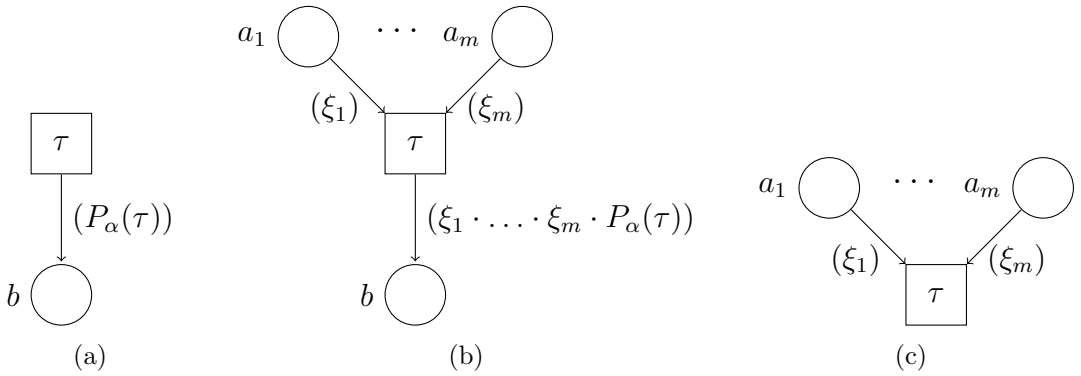


Figure 3.2.: Arc Label Function Types for a Probability Propagation Net

Definition 3.1 (Low-Level Probability Propagation Net)
Let α *be a Horn formula and* $\tau = \neg a_1 \vee \cdots \vee \neg a_m \vee b$ *a Horn clause of* α *with* $\neg A = \{\neg a_1, \ldots, \neg a_m\}, B = \{b\}$*;*
$\mathcal{PN}_\alpha = (S_\alpha, T_\alpha, F_\alpha, P_\alpha, L_\alpha)$ *is a* (low-level) probability propagation net (PPN) *for* α *iff*

- $\mathcal{N}_\alpha = (S_\alpha, T_\alpha, F_\alpha)$ *is the canonical net representation of* α*,*
- P_α *is a probability function for* α*,*
- L_α *is an* arc label function *for* α *where for* τ *the following holds:*
  - *if* τ *is a fact clause* ($\tau \in \mathbb{F}(\alpha), \neg A = \varnothing, B \neq \varnothing$)*:*
    $L_\alpha(\tau, b) = (P_\alpha(\tau))$, $(\tau, b) \in F_\alpha$ *(see Figure 3.2(a))*
  - *if* τ *is a rule clause* ($\tau \in \mathbb{R}(\alpha), \neg A \neq \varnothing, B \neq \varnothing$)*:*
    $L_\alpha(a_i, \tau) = (\xi_i)$ *for* $1 \leq i \leq m$
    $L_\alpha(\tau, b) = (\xi_1 \cdots \xi_m \cdot P_\alpha(\tau))$
    where the ξ_i *are variables ranging over* $[0, 1]$ *(see Figure 3.2(b))*
  - *if* τ *is a goal clause* ($\tau \in \mathbb{G}(\alpha), \neg A \neq \varnothing, B = \varnothing$)*:*
    $L_\alpha(a_i, \tau) = (\xi_i)$ *for* $1 \leq i \leq m$ *(see Figure 3.2(c)).*

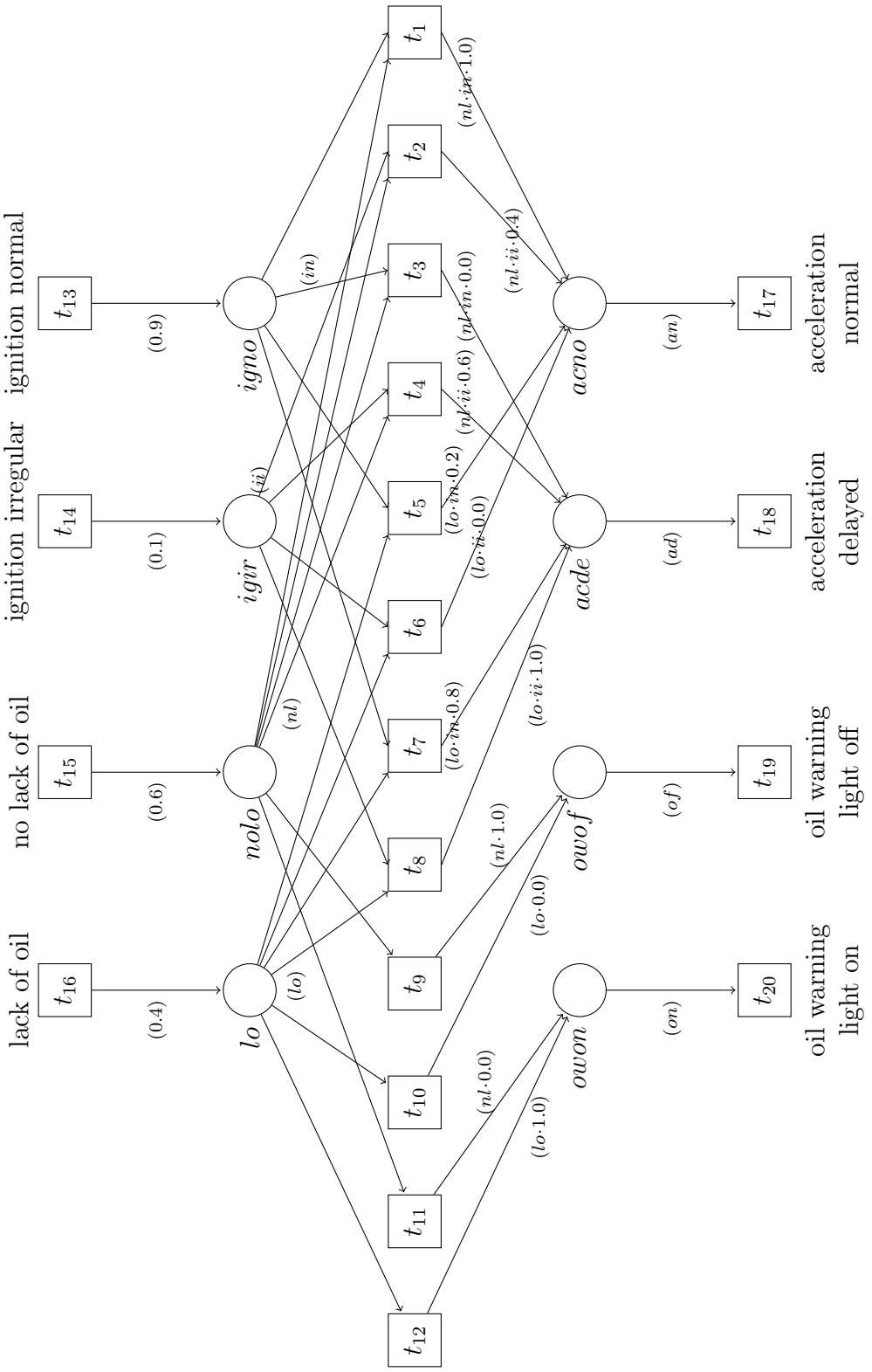


Figure 3.3.: $\mathcal{PN}_\alpha$ of Example 3.2

Example 3.2 (Lack of Oil: Probability Propagation Net)
Figure 3.3 shows the low-level probability propagation net of Example 2.4. It combines the net $\mathcal{N}_\alpha$ of Figure 2.2 and the probabilities of Table 2.4. □

Basically, the canonical net representation of the Horn formula is extended by specific arc labels. To understand the functionality, the marking of probability propagation nets has to be defined.

Definition 3.2 (Probability Propagation Net Marking)
Let α be a Horn formula and $\mathcal{PN}_\alpha = (S_\alpha, T_\alpha, F_\alpha, P_\alpha, L_\alpha)$ a probability propagation net for α; let W be a finite subset of $[0,1]$, and let $(W) := \{(w) \mid w \in W\}$ be the corresponding set of 1-tuples; let be $\tau \in T_\alpha$ with ${}^\bullet\tau = \{s_1, \ldots, s_m\}, \tau^\bullet = \{s_{m+1}\}$ (that means $\tau = \neg s_1 \vee \cdots \vee \neg s_m \vee s_{m+1}$);
then $M : S_\alpha \to \mathbb{M}((W))$ is a marking *of $\mathcal{PN}_\alpha$;*
τ is enabled by *M for $\{(w_1), \ldots, (w_m)\}$ iff $(w_1) \in M(s_1), \ldots, (w_m) \in M(s_m)$,*
the follower marking *M' of M after one firing of τ for $\{(w_1), \ldots, (w_m)\}$ is given by*

$$\begin{aligned} M'(s_1) &= M(s_1) - (w_1), \\ &\vdots \\ M'(s_m) &= M(s_m) - (w_m), \\ M'(s_{m+1}) &= M(s_{m+1}) + (w_1 \cdot w_2 \ldots w_m \cdot P_\alpha(\tau)); \end{aligned}$$

if $(\xi_1), \ldots, (\xi_m)$ are the arc labels of $(s_1, \tau), \ldots, (s_m, \tau) \in F_\alpha$, one may write

$$\begin{aligned} M'(s_1) = M(s_1) - (\xi_1), \ldots, M'(s_m) &= M(s_m) - (\xi_m), \\ M'(s_{m+1}) &= M(s_{m+1}) + (\xi_1 \ldots \xi_m \cdot P_\alpha(\tau)), \end{aligned}$$

if the ξ_i are bound by the corresponding $w_i, 1 \leq i \leq m$.

Example 3.3 (Lack of Oil: Probability Propagation Net Operation)
The prior probabilities of the source variables are annotated as constant arc labels connecting the boundary transitions $t_{13} \ldots t_{16}$ and their postset places *lo*, *nolo*, *igir*, *igno*. The variables inscribed on the outgoing arcs of these places bind tuples located on the places to the variables *lo*, *nl*, *ir*, *in*, thus representing the probability for 'lack

of oil', 'no lack of oil', 'ignition irregular', and 'ignition normal', respectively.

Transitions $t_1 \dots t_{12}$ are rule transitions. Their outgoing arcs weight products of the input variables lo, nl, ii, in with the probabilities of the corresponding rules and put the result on the postset places $owon, owof, acde, acno$.

After binding these results to the variables on, of, ad, an, the boundary transitions $t_{17} \dots t_{20}$ remove them from their respective input place. □

In analogy to the canonical net representation of Horn formulas, t-invariants play an important role to structure probability propagation nets and to calculate probabilities of certain events or explanations. This can easily be demonstrated by simulating the t-invariants as shown in the next section.

3.2. Calculating Probabilities by Simulation

To calculate the probability of a goal variable, all t-invariants containing the corresponding goal transition in their support have to be simulated. Basically, the single results (one-tuples being consumed by the goal transition to complete the **0**-reproduction of the empty marking) represent single probabilities related to the specific constellation of the input parameters. If the t-invariants used for the calculation do not contain any loop, the single probabilities can be summed up to get the probability value of the goal variable. This is demonstrated in Example 3.4. In case there is at least one loop in the set of used t-invariants, the double effect of one or more input variable has to be eliminated as shown in Example 3.5.

Example 3.4 (Lack of Oil: Simulation of T-Invariants)
The probabilities of Example 2.8 will now be calculated by simulating the t-invariants I_5, I_6, I_7, I_8. For example, simulating I_5 yields the maximal probability $P(\varepsilon_5) = 0.288$:

1. Firing t_{13} and t_{16} yields tuples (0.9) and (0.4) on places $igno$ and lo, respectively.
2. Firing t_7 removes these tuples and puts the tuple $(0.4 \cdot 0.9 \cdot 0.8) = (0.288)$ on place $acde$.
3. From there, it is removed by t_{18}, thus completing the reproduction of the empty marking by simulating I_5.

Altogether,

$$I_5 : P(\varepsilon_5) = 0.9 \cdot 0.4 \cdot 0.8 \cdot 1.0 = 0.288$$
$$I_6 : P(\varepsilon_6) = 0.9 \cdot 0.6 \cdot 0 \cdot 1.0 = 0.0$$
$$I_7 : P(\varepsilon_7) = 0.1 \cdot 0.4 \cdot 1.0 \cdot 1.0 = 0.04$$
$$I_8 : P(\varepsilon_8) = 0.1 \cdot 0.6 \cdot 0.6 \cdot 1.0 = 0.036$$

are the simulation results of the four t-invariants, which are in accordance with the values calculated by using the canonical net representation in Example 2.8 and the results of probabilistic Horn abduction in [PorTor97], as well. □

A major problem, the "loopiness", arises from the fact that the conjunction operator $\wedge$ is idempotent ($a \wedge a = a$) but the corresponding product of probabilities is not idempotent in general:

$$P(a) \cdot P(a) \begin{cases} = P(a) & \text{if } P(a) = 1 \text{ or } P(a) = 0 \\ \neq P(a) & \text{else} \end{cases}$$

The following example shows a case of loopiness and a method to get over that difficulty.

Example 3.5 (Lack of Oil: Loopiness)
To calculate the probability of $acde \wedge owon$, the probability propagation net of Figure 3.3 is modified in several steps (cf. [LauPin07]):

- transitions (goal clauses) $t_{18} = \neg acde$ and $t_{20} = \neg owon$ are unified to one transition (goal clause) $t_{20} = \neg acde \vee \neg owon = \neg(acde \wedge owon)$;
- the transitions $t_{19} = \neg owof$ and $t_{17} = \neg acno$ are omitted because they are not needed any more; as a consequence, also $t_9, t_{10}, t_1, t_2, t_5, t_6$ are no longer needed.
- all rule transitions t where $P_\alpha(t) = 0$ are omitted: t_{11}, t_3;
- t_{15} and t_4 are omitted because the only t-invariant they belong to contains a factual contradiction: t_{16} (lack of oil) and t_{15} (no lack of oil).

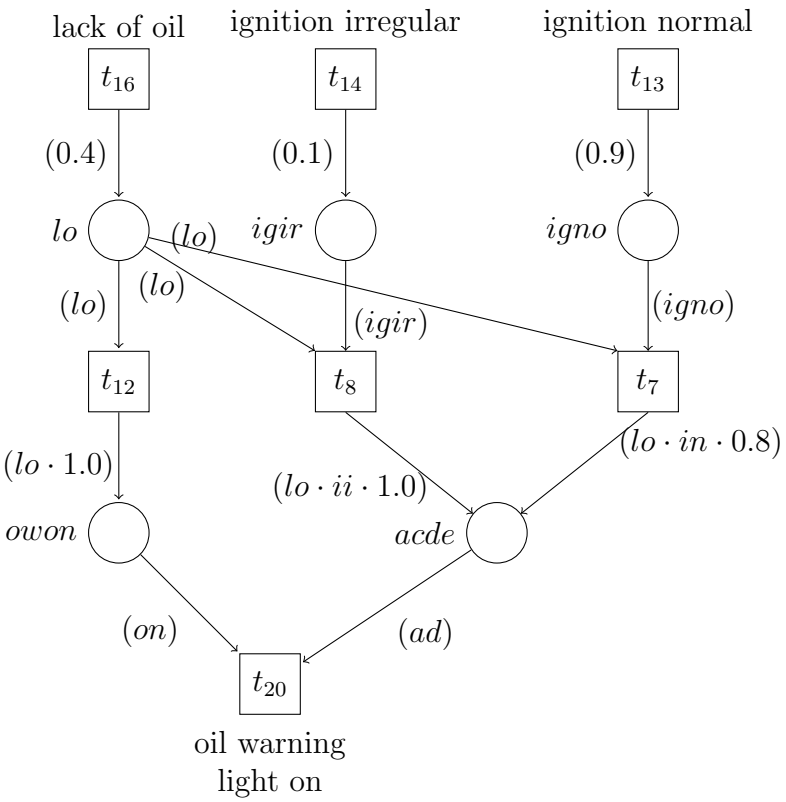


Figure 3.4.: Probability Propagation Net $\mathcal{PN}_\alpha$ of Example 3.5

	t_5	t_7	t_8	t_{12}	t_{13} $igno$	t_{14} $igir$	t_{16} lo	t_{20} $\neg owon$
I_1			1	1		1	2	1
I_2		1		1	1		2	1

Table 3.1.: T-invariants of $\mathcal{PN}_\alpha$ (Example 3.5)

The result is the probability propagation net $\mathcal{PN}_\alpha$ shown in Figure 3.4. From a structural point of view, this net is well suited for solving the given problem because its set of t-invariants (see Table 3.1) is reduced to the relevant ones. From a probabilistic point of view, it stands out that the net is loopy. On the other hand, the net is optimal to apply a variation of Pearl's conditioning method ([Pearl88], see section 2.3.3).

In contrast to his technique to cut the loops, in the Petri net representation there is no need to cut the net because of the t-invariant structure that forces to fire t_{16} twice in both t-invariants (see Table 3.1). This, in principle, leads to a double effect of (lo) when t_{20} fires (via *owon* and via *acde*). For $L_\alpha(t_{16}, lo) = (P(t_{16})) = (1.0)$, however,

this effect is neutralized. So, by simulating or simply multiplying the probabilities, the following temporary values are calculated for the t-invariants:

$$I_1 : P_\alpha(t_{16})^2 \cdot P_\alpha(t_{14}) \cdot P_\alpha(t_{12}) \cdot P_\alpha(t_8) \cdot P_\alpha(t_{20}) = 0.1$$
$$I_2 : P_\alpha(t_{16})^2 \cdot P_\alpha(t_{13}) \cdot P_\alpha(t_{12}) \cdot P_\alpha(t_7) \cdot P_\alpha(t_{20}) = 0.72$$

Finally, both values have to be multiplied by the weight 0.4 which is the original value of $P_\alpha(t_{16})$:

$$P(acde \wedge owon) = 0.04 \qquad \text{w.r.t. } I_1$$
$$P(acde \wedge owon) = 0.288 \qquad \text{w.r.t. } I_2$$

These values are also the probabilities for the two explanations:

$$\varepsilon_1 : \{lo, igir\} = lo \wedge igir, P(\varepsilon_1) = 0.04$$
$$\varepsilon_2 : \{lo, igno\} = lo \wedge igno, P(\varepsilon_2) = 0.288.$$

Finally, $P(acde \wedge owon) = 0.04 + 0.288 = 0.328$. □

3.3. Folding Low-Level Probability Propagation Nets

In low-level probability propagation nets every fact, rule, and goal clause of the underlying Horn formula is represented by an own transition. This structure can be simplified by a folding of corresponding places and transitions and by transferring the complexity of the original graph structure to the labels, markings and the firing rule:

Definition 3.3 (Higher-Level Probability Propagation Net)
Let α be a Horn formula and let $\mathcal{PN}_\alpha = (S_\alpha, T_\alpha, F_\alpha, P_\alpha, L_\alpha)$ be the corresponding low-level probability propagation net. Let $\mathbb{A}(\alpha) = \{a_1, \ldots, a_n, \bar{a}_1, \ldots \bar{a}_n\}$ be the set of atoms of α where a_i is the complementary atom of $\bar{a}_i$ and vice versa for $i = 1, \ldots, n$.

- *Define a partial order on all atoms in $\mathbb{A}(\alpha)$ such that for every set $\{a, \bar{a}\}$ of complementary atoms (see Definition 2.12) either $a < \bar{a}$ or $\bar{a} < a$ holds.*
- *For every pair $(a_i, \bar{a}_i)$ of complementary atoms $a_i, \bar{a}_i \in \mathbb{A}(\alpha)$ let a_i' be a new*

variable, whose values are pairs $(a_i, \bar{a}_i)$ *with* $a_i < \bar{a}_i$ *with respect to the partial order defined above.*

- *Let* $S'_\alpha := \{a'_i \mid 1 \leq i \leq n\}$ *be the set of the new variables created in the previous step.*

- *Let* $f_{\mathbb{A}} : \mathbb{A}(\alpha) \to S'_\alpha$ *be the corresponding mapping, such that* $f_{\mathbb{A}}(a_i) = f_{\mathbb{A}}(\bar{a}_i) = a'_i$.

- *For every* $\tau_i \in \mathbb{C}(\alpha), \quad \tau_i = \neg a_1 \vee \cdots \vee \neg a_m \vee b$ *let* $\tau'_i := \neg f_{\mathbb{A}}(a_1) \vee \cdots \vee \neg f_{\mathbb{A}}(a_m) \vee f_{\mathbb{A}}(b) = \neg a'_1 \vee \cdots \vee \neg a'_m \vee b'$ *be the clause which results from applying the mapping* $f_{\mathbb{A}}$ *to every atom of* τ_i.

- *Let* $T'_\alpha := \{\tau'_i \mid \tau_i \in \mathbb{C}(\alpha)\}$ *be the set of new clauses which results from applying the mapping* $f_{\mathbb{A}}$ *to every atom of every clause* $\tau_i \in \mathbb{C}(\alpha)$.

- *Let* $f_{\mathbb{C}} : \mathbb{C}(\alpha) \to T'_\alpha$ *be the corresponding mapping for the clauses, such that* $f_{\mathbb{C}}(\tau_i) = \tau'_i$. *If* τ_i *is a fact (rule, goal) clause,* τ'_i *is called a fact (rule, goal) clause, too.*

- *Let* $F'_\alpha := \{(a'_i, \tau'_j) \mid (a_i, \tau_j) \in F_\alpha \wedge a'_i = f_{\mathbb{A}}(a_i) \wedge \tau'_j = f_{\mathbb{C}}(\tau_j)\} \cup \{(\tau'_i, a'_j) \mid (\tau_i, a_j) \in F_\alpha \wedge \tau'_i = f_{\mathbb{C}}(\tau_i) \wedge a'_j = f_{\mathbb{A}}(a_j)\}$ *be the set of edges which results from applying* $f_{\mathbb{A}}$ *to the atoms and* $f_{\mathbb{C}}$ *to the clauses of* F_α.

- *Let* L'_α *be a new arc label function for* α *where for every clause* $\tau' \in T'_\alpha$, $\tau' = \neg a'_1 \vee \cdots \vee \neg a'_m \vee b', \quad a'_1, \ldots, a'_m, b' \in S'_\alpha$ *the following holds:*

  - *if* τ' *is a fact clause and* $b' = (b, \bar{b})$, $b < \bar{b}$:
    $L'_\alpha(\tau', b') = \big(P_\alpha(b), P_\alpha(\bar{b})\big)$,

  - *if* τ' *is a rule clause:*
    $L'_\alpha(a'_i, \tau') = (\xi_i) \quad$ *for* $1 \leq i \leq m$
    $L'_\alpha(\tau', b') = (\xi_1 \times \cdots \times \xi_m) \cdot M_{\tau'}$
    where the ξ_i *are pairs ranging over* $([0,1] \times [0,1])$ *and* $M_{\tau'}$ *is a* $(2^m \times 2)$-

matrix built as follows:

$$M_{\tau'} := \begin{pmatrix} P_\alpha\left(\neg a_1 \vee \cdots \vee \neg a_m \vee b\right) & P_\alpha\left(\neg a_1 \vee \cdots \vee \neg a_m \vee \bar{b}\right) \\ P_\alpha\left(\neg a_1 \vee \cdots \vee \neg \bar{a}_m \vee b\right) & P_\alpha\left(\neg a_1 \vee \cdots \vee \neg \bar{a}_m \vee \bar{b}\right) \\ \vdots & \vdots \\ P_\alpha\left(\neg \bar{a}_1 \vee \cdots \vee \neg a_m \vee b\right) & P_\alpha\left(\neg \bar{a}_1 \vee \cdots \vee \neg a_m \vee \bar{b}\right) \\ \vdots & \vdots \\ P_\alpha\left(\neg \bar{a}_1 \vee \cdots \vee \neg \bar{a}_m \vee b\right) & P_\alpha\left(\neg \bar{a}_1 \vee \cdots \vee \neg \bar{a}_m \vee \bar{b}\right) \end{pmatrix}$$

for $a'_i = (a_i, \bar{a}_i), \quad a_i < \bar{a}_i, \quad 1 \leq i \leq m$ *and* $b' = (b, \bar{b})$, $b < \bar{b}$,

– *if* τ' *is a goal clause:*
$L'_\alpha(a'_i, \tau') = (\xi_i) \quad$ *for* $1 \leq i \leq m$.

Then

$$\mathcal{FPN}_\alpha := (S'_\alpha, T'_\alpha, F'_\alpha, P_\alpha, L'_\alpha)$$

is a folding of $\mathcal{PN}_\alpha$ *and is called a* higher-level probability propagation net.

Remark 3.1
This definition assumes that the dependencies between fact and goal clauses are completely described by the rules, such that the matrix $M_{\tau'}$ can always be constructed. In fact, probabilistic Horn formulas used in probabilistic Horn abduction should meet this requirement. If the set of rule clauses is incomplete, the missing probabilities must be added. For a rule clause $\neg a_1 \vee \neg a_2 \vee b$ of a Horn formula α for instance, the equation

$$P_\alpha(\neg a_1 \vee \neg a_2 \vee b) = 1 - P_\alpha(\neg a_1 \vee \neg a_2 \vee \bar{b})$$

must hold. ◇

Example 3.6 (Lack of Oil: Folded Probability Propagation Net)
The results of Example 2.8 can be calculated in a simpler way by folding the probability propagation net of Example 3.4 according to Definition 3.3. Figure 3.5 shows the

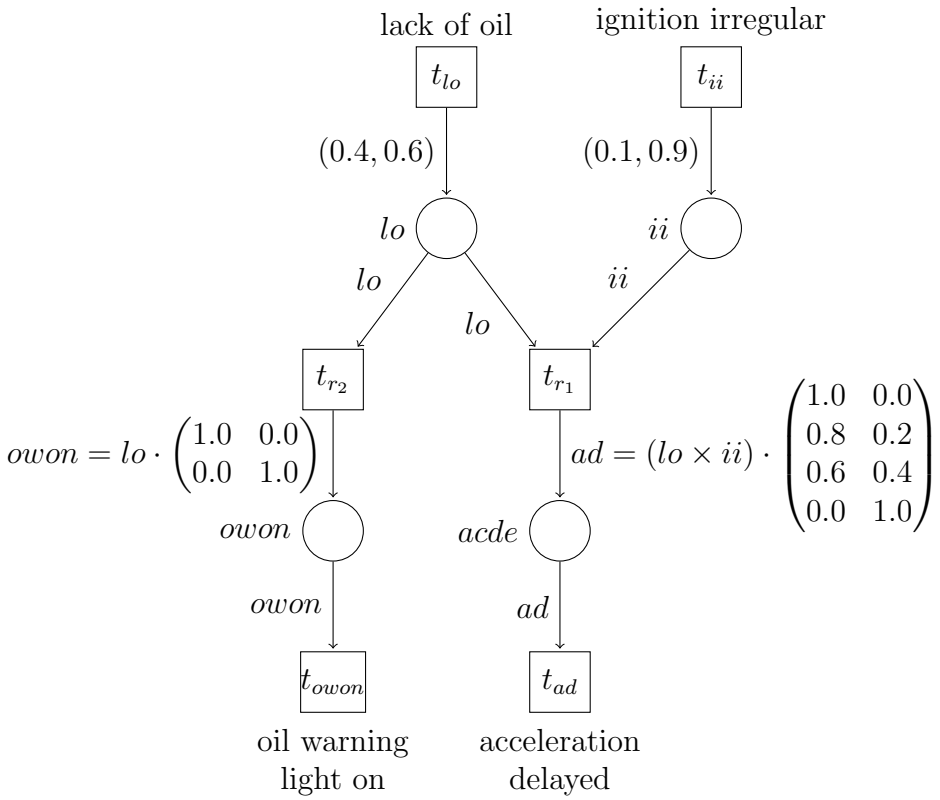


Figure 3.5.: Folded Probability Propagation Net $\mathcal{FPN}_\alpha$ of Example 3.6

$\mathcal{PN}_\alpha$	$\mathcal{FPN}_\alpha$
$lo, nolo$	lo
$igir, igno$	ii
$owon, owof$	$owon$
$acde, acno$	ad
t_{16}, t_{15}	t_{lo}
t_{14}, t_{13}	t_{ii}
t_{20}, t_{19}	t_{owon}
t_{18}, t_{17}	t_{ad}
$t_1, \ldots, t_8$	t_{r_1}
$t_9, \ldots, t_{12}$	t_{r_2}

Table 3.2.: Variable Mapping of the Folded Probability Propagation Net

higher-level probability propagation net $\mathcal{FPN}_\alpha$, which is a folding of the probability propagation net $\mathcal{PN}_\alpha$ depicted in Figure 3.3. The variable mapping of the folding is shown in Table 3.2. □

After defining the structure of higher-level probability propagation nets, their marking and the corresponding firing rule have to be defined, too:

Definition 3.4 (Higher-Level Probability Propagation Net Marking)
Let α be a Horn formula and $\mathcal{FPN}_\alpha = (S'_\alpha, T'_\alpha, F'_\alpha, P_\alpha, L'_\alpha)$ a higher-level probability propagation net for α; let W be a finite subset of $[0,1]$ and let $(W) := \{(w_1, w_2) \mid w_1, w_2 \in W\}$ be the corresponding set of pairs; let be $\tau \in T'_\alpha$ with ${}^\bullet\tau = \{s_1, \ldots, s_m\}, \tau^\bullet = \{s_{m+1}\}$ (that means $\tau = \neg s_1 \vee \cdots \vee \neg s_m \vee s_{m+1}$);
then $M : S'_\alpha \to \mathbb{M}((W))$ is a marking *of $\mathcal{FPN}_\alpha$;*
τ is enabled by *M for $\{(w_1), \ldots, (w_m)\}$ iff $(w_1) \in M(s_1), \ldots, (w_m) \in M(s_m)$,*
the follower marking *M' of M after one firing of τ for $\{(w_1), \ldots, (w_m)\}$ is given by*

- $M'(s_1) = M(s_1) - (w_1)$,
- $M'(s_2) = M(s_2) - (w_2)$,
- $\ldots$
- $M'(s_m) = M(s_m) - (w_m)$,
- *If τ is a fact transition and the arc label of (τ, s_{m+1}) is (p_1, p_2):*

$$M(s_{m+1}) + (p_1, p_2),$$

- *If τ is a rule transition and M_τ is the corresponding matrix (cf. Definition 3.3):*

$$M'(s_{m+1}) = M(s_{m+1}) + ((w_1 \times w_2 \times \cdots \times w_m) \cdot M_\tau),$$

- *If τ is a goal transition, there is no s_{m+1} because $\tau^\bullet = \varnothing$.*

If $(\xi_1), \ldots, (\xi_m)$ are the arc labels of $(s_1, \tau), \ldots, (s_m, \tau) \in F_\alpha$ and if the ξ_i are bound by the corresponding $w_i, 1 \leq i \leq m$, one may write

$$\begin{aligned} M'(s_1) &= M(s_1) - (\xi_1), \\ &\vdots \\ M'(s_m) &= M(s_m) - (\xi_m), \\ M'(s_{m+1}) &= M(s_{m+1}) + (p_1, p_2) \end{aligned}$$

for the firing of fact transitions with $p_1, p_2 \in W$ being the values of the corresponding arc label and

$$\begin{aligned} M'(s_1) &= M(s_1) - (\xi_1), \\ &\vdots \\ M'(s_m) &= M(s_m) - (\xi_m), \\ M'(s_{m+1}) &= M(s_{m+1}) + ((\xi_1 \times \cdots \times \xi_m) \cdot M_\tau) \end{aligned}$$

for the firing of rule transitions with M_τ being the corresponding matrix and

$$\begin{aligned} M'(s_1) &= M(s_1) - (\xi_1), \\ &\vdots \\ M'(s_m) &= M(s_m) - (\xi_m) \end{aligned}$$

for the firing of goal transitions.

This definition of markings and the firing rule for higher-level probability propagation nets will now be applied to the folded net modeling the Lack of Oil example. It will be shown how the probabilities are cumulatively calculated by the scalar and matrix products of the arc labels.

In higher-level probability propagation nets probabilities are again calculated by simulation of appropriate t-invariants and again, these t-invariants are related to the underlying p/t-net and thus trivial to find.

Example 3.7 (Lack of Oil: Simulation of the Higher-Level Net)
Consider the folded probability propagation net shown in Figure 3.5. Assume that the initial marking M_0 is the empty marking.

After firing of t_{lo} and t_{ii}, the marking changed to M_1 with

$$M_1(p) = \begin{cases} (0.4, 0.6) & \text{if } p = lo \\ (0.1, 0.9) & \text{if } p = ii \\ \varnothing & \text{else.} \end{cases}$$

If t_{r_1} fires, lo and ii are cleared and $ad = (lo \times ii) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.8 & 0.2 \\ 0.6 & 0.4 \\ 0.0 & 1.0 \end{pmatrix}$ is put on $acde$;

$$(lo \times ii) = ((0.4, 0.6) \times (0.1, 0.9)) = (0.04, 0.36, 0.06, 0.54)$$

$$\begin{aligned} ad &= (0.04, 0.36, 0.06, 0.54) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.8 & 0.2 \\ 0.6 & 0.4 \\ 0.0 & 1.0 \end{pmatrix} \\ &= (0.04 + 0.288 + 0.036 + 0.0, 0.0 + 0.072 + 0.024 + 0.54) \\ &= (0.364, 0.636) = (P(acde), P(\neg acde)) = (P(acde), P(acno)) \end{aligned}$$

Again, the values coincide with the values calculated

- by probabilistic Horn abduction (see [PorTor97]),
- by canonical Petri net representation of the Horn formula enhanced with the corresponding probability function (see Example 2.8),
- and by simulation of the corresponding low-level probability propagation net t-invariants (see Example 3.4).

Note that these values have been calculated by one pass through the net, that means in this case the simulation of one t-invariant suffices to calculate the probability for a delayed acceleration (and a normal acceleration, as well). □

In general, because of the folding, the calculation of probabilities for goal clauses in higher-level probability propagation nets requires less simulation steps than in the equivalent low-level probability propagation nets. The reason for that is the folding of rule clauses belonging together and their probabilities being represented in the matrix of the corresponding label, such that the single components' probabilities are calculated and summed up by the scalar and matrix product in one simulation step.

On the one hand, one may consider it a disadvantage that – in contrast to the low-level probability propagation net representation – single explanations for a goal

cannot be found or evaluated in this net representation, because only accumulated probabilities are calculated. On the other hand, the simplification of the net structure caused by the folding has its advantages, too. Besides the single pass calculation of probabilities, it is easier to find t-invariants as the net structure usually is much smaller.

By the folding rules described above and by the extended marking concept and firing rule not only probabilistic Horn formulas can be modeled. The approach can also be transferred to the world of Bayesian networks, such that so-called "high-level probability propagation nets" model the complete $\lambda-$ and $\pi-$message propagation in Bayesian networks. The next chapter describes how to represent Bayesian networks with probability propagation nets and shows the propagation process for the initialization phase and when evidence is brought into the system.

4. High-Level Probability Propagation Nets

Low-level probability propagation nets are suited to model propagations of single probabilities. In order to model more complex propagations such as π- and λ-messages in Bayesian networks, the probability propagation nets can be lifted to a higher level by special foldings. Instead of propagating single probabilities along the arcs, tuples can be propagated. Instead of multiplying random variables with single constants (for example conditional probabilities), the tuples being propagated can be multiplied with matrices or other tuples in a special way. By that, the complexity of low-level probability propagation nets is partially hidden in arc labels and propagations according to the message passing in Bayesian networks can be represented straight forward.

In section 4.1 the transformation rules for representing Bayesian networks with high-level probability propagation nets are defined. Some popular examples demonstrate the applicability and the advantages of the probability propagation net representation. The initialization phase is explained in section 4.2, evidence and change propagation in section 4.3. Another folding leads to probability propagation nets which are suited to model the propagation flows of conditioned Bayesian networks as described in section 4.4.

4.1. Transforming Bayesian Networks into High-Level Probability Propagation Nets

In essence three structural elements in Bayesian networks exist, which are shown in Figure 4.1. A simple *chain* is shown in Figure 4.1(a). It models the probabilistic dependency of Y given X. Example 4.2 contains this type of graph structure.

Figure 4.1(b) indicates the probabilistic dependency of Y given $X_1, \ldots, X_n$, $n > 1$. So, the probability of Y is defined as $P(Y \mid X_1, \ldots, X_n)$. Case $n = 2$ can be found in Example 4.1. This kind of graph structure is called *join*.

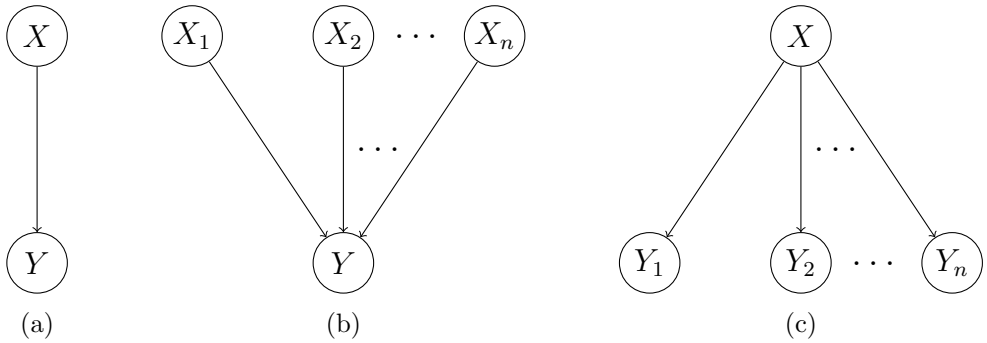


Figure 4.1.: Basic Structures in Bayesian Networks

The structure shown in Figure 4.1(c) is called *split* and represents probabilistic dependencies of variables $Y_1, \ldots, Y_n$ given X, $n > 1$. Example 4.2 is to throw light on this structural element with $n = 2$.

In fact, from a structural point of view, a chain can be defined as a special kind of join or split, namely for $n = 1$. For defining transformation rules it is more comfortable to consider it as a special kind of join, because in this case the rules for joins can be applied straight forward. Formally, joins and splits as substructures of Bayesian networks can be defined as follows:

Definition 4.1
Let $\mathcal{B} = (R, E)$ be a Bayesian network, let be $R' := \{X_1, \ldots, X_n, Y\}$ with $R' \subseteq R, n > 0$. Let be $E' := \{(X_i, Y) \mid (X_i, Y) \in E\}$, such that $E' \subseteq E$ and $|E'| = n$ and $\nexists V \in (R \setminus R') : (V, Y) \in E$.

The tuple (R', E') is called

- *a* (Bayesian network) chain, *if $n = 1$ (see Figure 4.1(a))*
- *a* (Bayesian network) join, *if $n > 1$ (see Figure 4.1(b)).*

The different structures and their transformations into the Petri net representation are demonstrated by means of two popular examples given in [Neap90]. The following example shows a Bayesian network join.

Example 4.1 (Burglar Alarm, cf. [Neap90])
In this example, Mr. Holmes is sitting in his office when he gets a call that his burglar alarm is sounding (A). Of course he suspects a burglary (B) (even though there might be other reasons for activating the alarm, for example an earthquake (C)). On his ride home, he hears on the radio an announcement about some earthquake. How do the phone call and the radio announcement influence his belief about getting burglarized?

The Bayesian network $\mathcal{B}$ with prior and conditional probabilities is shown in Figure 4.2. The random variables A, B, C have two attributes ($-_1$ and $-_2$ meaning "yes" and "no") listed in Table 4.1.

a_1	Mr. Holmes' burglar alarm sounds
a_2	Mr. Holmes' burglar alarm does not sound
b_1	Mr. Holmes' residence is burglarized
b_2	Mr. Holmes' residence is not burglarized
c_1	there is an earthquake
c_2	there is no earthquake

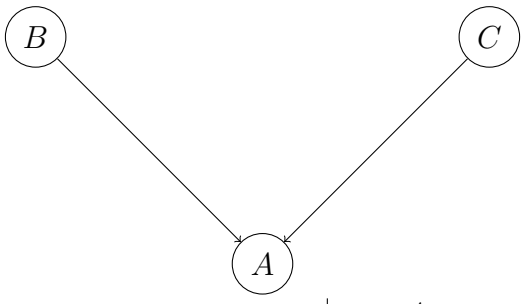
Table 4.1.: Random Variables of Example 4.1

$P(B) =$

B	1	2
	0.01	0.99

$P(C) =$

C	1	2
	0.001	0.999



$P(A \mid BC) =$

		A	
B	C	1	2
1	1	0.99	0.01
1	2	0.9	0.1
2	1	0.5	0.5
2	2	0.01	0.99

Figure 4.2.: Bayesian Network $\mathcal{B}$ of Example 4.1

The structure of this Bayesian network obviously is a join. □

Definition 4.2
Let $\mathcal{B} = (R, E)$ *be a Bayesian network, let be* $R' := \{X, Y_1, \ldots, Y_n\}$ *with* $R' \subseteq R, n > 1$. *Let be* $E' := \{(X, Y_i) \mid (X, Y_i) \in E\}$ *such that* $E' \subseteq E$ *and* $|E'| = n$ *and* $\nexists V \in (R \setminus R') : (X, V) \in E$.

The tuple (R', E') *is called a* (Bayesian network) split *(see Figure 4.1(c)).*

Example 4.2 (Cheating Spouse, cf. [Neap90])
The scenario of this popular example given in [Neap90] consists of a spouse and a strange man/lady. It has to be reported that spouse might be cheating. As a consequence, there are four important random variables: spouse is cheating (A), spouse dines with another (B), spouse is reported seen dining with another (C), strange man/lady calls on the phone (D).

a_1	spouse is cheating
a_2	spouse is not cheating
b_1	spouse dines with another
b_2	spouse does not dine with another
c_1	spouse is reported seen dining with another
c_2	spouse is not reported seen dining with another
d_1	strange man/lady calls on the phone
d_2	no strange man/lady calls on the phone

Table 4.2.: Random Variables of Example 4.2

The Bayesian network with prior and conditional probabilities is shown in Figure 4.3. The random variables A, B, C, D have two attributes ($_{-1}$ and $_{-2}$ meaning "yes" and "no") which are listed in Table 4.2.

This Bayesian network's structure is a combination of a split and a chain. □

Every Bayesian network can be recursively constructed with the basic structure types shown in Figure 4.1, because due to the fact that Bayesian networks are acyclic graphs, there can neither be any combination of nodes and edges containing incoming as well as outgoing edges between the same nodes nor any other composition that would lead to a cycle in the graph structure. Hence, for creating a Petri net representation of Bayesian networks, it is sufficient first to find transformations for these basic structures and second to develop rules to put the different parts together.

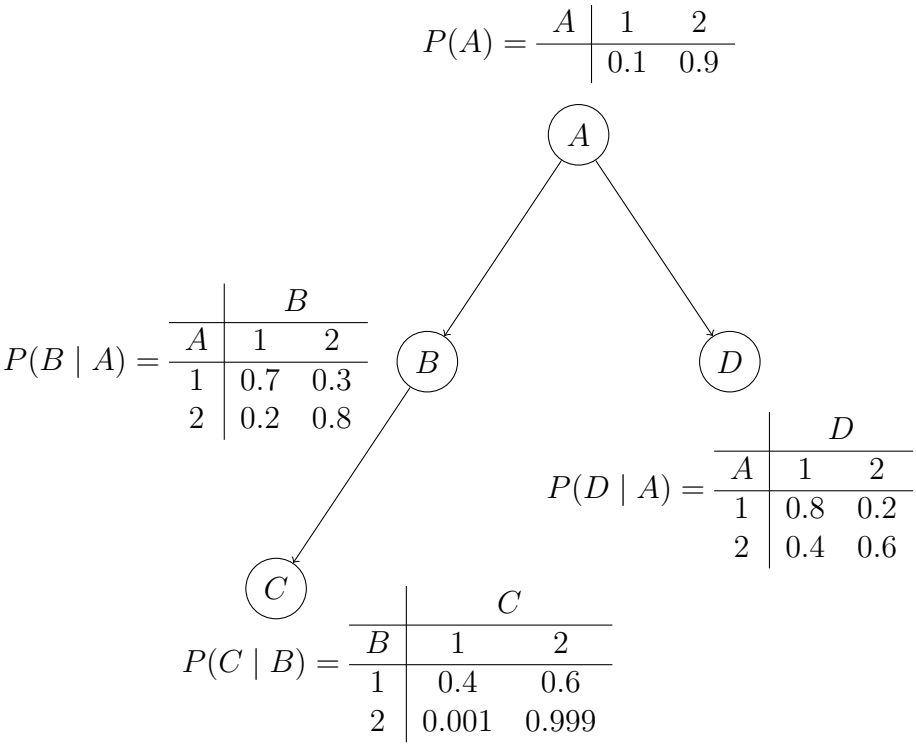


Figure 4.3.: $\mathcal{B}$ of Example 4.2

4.1.1. Transforming Joins

The transformation rule for transforming chains (see Figure 4.1(a)) and joins (see Figure 4.1(b)) is defined as follows:

Definition 4.3 (Chain and Join Transformation)
Let $\mathcal{B} = (R, E)$ be a Bayesian network, let (R', E') be a chain or join contained in $\mathcal{B}$ (and therefore $R' \subseteq R$ and $E' \subseteq E$). Let $X_1, \ldots, X_n$ be the parent nodes and Y be the child node, such that $R' = \{X_1, \ldots, X_n, Y\}$.

The transformation rule for representing the Bayesian network join (chain) with a probability propagation net is defined as follows:

- *For every $V \in R'$ create two corresponding places V^π and V^λ.*
- *Create $n + 1$* functional transitions *$f_Y^0, f_Y^1, \ldots, f_Y^n$.*
- *For every $X_i \in \{X_1, \ldots, X_n\}$ create an edge from X_i^π to f_Y^0 labeled with $\pi_Y(X_i)$.*

- *Create an edge from* f_Y^0 *to* Y^π *labeled with* $\pi(Y)$.
- *For every* $X_i \in \{X_1, \ldots, X_n\}$ *create the following edges:*
  - *One edge from* f_Y^i *to* X_i^λ *labeled with* $\lambda_Y(X_i)$.
  - *One edge from* Y^λ *to* f_Y^i *labeled with* $\lambda(Y)$.
  - *For every* $j \in \mathbb{N}, 1 \leq j \leq n, j \neq i$ *create an edge from* X_i^π *to* f_Y^j *labeled with* $\pi_Y(X_i)$ *(in case of a chain, there will be no* j *matching this condition and thus only the first two edges mentioned in this step will be created).*

The Petri net consisting of the places, functional transitions, edges and edge labels created by the rules above is called the probability propagation net representation of the Bayesian network join (chain) *(see Figure 4.4).*

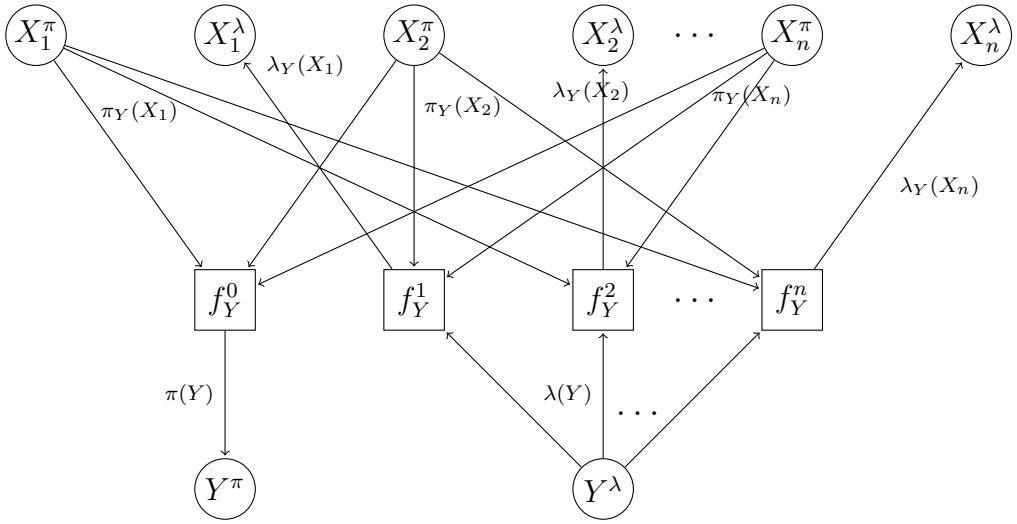


Figure 4.4.: Probability Propagation Net Representation of a Bayesian Network Join

Remark 4.1

The boundary of this Petri net consists of places. To transform Bayesian networks completely into probability propagation nets, the final step will be to add appropriate boundary transitions in order to make the net transition-bounded. By that, the

structure is an adequate representation of the message propagation because every t-invariant of the Petri net corresponds to a propagation path of the Bayesian network. This will be shown later on. ◇

Remark 4.2

To simplify the drawing of high-level probability propagation nets, it is possible to omit the superscripted λ and π symbols of the place names, because the "category" of a place (that means if it holds λ- or π-tuples) is closely related to the arc labels and the net structure and thus can be determined apparently. Instead of naming places Y_1^π and Y_1^λ, for example, both places can be named Y_1. Nevertheless, to define the transformation rules, these places must be distinguishable. Therefore the formal definitions (Definition 4.3 and Definition 4.4) contain the superscripts. ◇

Example 4.3 (Burglar Alarm: Probability Propagation Net)

The Petri net version $\mathcal{PB}$ of $\mathcal{B}$ – which is a Bayesian network join, see Figure 4.2 – is shown in Figure 4.5.

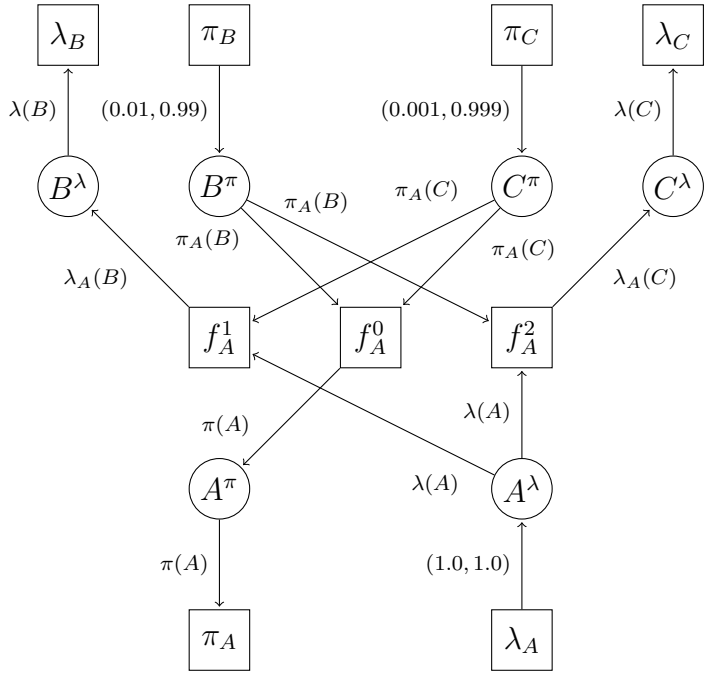


Figure 4.5.: $\mathcal{PB}$ of Example 4.1

The boundary consists of additional transitions, as mentioned in the remarks above.

In section 4.1.4 it is described how to create the transition boundary. □

Now, it will be shown how splits can be transformed into an adequate Petri net representation.

4.1.2. Transforming Splits

The transformation rule for splits (see Figure 4.1(c)) is a bit more complex than the one for chains and joins. This is due to the more complex message propagations as messages have to be combined with other knowledge or other messages before sending them to the appropriate destination node. For this reason, firstly the parent node of the Bayesian network split is transformed into more than two places in the Petri net representation. Secondly, so-called "multiplicative transitions" are inserted at an intermediate level between parent and child nodes to combine the propagation flows appropriately.

Definition 4.4 (Split Transformation)
Let $\mathcal{B} = (R, E)$ be a Bayesian network, let (R', E') be a split contained in $\mathcal{B}$ (and therefore $R' \subseteq R$ and $E' \subseteq E$). Let X be the parent node and $Y_1, \ldots, Y_n$ be the child nodes, such that $R' = \{X, Y_1, \ldots, Y_n\}$.

The transformation rule for representing the Bayesian network split with a probability propagation net is defined as follows:

- *Create two places X^π and X^λ.*
- *For every $Y_i \in \{Y_1, \ldots, Y_n\} \subset R'$ create two corresponding places Y_i^π and Y_i^λ and create two additional places $X_{Y_i}^\pi$ and $X_{Y_i}^\lambda$, which are called* split places of X referring Y_i.
- *Create $n + 1$* multiplicative transitions $m_X^0, m_X^1, \ldots, m_X^n$.
- *For every $Y_i \in \{Y_1, \ldots, Y_n\} \subset R'$ create two functional transitions $f_{Y_i}^0$ and $f_{Y_i}^1$.*
- *Create one edge from the first multiplicative transition m_X^0 to X^λ labeled with $\lambda(X)$ and for every $i = 1 \ldots n$ create one edge from $X_{Y_i}^\lambda$ to m_X^0 labeled with $\lambda_{Y_i}(X)$.*

- *For the last n multiplicative transitions m_X^i, $i = 1 \dots n$ create the following edges:*
  - *One edge from X^π to m_X^i labeled with $\pi(X)$.*
  - *One edge from m_X^i to $X_{Y_i}^\pi$ labeled with $\pi_{Y_i}(X)$.*
  - *For every $j \in \mathbb{N}, 1 \leq j \leq n, j \neq i$ create one edge from $X_{Y_j}^\lambda$ to m_X^i labeled with $\lambda_{Y_j}(X)$.*
- *For every $i = 1 \dots n$ create the following four edges:*
  - *One edge from $X_{Y_i}^\pi$ to $f_{Y_i}^0$ labeled with $\pi_{Y_i}(X)$.*
  - *One edge from $f_{Y_i}^0$ to Y_i^π labeled with $\pi(Y_i)$.*
  - *One edge from Y_i^λ to $f_{Y_i}^1$ labeled with $\lambda(Y_i)$.*
  - *One edge from $f_{Y_i}^1$ to $X_{Y_i}^\lambda$ labeled with $\lambda_{Y_i}(X)$.*

The Petri net consisting of the places, multiplicative and functional transitions, edges and edge labels created by the rules above is called the probability propagation net representation of the Bayesian network split *(see Figure 4.6).*

Example 4.4 (Cheating Spouse: Probability Propagation Net)
The Bayesian network of the cheating spouse example is a composition of a split and a chain. The corresponding probability propagation net is shown in Figure 4.7 which also gives a first impression on how the two probability propagation nets representing the split and the chain, respectively, can be combined. This is described in detail in section 4.1.4. □

Considering the edge labels, it has to be admitted that they are not yet very meaningful but it is easier to understand the transformation as a first approach. Labels can be improved by inscribing functions (see section 3.3) which corresponds to the operations of message propagation rules in Bayesian networks (see section 2.3.2) or by interpreting them as variables containing knowledge about the modeled system (for example evidence or observations). This will be shown in the next section.

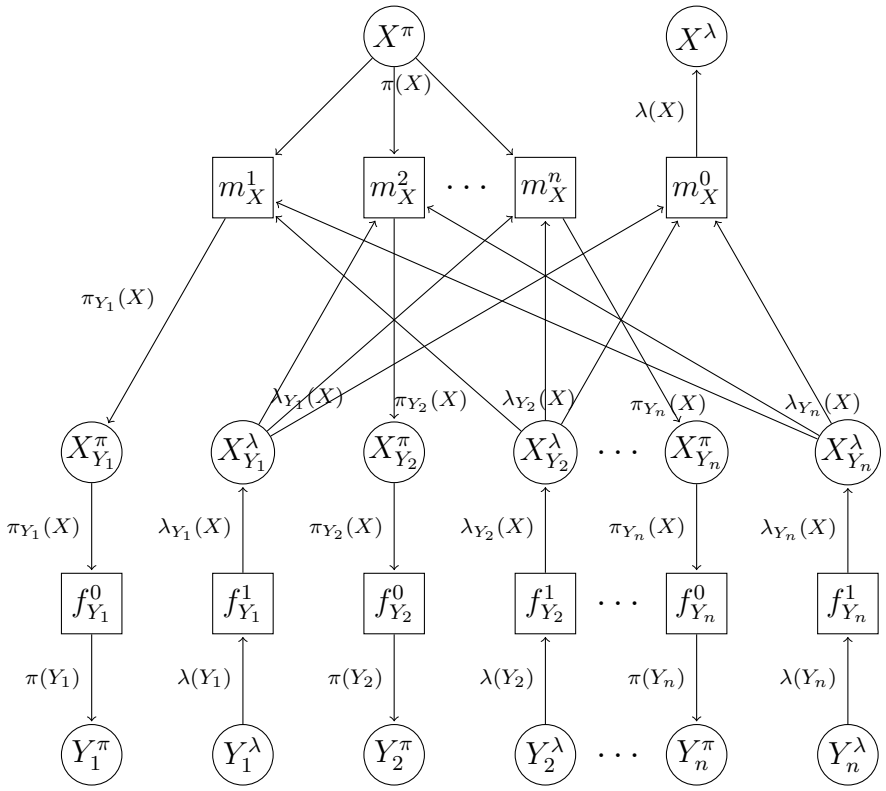


Figure 4.6.: Probability Propagation Net Representation of a Bayesian Network Split

4.1.3. Enhancing Labels to represent Message Propagation

To represent probability propagation according to the message passing in Bayesian networks, the labels of edges connected to functional and multiplicative transitions have to be enhanced.

There are transitions in the probability propagation nets which are closely related to the conditional probabilities in Bayesian networks. The functional transition[1] $f_Y^0 \simeq P(Y \mid X_1, \ldots, X_n)$ indicates f_Y^0 to be the transition that calculates $P(Y \mid X_1, \ldots, X_n)$. The superscript 0 points to a conditional probability. Missing superscripts denote prior probabilities. Superscripts ≥ 1 point to (generalized) transposes (or different sortings) of the probability tables (see Tables 4.3 and 4.4 and the following definition).

Definition 4.5 (Conditional Probability Matrix)
Let $\mathcal{B} = (R', E')$ be a Bayesian network chain (join, split). For every $Y \in R'$ let

[1] see Definition 4.3

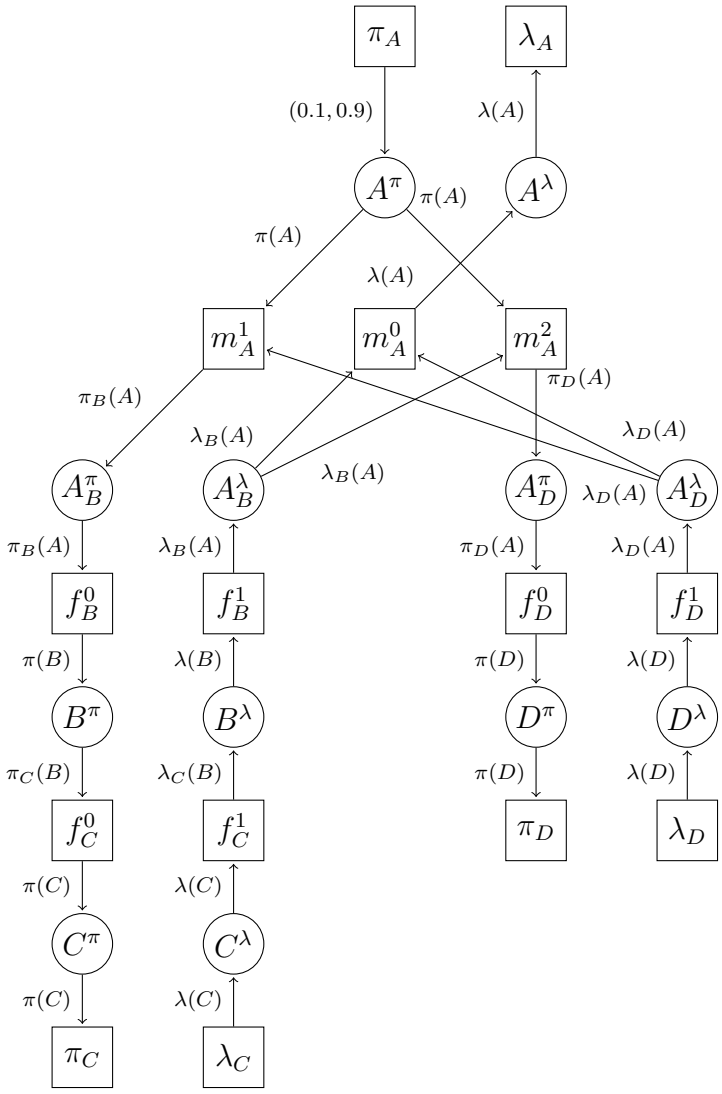


Figure 4.7.: $\mathcal{PB}$ of Example 4.2

CPT_Y be the corresponding conditional probability table assigning the conditional probability $P(Y \mid X_1, \ldots, X_n)$ of Y to the input variables $X_1, \ldots, X_n$. (If Y does not have any input nodes, then $P(Y)$ is the prior probability of Y.)

Let X_i have m_i disjoint values $\{x_i^j \mid j = 1 \ldots m_i\}$ for $i = 1 \ldots n$ and let Y have k disjoint values $y^1, \ldots, y^k$.

Let the conditional probability table CPT_Y representing the conditional probability $P(Y \mid X_1, \ldots, X_n)$ be given as follows:

			Y		
X_1	$\dots$	X_n	y^1	$\dots$	y^k
x_1^1	$\dots$	x_n^1	$P(Y = y^1 \mid X_1 = x_1^1, \dots, X_n = x_n^1)$	$\dots$	$P(Y = y^k \mid X_1 = x_1^1, \dots, X_n = x_n^1)$
x_1^1	$\dots$	x_n^2	$P(Y = y^1 \mid X_1 = x_1^1, \dots, X_n = x_n^2)$	$\dots$	$P(Y = y^k \mid X_1 = x_1^1, \dots, X_n = x_n^2)$
	$\ddots$			$\ddots$	
x_1^1	$\dots$	$x_n^{m_n}$	$P(Y = y^1 \mid X_1 = x_1^1 \dots, X_n = x_n^{m_n})$	$\dots$	$P(Y = y^k \mid X_1 = x_1^1, \dots, X_n = x_n^{m_n})$
x_1^2	$\dots$	x_n^1	$P(Y = y^1 \mid X_1 = x_1^2, \dots, X_n = x_n^1)$	$\dots$	$P(Y = y^k \mid X_1 = x_1^2, \dots, X_n = x_n^1)$
	$\ddots$			$\ddots$	
x_1^2	$\dots$	$x_n^{m_n}$	$P(Y = y^1 \mid X_1 = x_1^2 \dots, X_n = x_n^{m_n})$	$\dots$	$P(Y = y^k \mid X_1 = x_1^2, \dots, X_n = x_n^{m_n})$
	$\ddots$			$\ddots$	
$x_1^{m_1}$	$\dots$	x_n^1	$P(Y = y^1 \mid X_1 = x_1^{m_1}, \dots, X_n = x_n^1)$	$\dots$	$P(Y = y^k \mid X_1 = x_1^{m_1}, \dots, X_n = x_n^1)$
	$\ddots$			$\ddots$	
$x_1^{m_1}$	$\dots$	$x_n^{m_n}$	$P(Y = y^1 \mid X_1 = x_1^{m_1} \dots, X_n = x_n^{m_n})$	$\dots$	$P(Y = y^k \mid X_1 = x_1^{m_1}, \dots, X_n = x_n^{m_n})$

The matrix CPM_Y^0 consisting of the conditional probabilities in the same order as given in the conditional probability table (that means consisting of the lower right part of the table) is called the conditional probability matrix for Y. *Hence, the dimension of CPM_Y^0 is $((\prod_{i=1}^n m_i) \times k)$.*

Additionally, different sortings of the original conditional probability table are needed in which instead of Y a variable X_j, $j \in \{1, \dots, n\}$ is the target variable and Y is placed at the beginning of the input variables.

Let $CPT_Y^{X_j}$ be a different sorting of CPT_Y, such that for $a_i = 1, \dots, m_i$ with $i = 1, \dots, n$ and for $l = 1, \dots, k$ the table entries look as follows:

							X_j		
Y	X_1	$\dots$	X_{j-1}	X_{j+1}	$\dots$	X_n	x_j^1	$\dots$	$x_j^{m_j}$
			$\vdots$					$\vdots$	
y^l	$x_1^{a_1}$	$\dots$	$x_{j-1}^{a_{j-1}}$	$x_{j+1}^{a_{j+1}}$	$\dots$	$x_n^{a_n}$	$\mathbf{P_1^l}$	$\dots$	$\mathbf{P_{m_j}^l}$
			$\vdots$					$\vdots$	

with

$$\mathbf{P_1^l} := P(Y = y^l \mid X_1 = x_1^{a_1}, \dots X_{j-1} = x_{j-1}^{a_{j-1}}, \\ X_j = x_j^1, X_{j+1} = x_{j+1}^{a_{j+1}}, \dots, X_n = x_n^{a_n})$$

and

$$\mathbf{P}^{\mathbf{l}}_{\mathbf{m_j}} := P(Y = y^l \,|X_1 = x_1^{a_1}, \dots X_{j-1} = x_{j-1}^{a_{j-1}}, \\ X_j = x_j^{m_j}, X_{j+1} = x_{j+1}^{a_{j+1}}, \dots, X_n = x_n^{a_n}).$$

If X_j is the target variable of $CPT_Y^{X_j}$, the matrix CPM_Y^j consists of the conditional probabilities in the same order as given in $CPT_Y^{X_j}$ and thus is called a different sorting of CPM_Y^0 with target variable X_j.

Altogether, the matrices $CPM_Y^1, \dots, CPM_Y^n$ are different sortings of the conditional probability matrix CPM_Y^0, such that for CPM_Y^j, with $j = 1, \dots, n$, in the corresponding conditional probability table X_j is the target variable and Y becomes the first input variable. The dimension of CPM_Y^j is $\left((k \cdot (\prod_{i=1;i\neq j}^{n} m_i)) \times m_j\right)$.

These matrices are needed to represent the message propagation in Bayesian networks with Petri nets. For every conditional probability table of a Bayesian network node which is connected to another node, there exist several functional transitions in the probability propagation net. Until now, the corresponding edge labels simply consist of the name of the message which is propagated via the connected functional transition. By making use of the conditional probability matrices, the operations for message propagation can be expressed as extensions of the arc labels:

Definition 4.6 (Functional Transition Operations)
Let f_Y^0 be a functional transition of a probability propagation net representation of a Bayesian network chain (join, split) $\mathcal{B} = (R, E)$ connecting the input places $X_1^\pi, \dots, X_n^\pi$ ($n \in \mathbb{N}$) with the output place Y^π. Let $Y \in R$ be the node in the Bayesian network chain (join, split) corresponding to the places Y^π and Y^λ, and let Y have k disjoint values.

Thus, $f_Y^1, \dots, f_Y^n$ are the related functional transitions, such that for every $j = 1, \dots, n$ the postset of f_Y^j is $\{X_j^\lambda\}$ (see Figure 4.4 and Figure 4.6).

Let furthermore CPM_Y^0 be the conditional probability matrix corresponding to the conditional probability table CPT_Y assigned to Y.

Given the input probabilities $\pi_Y(X_i)$ as defined in Definition 2.21, the transition

f_Y^0 calculates $\pi(Y) := P(Y \mid X_1, \ldots, X_n)$ according to the following formula:

$$\pi(Y) = \big(\pi_Y(X_1) \times \pi_Y(X_2) \times \cdots \times \pi_Y(X_n)\big) \cdot CPM_Y^0$$

This is denoted $f_Y^0 \simeq P(Y \mid X_1, \ldots, X_n)$.

For $j = 1, \ldots, n$ and $\lambda(Y)$ being the current λ-message of Y (that means a vector of length k, see Definition 2.20), the transition f_Y^j operates as follows:

$$\begin{aligned}\lambda_Y(X_j) := \big(&\lambda(Y) \times \pi_Y(X_1) \times \pi_Y(X_2) \times \ldots \\ &\times \pi_Y(X_{j-1}) \times \pi_Y(X_{j+1}) \times \cdots \times \pi_Y(X_n)\big) \cdot CPM_Y^j\end{aligned}$$

This is denoted $f_Y^j \simeq P^{X_j} \leftarrow YX_1 \ldots X_{j-1}X_{j+1} \ldots X_n(Y \mid X_1, \ldots, X_n)$.

By that, the label inscriptions for edges connected to functional transitions are defined as follows:

- *Edges from an input place X_i^π to a functional transition keep their label $\pi_Y(X_i)$ ($i = 1, \ldots, n$).*
- *The edge from f_Y^0 to Y^π is labeled with $\pi(Y) = \big(\pi_Y(X_1) \times \pi_Y(X_2) \times \cdots \times \pi_Y(X_n)\big) \cdot CPM_Y^0$.*
- *The outgoing edges from Y^λ to the functional transitions $f_Y^1, \ldots f_Y^n$ keep their labels $\lambda(Y)$.*
- *Every edge from a functional transition f_Y^j with $j \in \{1, \ldots, n\}$ to the corresponding output place X_j^λ is labeled with $\lambda_Y(X_j) = \big(\lambda(Y) \times \pi_Y(X_1) \times \pi_Y(X_2) \times \cdots \times \pi_Y(X_{j-1}) \times \pi_Y(X_{j+1}) \times \cdots \times \pi_Y(X_n)\big) \cdot CPM_Y^j$.*

Remark 4.3
If for a Bayesian network node Y there are plenty of associated inbound nodes X_i (see definitions above), the edge labels can become too complex to be completely annotated in the net. In such cases the short version of the edge label (that means either $\pi(Y)$ or $\lambda_Y(X_i)$) is used as an abbreviation. ◇

$f_A^0 \simeq P(A \mid BC) =$

		A	
B	C	1	2
1	1	0.99	0.01
1	2	0.9	0.1
2	1	0.5	0.5
2	2	0.01	0.99

$\pi_B \simeq P(B) =$

B	
1	2
0.01	0.99

$\pi_C \simeq P(C) =$

C	
1	2
0.001	0.999

$f_A^1 \simeq P^{B \leftarrow AC}(A \mid BC) =$

		B	
A	C	1	2
1	1	0.99	0.5
1	2	0.9	0.01
2	1	0.01	0.5
2	2	0.1	0.99

$f_A^2 \simeq P^{C \leftarrow AB}(A \mid BC) =$

		C	
A	B	1	2
1	1	0.99	0.9
1	2	0.5	0.01
2	1	0.01	0.1
2	2	0.5	0.99

Table 4.3.: Transition Functions of Example 4.1

Example 4.5 (Burglar Alarm: Transition Functions)
The transition functions of Example 4.3 are shown in Table 4.3. For example, f_A^2 is the transition that belongs to the sorting of ($f_A^0 \simeq$) $P(A \mid BC)$ where C is written as depending on A and B, in symbols $P^{C \leftarrow AB}(A \mid BC)$, (for example the value belonging to $A = 2, B = 1, C = 2$ is equal to 0.1 in both tables). For $n = 1$ the sortings coincide with the transpose of a matrix (like f_B^0 and f_B^1 in Table 4.4). □

The transitions in the probability propagation nets representing structure elements according to Figure 4.1(c) cause a multiplication of vectors with equal length by components according to Definition 2.7.

Definition 4.7 (Multiplicative Transition Operations)
Let $\mathcal{B} = (R, E)$ be a Bayesian network split with $R = \{X, Y_1, \ldots, Y_n\}$ and $E = \{(X, Y_i) \mid i = 1, \ldots, n\}$. Let X have m disjoint values.

Let $m_X^0, \ldots m_X^n$ be the multiplicative transitions of the split's probability propa-

gation net representation. The λ- and π-messages being propagated by the edges connected to the multiplicative transitions are vectors of length m, too.

Then, by the multiplicative transition m_X^0, the vectors $\lambda_{Y_1}(X), \ldots, \lambda_{Y_n}(X)$ are transformed into

$$\begin{aligned}\lambda(X) &:= \lambda_{Y_1}(X) \circ \cdots \circ \lambda_{Y_n}(X) \\ &= \big(\prod_{i=1}^{n} (\lambda_{Y_i}(X))^1, \ldots, \prod_{k=1}^{n} (\lambda_{Y_k}(X))^m\big)\end{aligned}$$

(see Definition 2.7).

For $j = 1, \ldots n$, the multiplicative transition m_X^j transforms the vectors

$$\pi(X), \lambda_{Y_1}(X), \ldots, \lambda_{Y_{j-1}}(X), \lambda_{Y_{j+1}}(X), \ldots, \lambda_{Y_n}(X)$$

into

$$\begin{aligned}\pi_{Y_j}(X) &:= \pi(X) \circ \lambda_{Y_1}(X) \circ \cdots \circ \lambda_{Y_{j-1}}(X) \circ \lambda_{Y_{j+1}}(X) \circ \cdots \circ \lambda_{Y_n}(X) \\ &= \pi(X) \circ \big(\prod_{\substack{i=1,\\ i\neq j}}^{n} (\lambda_{Y_i}(X))^1, \ldots, \prod_{\substack{k=1,\\ k\neq j}}^{n} (\lambda_{Y_k}(X))^m\big) \\ &= \big((\pi(X))^1 \cdot \prod_{\substack{i=1,\\ i\neq j}}^{n} (\lambda_{Y_i}(X))^1, \ldots, (\pi(X))^m \cdot \prod_{\substack{k=1,\\ k\neq j}}^{n} (\lambda_{Y_k}(X))^m\big).\end{aligned}$$

By that, the label inscriptions for edges connected to multiplicative transitions are defined as follows:

- *Every edge from the places $X_{Y_i}^{\lambda}$ to a multiplicative transition keeps its edge label $\lambda_{Y_i}(X)$ for $i = 1, \ldots, n$.*
- *The edge from m_X^0 to X^{λ} is labeled with $\lambda(X) = \lambda_{Y_1}(X) \circ \cdots \circ \lambda_{Y_n}(X)$.*
- *The outgoing edges of X^{π} keep their labels $\pi(X)$.*
- *For every $j = 1, \ldots, n$, the edge from multiplicative transition m_X^j to the place $X_{Y_j}^{\pi}$ is labeled with $\pi_{Y_j}(X) = \pi(X) \circ \lambda_{Y_1}(X) \circ \cdots \circ \lambda_{Y_{j-1}}(X) \circ \lambda_{Y_{j+1}}(X) \circ \cdots \circ \lambda_{Y_n}(X)$.*

Remark 4.4

m_x^0 is the only transition that causes a product of only λ-factors. Superscripts ≥ 1 belong to mixtures of λ- and π-factors. $\diamond$

Remark 4.5

Again, if there are too many nodes Y_i involved in the split of X, the short version of the edge label (that means either $\lambda_Y(X)$ or $\pi_{Y_i}(X)$) is used as an abbreviation. $\diamond$

Remark 4.6

Considering the normalization of vectors representing λ- and π-messages, there are different alternatives. Usually, λ-messages are not normalized whereas π-messages are (cf. [Neap90]).

In this thesis, the vectors or matrices (see section 4.4) are normalized whenever necessary but the normalizing constants are omitted in the arc labels. $\diamond$

$\pi_A \simeq P(A) =$

	A	
	1	2
	0.1	0.9

$f_B^0 \simeq P(B \mid A) =$

	B	
A	1	2
1	0.7	0.3
2	0.2	0.8

$f_B^1 \simeq P^{A \leftarrow B}(B \mid A) =$

	A	
B	1	2
1	0.7	0.2
2	0.3	0.8

$f_C^0 \simeq P(C \mid B) =$

	C	
B	1	2
1	0.4	0.6
2	0.001	0.999

$f_C^1 \simeq P^{B \leftarrow C}(C \mid B) =$

	B	
C	1	2
1	0.4	0.001
2	0.6	0.999

$f_D^0 \simeq P(D \mid A) =$

	D	
A	1	2
1	0.8	0.2
2	0.4	0.6

$f_D^1 \simeq P^{A \leftarrow D}(D \mid A) =$

	A	
D	1	2
1	0.8	0.4
2	0.2	0.6

Table 4.4.: Transition Functions of Example 4.2

Example 4.6 (Cheating Spouse: Transition Functions)
Table 4.4 shows the transition functions of the cheating spouse example. As mentioned before, the different sortings of conditional probability matrices having exactly one input and one output variable simply are transposes of the original matrix. By applying the rules of Definition 4.7, the labels of the edges connected to the multiplicative transitions change as follows:

- $\lambda(A)$ changes to $\lambda(A) = \lambda_B(A) \circ \lambda_D(A)$,
- $\pi_B(A)$ changes to $\pi_B(A) = \pi(A) \circ \lambda_D(A)$,
- $\pi_D(A)$ changes to $\pi_D(A) = \pi(A) \circ \lambda_B(A)$.

The complete labels including the ones of the functional transitions are shown in the t-invariants (see Figure 4.15 to Figure 4.17). □

In the following section, it is explained how complete Bayesian networks can be translated into high-level probability propagation nets by composing the single probability propagation net representations of the Bayesian network's basic structures.

4.1.4. Composing High-Level Probability Propagation Nets

The basic idea of connecting single probability propagation net representations of basic Bayesian network structures is to take a look at the incoming and outgoing edges in the Bayesian network. In general, edges of the Petri net representation of a Bayesian network node X are connected to the places X^π or X^λ. But if there is a combination of a split and another basic structure, for example if X splits up into Y_1 and Y_2 and besides X, Y_2 has a second incoming node Z, the edges of the Petri net corresponding to the connection of X and Y_2 must be connected to $X^\pi_{Y_2}$ or $X^\lambda_{Y_2}$ instead of X^π or X^λ.

Altogether, in a Bayesian network incoming edges of a Bayesian network node are always related to its π- and λ-places in the Petri net. But if there is a split, outgoing edges are related to the π- and λ-split places referring the corresponding child node.

Definition 4.8 (High-Level Probability Propagation Net)
Let be $\mathcal{B} = (R, E)$ a loop-free Bayesian network. For every chain (join, split) $\mathcal{B}' = (R', E')$ contained in $\mathcal{B}$ with $R' \subseteq R$ and $E' \subseteq E$ create the corresponding probability propagation net representation considering the following additional constraint:

When transforming a join $X_1, \ldots, X_n \rightarrow Y$ ($n \in \mathbb{N}, n > 1$), check for all parent nodes X_i with $1 \leq i \leq n$ if X_i has more than one child node. If X_i has at least two child nodes (and thus is the root node of a split), then the Petri net edges related to the Bayesian network edge between X_i and Y must be connected to X_{iY}^{π} and X_{iY}^{λ} instead of X^{π} or X^{λ}, when transforming the join. Additionally, the edges between $\{X_{iY}^{\pi}, X_{iY}^{\lambda}\}$ and $\{f_Y^0, f_Y^1\}$ generated by the split transformation are omitted, because they are substituted by the edges of the join transformation.

Let S be the set of places, T be the set of transitions, $F \subset ((S \times T) \cup (T \times S))$ the set of edges and L the set of edge labels created by the rules above.

Then, $\mathcal{PB} := (S, T, F, L)$ is a Petri net which is place-bounded, until now. Make finally the net $\mathcal{PB}$ transition-bounded by adding the following transitions and edges:

- *For every boundary place X^{π}*
  - *create a transition π_X,*
  - *if X^{π} has no incoming edges, create an edge from π_X to X^{π} labeled with $\pi(X) = CPM_X^0$ (which would be $\pi(X) = \big(P(X = x^1), P(X = x^2), \ldots, P(X = x^m)\big)$ if m is the number of disjoint values of X),*
  - *if X^{π} has no outgoing edges, create an edge from X^{π} to π_X labeled with $\pi(X)$*
- *For every boundary place X^{λ}*
  - *create a transition λ_X,*
  - *if X^{λ} has no incoming edges, create an edge from λ_X to X^{λ} labeled with $\lambda(X) = \big(L(X = x^1), L(X = x^2), \ldots, L(X = x^m)\big)$, if m is the number of disjoint values of X, where $L(X = x_i)$ is the likelihood of X being x_i given all current knowledge, observations and evidences of the modeled system (see Definition 2.18),*
  - *if X^{λ} has no outgoing edges, create an edge from X^{λ} to λ_X labeled with $\lambda(X)$.*

The Petri net $\mathcal{PB} = (S, T, F, L)$ *is called the* high-level probability propagation net representing the Bayesian network $\mathcal{B}$.

Remark 4.7
The transformation rules are given for Bayesian networks, which are singly connected, that means which are free of loops. If a loopy Bayesian network is to be transformed into a probability propagation net, one has to eliminate the loops in the Bayesian network first – for example by conditioning (see section 2.3.3) – and then apply the transformation rules. ◇

A third popular example shows a combination of a split and a join illustrating the composition of single probability propagation nets as defined above.

Example 4.7 (Wet Grass, cf. [Jensen96])
This is a version of the popular "wet grass example" (see for example [Jensen96]). Mr. Holmes and Dr. Watson are living in LA. One morning, Holmes realizes that his grass is wet. He thinks that there are two explanations: it is due to rain (R) or he has forgotten to turn off the sprinkler (S). Next, he notices that the grass of his neighbor, Dr. Watson, is also wet. Of course, now Holmes is almost certain that it has rained.

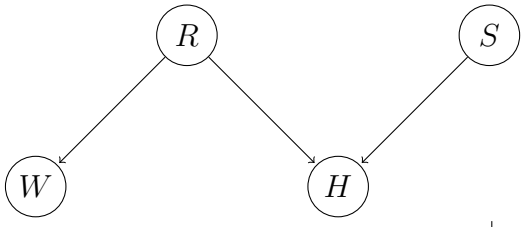
The BN $\mathcal{B}$ describing the initial situation is shown in Figure 4.8.

$h_1 =$	Mr. Holmes' grass is wet
$h_0 =$	Mr. Holmes' grass is not wet
$r_1 =$	rain is the cause for Mr. Holmes' grass being wet
$r_0 =$	rain is not the cause for Mr. Holmes' grass being wet
$s_1 =$	sprinkler is the cause for Mr. Holmes' grass being wet
$s_0 =$	sprinkler is not the cause for Mr. Holmes' grass being wet
$w_1 =$	Dr. Watson's grass is wet
$w_0 =$	Dr. Watson's grass is not wet

Table 4.5.: Random Variables of Example 4.7

The random variables H, R, S, W have two attributes (1 and 0, "yes" and "no") whose meanings and probabilities are given in Table 4.5 and Figure 4.8, respectively. The Bayesian network $\mathcal{B}$ consists of two basic structures:

$$P(R) = \begin{array}{c|cc} R & 1 & 0 \\ \hline & 0.2 & 0.8 \end{array} \qquad P(S) = \begin{array}{c|cc} S & 1 & 0 \\ \hline & 0.1 & 0.9 \end{array}$$



$$P(W \mid R) = \begin{array}{c|cc} & W & \\ \hline R & 1 & 0 \\ \hline 1 & 1.0 & 0.0 \\ 0 & 0.2 & 0.8 \end{array} \qquad P(H \mid RS) = \begin{array}{cc|cc} & & H & \\ \hline R & S & 1 & 0 \\ \hline 1 & 1 & 1.0 & 0.0 \\ 1 & 0 & 1.0 & 0.0 \\ 0 & 1 & 0.9 & 0.1 \\ 0 & 0 & 0.0 & 1.0 \end{array}$$

Figure 4.8.: Bayesian Network $\mathcal{B}$ of Example 4.7

- the split $R \rightarrow \{W, H\}$,
- and the join $\{R, S\} \rightarrow H$.

Figure 4.9 shows the probability propagation net representation of the split, Figure 4.10 the one of the join.

Obviously, the transformation of the join leads to edges between $\{R^\pi, R^\lambda\}$ and $\{f_H^0, f_H^1, f_H^2\}$, whereas the split transformation leads to edges between $\{R_H^\pi, R_H^\lambda\}$ and $\{f_H^0, f_H^1\}$. When composing the two probability propagation net representations, the additional constraint of Definition 4.8 has to be considered. Thus, only the edges related to R_H^π and R_H^λ are used for the high-level probability propagation net, whereas the edges from R^π to f_H^0 and f_H^2 and from f_H^1 to R^λ are omitted. The result is the high-level probability propagation net $\mathcal{PB}$, which is shown in Figure 4.11 and which represents the Bayesian network $\mathcal{B}$ (see Figure 4.8). □

Compared to the graphical representation of Bayesian networks, probability propagation nets may seem considerably complex. The number of nodes and edges is greater than in the corresponding Bayesian network. At first glance, this may be irritating but there are some advantages of the Petri net representation which will be

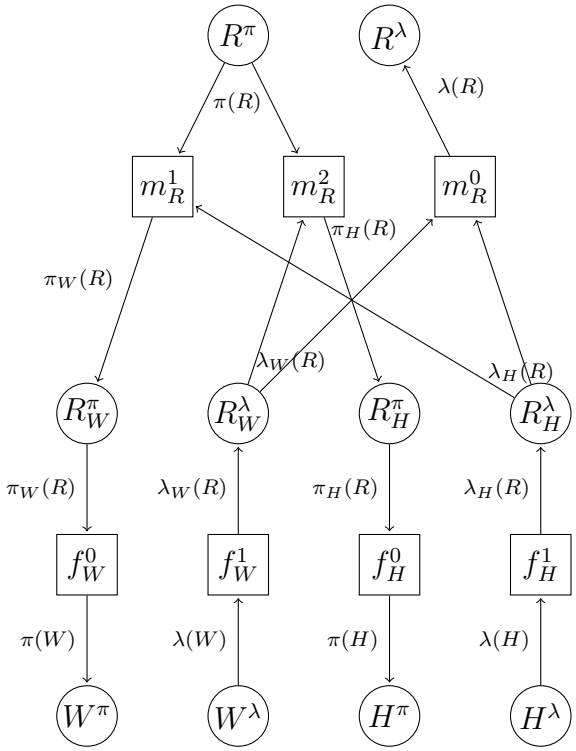


Figure 4.9.: Probability Propagation Net for the Split of Example 4.7

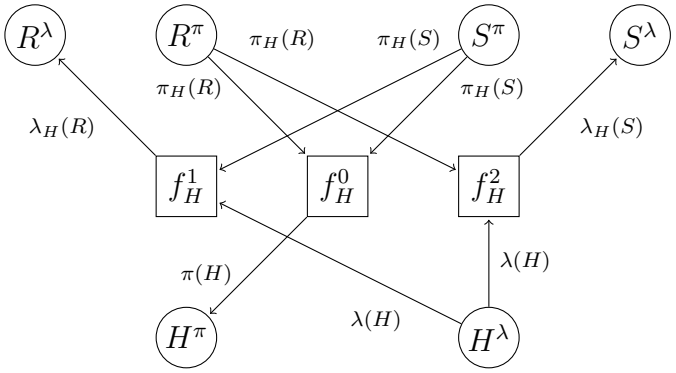


Figure 4.10.: Probability Propagation Net for the Join of Example 4.7

demonstrated in the next sections. Two of them shall be mentioned here:

- First, the more complex Petri net structure is due to the propagation algorithms of Bayesian networks, which are not at all represented in the Bayesian network itself. For some applications it can be important to have network and propagations combined in one representation. Additionally, this eases the un-

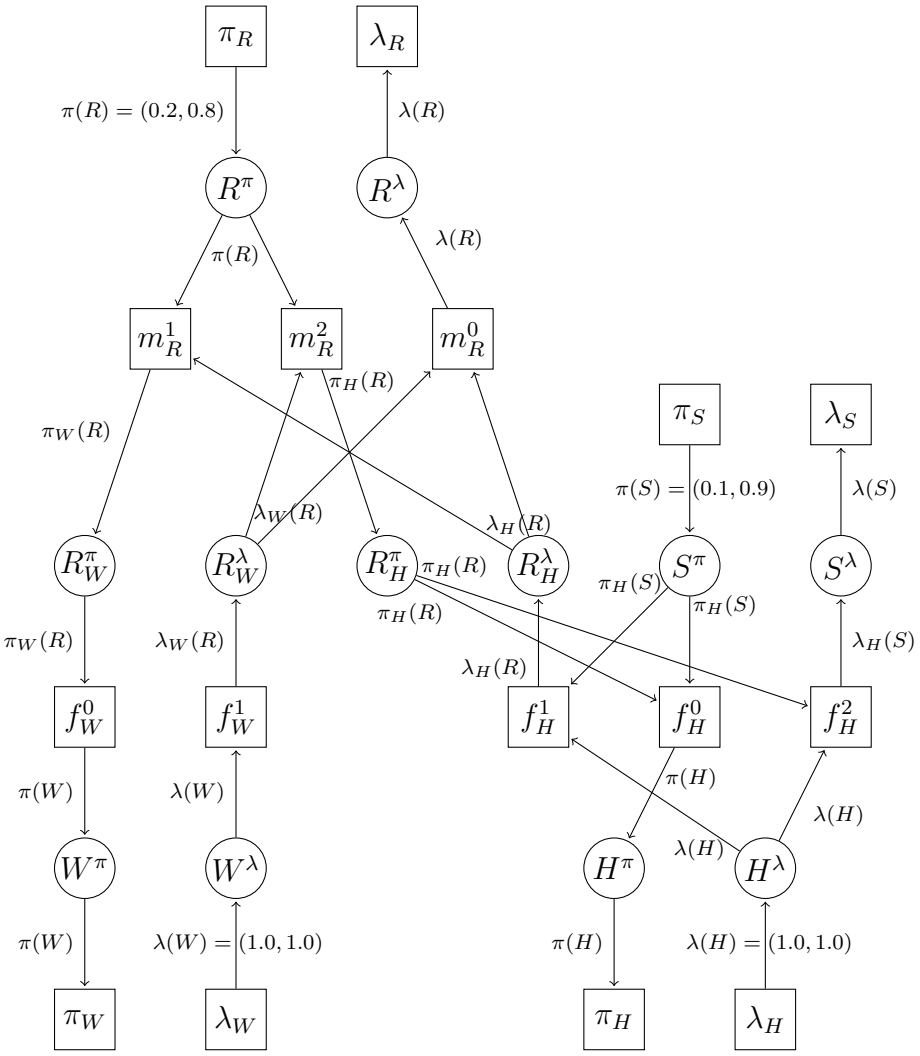


Figure 4.11.: Probability Propagation Net $\mathcal{PB}$ of Example 4.7

derstanding of how probabilistic processes in Bayesian networks work and how the random variables influence each other during propagation.

- Second, the marking concept of Petri nets allow for a distributed state representation which is very close to the concept of situation in Bayesian networks, that means the marking and arc labels completely represent the current state of the modeled system in a quite descriptive way.

The next sections show how message propagation of Bayesian networks can be reproduced by means of simulation in high-level probability propagation nets. In the

first part, the initialization process will be explained. After that, it will be described how evidences or instantiations (for example caused by certain observations in the modeled system) influence the beliefs and how they are propagated.

4.2. Initialization

The t-invariants of high-level probability propagation nets as solutions of a homogeneous linear equation system considering the p/t-net level result in net representations in which markings are reproducible. The Petri nets are cycle-free and have a transition boundary. This implies the reproducibility of the empty marking by every t-invariant as a flow of tuples from input to output boundary. Fortunately, these flows describe exactly the flow of λ- and π-messages. Thus, the net representations of t-invariants are a framework for the propagation of λ- and π-tuples.

The invariants of the "black" net (that means the ones of the underlying p/t-net) suffice to determine the propagation paths because only the graph structure is of interest. The arc labels modifying propagated tuples have no influence on propagation paths.

Hence, for message propagation in probability propagation nets, t-invariants are the basic concept, because every propagation path in a Bayesian network corresponds to a t-invariant in the Petri net representation. To initialize a probability propagation net, all λ- and π-labels connected with the appropriate boundary transitions have to be set to their initial (a priori) values. Then, all of the t-invariants are simulated one by one, starting with and reproducing the empty marking. The tuples being removed by the output boundary transitions indicate the current λ- and π-values for the respective variables. These values have to be stored, because with these values the initial beliefs for the corresponding variables are calculated according to the formula given in Definition 2.22 ($\lambda(X)$ and $\pi(X)$ are the current λ- and π-values, which have been calculated by the initialization process described above).

Remark 4.8
From now on in the context of probability propagation nets, *simulation of a t-invariant* means the reproduction of the empty marking in the net representation of the respective minimal t-invariant of the probability propagation net. The term is used as an abbreviation. ◇

The initialization process will now be demonstrated by continuing the Examples 4.1, 4.2 and 4.7.

Example 4.8 (Burglar Alarm: Initialization)
The high-level probability propagation net of the burglar alarm example is shown in Figure 4.5, the corresponding transition functions are given in Table 4.3. Due to lack of space, the rules for calculating $\pi(A), \lambda_A(B), \lambda_A(C)$ are missing in Figure 4.5 but they are specified in the t-invariants (Figure 4.12 to Figure 4.14).

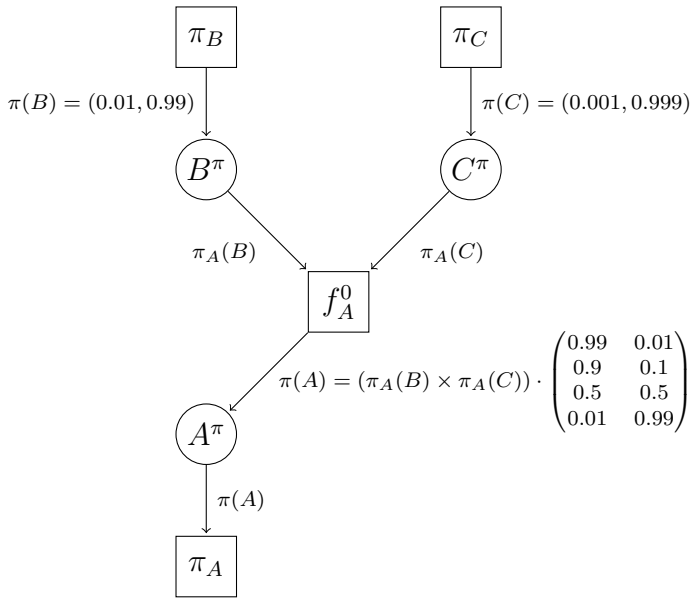


Figure 4.12.: $\pi(A)$-t-invariant Example 4.8

To open the initialization phase, all λ-tuples are set to $(1.0, 1.0)$. This states that there is no evidence to change the prior probabilities of B and C (initialization rule A, see section 2.3.2). Moreover, $\pi(B) = P(B) = (0.01, 0.99)$ and $\pi(C) = P(C) = (0.001, 0.999)$ are set (initialization rule B). Next, $\pi(A)$ is calculated by reproducing the empty marking in the $\pi(A)$-t-invariant of Figure 4.12. When firing, π_B and π_C put tokens $(0.01, 0.99)$ and $(0.001, 0.999)$ on the places B^π and C^π, respectively.

Then f_A^0 is activated and fires. By that, the tuples are removed from places B^π

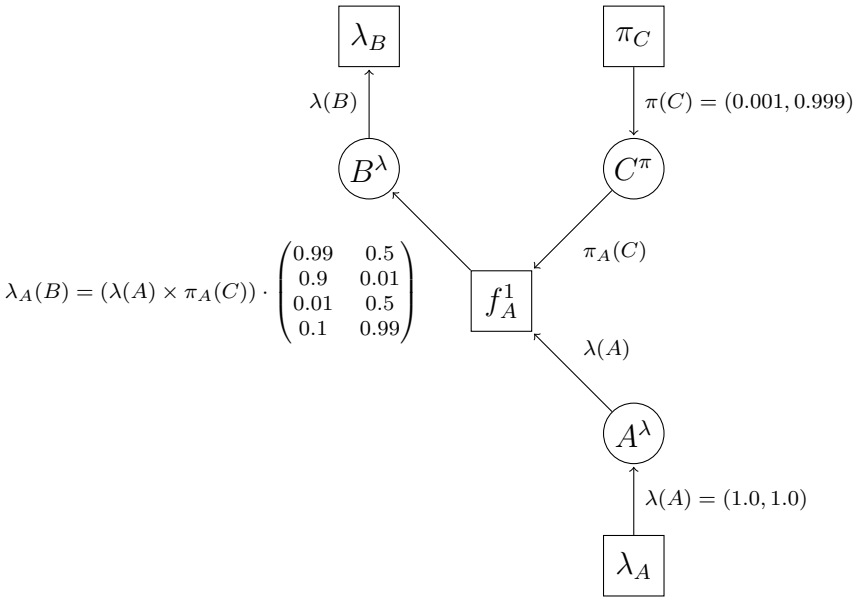


Figure 4.13.: $\lambda(B)$-t-invariant of Example 4.8

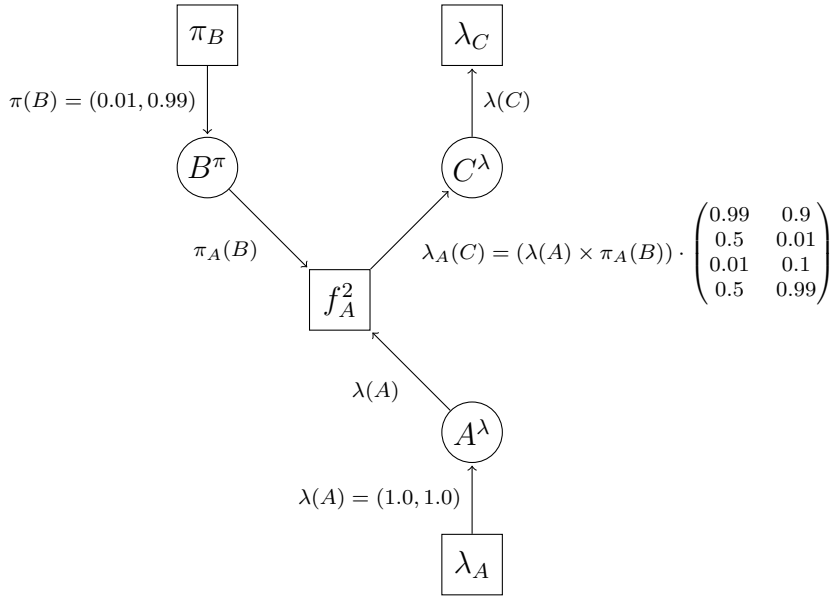


Figure 4.14.: $\lambda(C)$-t-invariant of Example 4.8

and C^π and the following tuple is put on place A^π (operative formula 4):

$$\pi(A) = (\pi_A(B) \times \pi_A(C)) \cdot \begin{pmatrix} 0.99 & 0.01 \\ 0.9 & 0.1 \\ 0.5 & 0.5 \\ 0.01 & 0.99 \end{pmatrix}$$

$$= ((0.01, 0.99) \times (0.001, 0.999)) \cdot \begin{pmatrix} 0.99 & 0.01 \\ 0.9 & 0.1 \\ 0.5 & 0.5 \\ 0.01 & 0.99 \end{pmatrix}$$

$$= (0.019, 0.982).$$

So,

$$\begin{aligned} BEL(A) &= \alpha \cdot (\lambda(A) \circ \pi(A)) \\ &= \alpha\,((1.0, 1.0) \circ (0.019, 0.981)) \\ &= (0.019, 0.982) \text{ for } \alpha = 1 \end{aligned}$$

Firing of π_A completes the reproduction of the empty marking. □

Remark 4.9
As all λ-values are set to the neutral element $(1.0, 1.0)$, the simulation of the two λ-invariants (Figure 4.13 and 4.14) does not change any beliefs. Because of that, the simulation of these invariants is not shown in detail. The λ-invariants are important when evidence is brought into the system. This will be described later by continuing this example and executing the simulation of the $\lambda(B)$-invariant step by step (see Example 4.11). ◇

Now, that all t-invariants have been simulated, the net is initialized. In contrast to the above example, when initializing the probability propagation net of the cheating spouse example (see Figure 4.7), λ-values have to be considered as well, because they are part of the π-invariants. This is shown in the following example.

Example 4.9 (Cheating Spouse: Initialization)
The net representations of the t-invariants of the high-level probability propagation net modeling the cheating spouse example (see Figure 4.7) are shown in Figure 4.15, 4.16 and 4.17. The functions belonging to the respective transitions are given in Table 4.4. Due to lack of space, the rules for calculating the output tuples of the transitions are not listed here but they are specified in the net representations of all t-invariants. All three minimal t-invariants in vector form are given in Table 4.6.

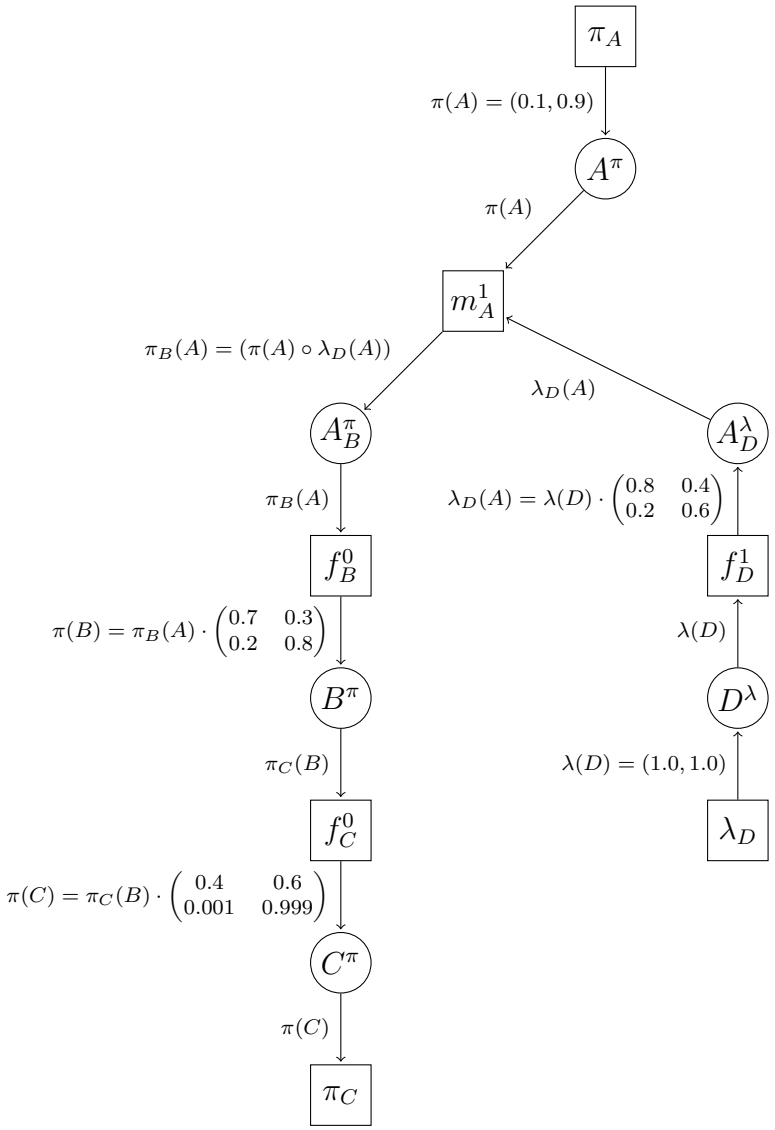


Figure 4.15.: $\pi(C)$-t-invariant of Example 4.9

	π_A	λ_A	π_C	λ_C	π_D	λ_D	f_B^0	f_B^1	f_C^0	f_C^1	f_D^0	f_D^1	m_A^0	m_A^1	m_A^2
$\lambda(A)$		1		1		1		1		1		1	1		
$\pi(C)$	1		1			1	1		1			1		1	
$\pi(D)$	1			1	1			1		1	1				1

Table 4.6.: T-invariants of $\mathcal{PB}$ (see Example 4.2)

In the initialization phase, first all λ-tuples are set to $(1.0, 1.0)$ (initialization rule A), thus expressing that no external evidence concerning the random variables is known. Next, by setting $\pi(A) = P(A) = (0.1, 0.9)$ (initialization rule B), the propa-

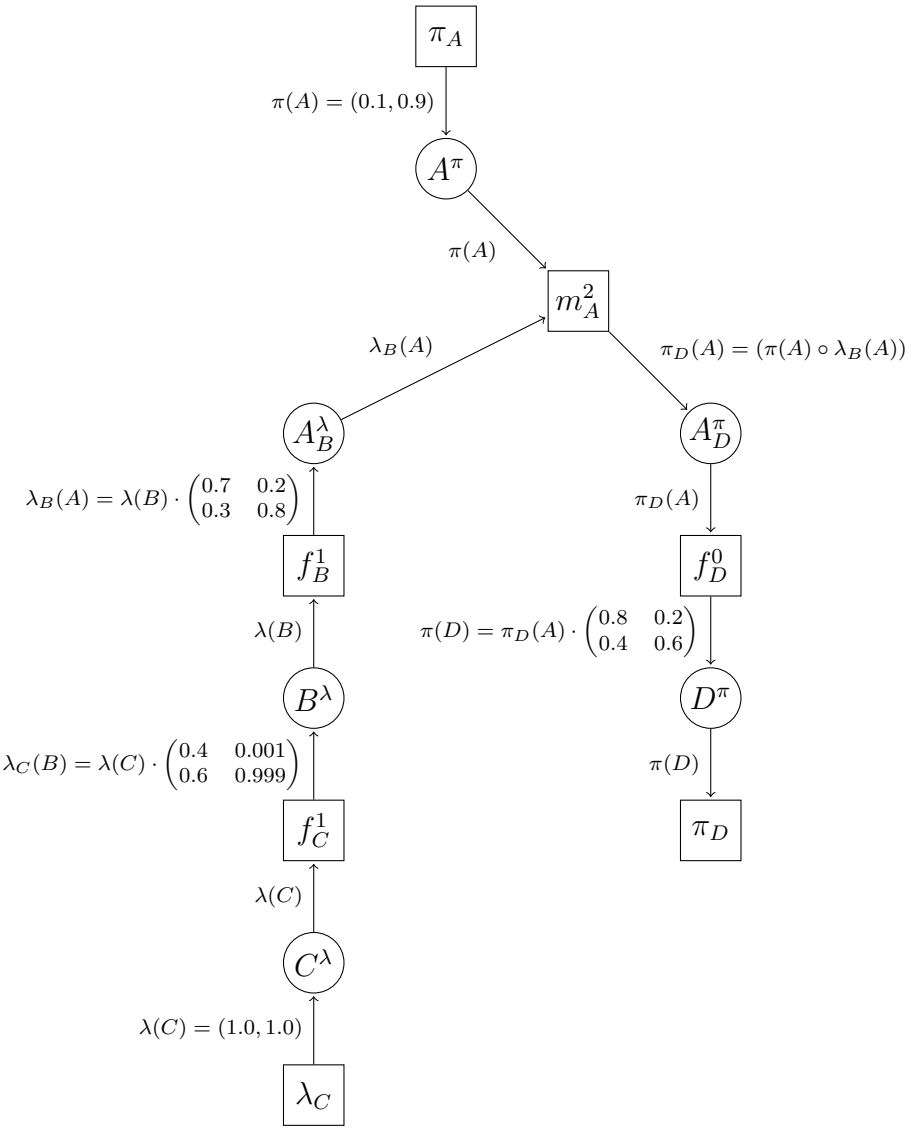


Figure 4.16.: $\pi(D)$-t-invariant of Example 4.9

gation of $P(a_1) = 0.1, P(a_2) = 0.9$ as a π-message is prepared. Finally, $BEL(C)$ and $BEL(D)$ have to be calculated. Thus the $\pi(C)$- and $\pi(D)$-t-invariant (Figure 4.15 and 4.16) are used. In Figure 4.15, the boundary transitions π_A and λ_D are enabled because they have no input places. When firing, π_A puts the tuple $\pi(A) = (0.1, 0.9)$ on place A^π (initialization rule B), λ_D puts $\lambda(D) = (1.0, 1.0)$ on place D^λ (initialization rule A). Now, transition f_D^1 is enabled and removes $(1.0, 1.0)$ from place D^λ and puts $\lambda_D(A) = \lambda(D) \cdot \left(\begin{smallmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{smallmatrix}\right) = (1.0, 1.0)$ on place A_D^λ (operative formulas 1 and 3).

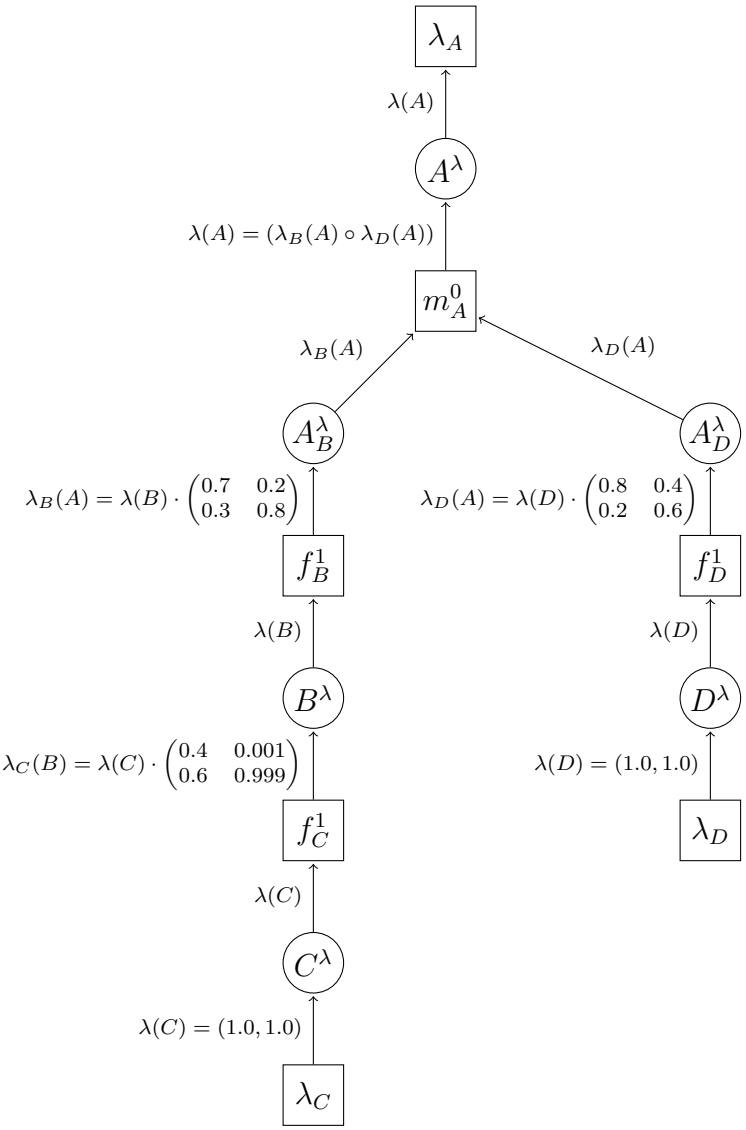


Figure 4.17.: $\lambda(A)$-t-invariant of Example 4.9

(The boundary transitions π_A and λ_D are permanently enabled. But for reproducing the empty marking they need to fire only once. See Table 4.6 where their entries are 1 in the corresponding row.)

The next transition to fire is m_A^1. It removes $\pi(A) = (0.1, 0.9)$ and $\lambda_D(A) = (1.0, 1.0)$ from the respective places A^π and A_D^λ and puts $\pi_B(A) = \pi(A) \circ \lambda_D(A) = (0.1, 0.9)$ on place A_B^π (Definition 2.20).

When a tuple is removed from a place X, that does not mean that the values of

the tuple are no longer valid for X. It simply says that they have been used and that they can be re-generated any time.

Next, the transitions f_B^0, f_C^0, π_C are sequentially enabled and fire in that order. By firing of f_B^0 the tuple $(0.1, 0.9)$ is removed from A_B^π, and the tuple $\pi(B) = (0.1, 0.9) \cdot \left(\begin{smallmatrix} 0.7 & 0.3 \\ 0.2 & 0.8 \end{smallmatrix}\right) = (0.25, 0.75)$ is put on B^π (operative formula 4). This tuple is removed by firing of f_C^0 and the tuple $\pi(C) = (0.25, 0.75) \cdot \left(\begin{smallmatrix} 0.4 & 0.6 \\ 0.001 & 0.999 \end{smallmatrix}\right) = (0.10075, 0.89925)$ is put on C^π (operative formula 4) from where it is removed by transition π_C, thus completing the reproduction of the empty marking.

The beliefs $BEL(B)$ and $BEL(C)$ are then calculated as follows:

$$\begin{aligned} BEL(B) &= \alpha\left(\lambda(B) \circ \pi(B)\right) = \alpha\left((1.0, 1.0) \circ (0.25, 0.75)\right) \\ &= (0.25, 0.75) \\ BEL(C) &= \alpha\left(\lambda(C) \circ \pi(C)\right) \\ &= \alpha\left((1.0, 1.0) \circ (0.10075, 0.89925)\right) \\ &= (0.10075, 0.89925) \text{ (operative formula 5).} \end{aligned}$$

Similarly $BEL(D) = (0.44, 0.56)$ is calculated on the basis of the $\pi(D)$-invariant (Figure 4.16). Again, the simulation of the λ-invariant does not influence any beliefs.

In conclusion the initialization of the probability propagation net results in the beliefs

$$\begin{aligned} BEL(A) &= (0.1, 0.9) \\ BEL(B) &= (0.25, 0.75) \\ BEL(C) &= (0.10075, 0.89925) \\ BEL(D) &= (0.44, 0.56) \end{aligned}$$

which are in accordance with the beliefs calculated by the propagation rules for Bayesian networks as shown in [Neap90]. □

For the sake of completeness, the initialization process for the wet grass example will now be sketched as well.

Example 4.10 (Wet Grass: Initialization)
The probability propagation net of this example is shown in Figure 4.11. Table 4.7 shows the functions belonging to the respective transitions.

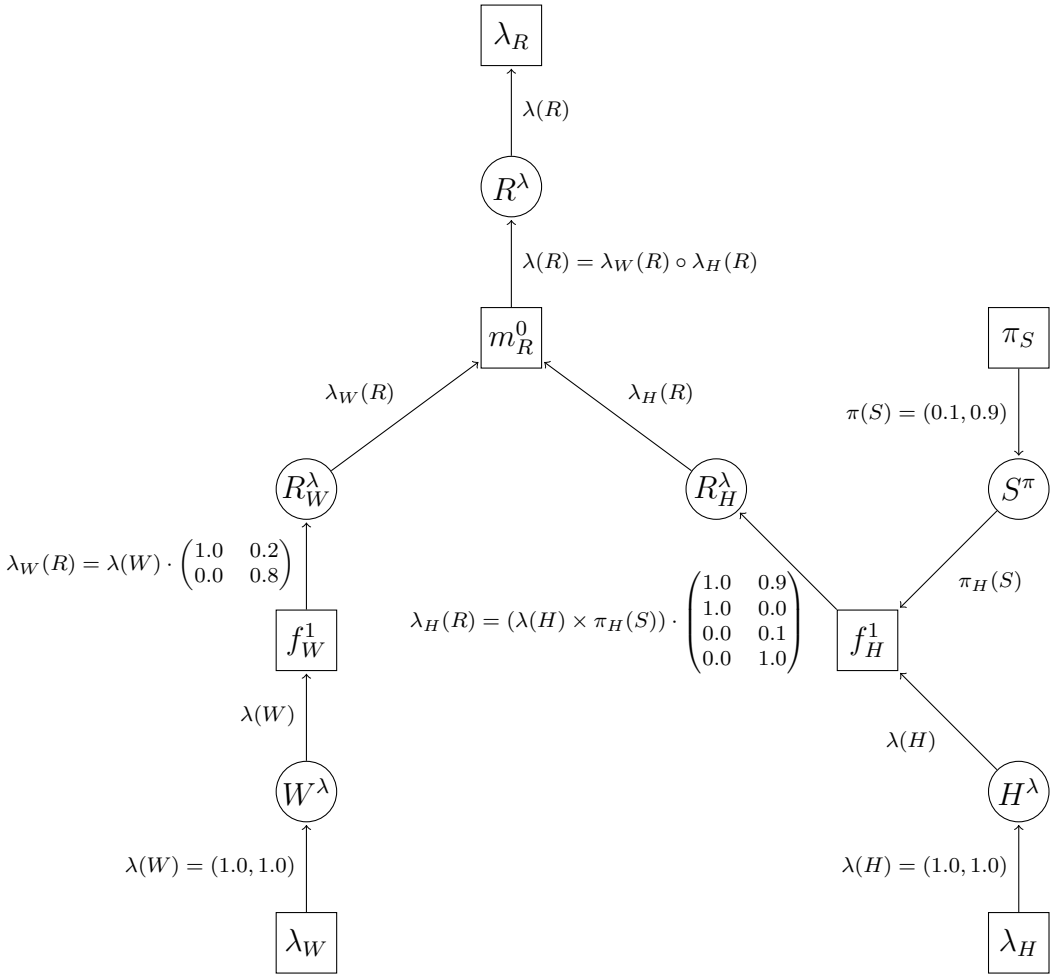


Figure 4.18.: $\lambda(R)$-Invariant of Example 4.7

To start the initialization, the propagations within the net representations of the four minimal t-invariants of the probability propagation net $\mathcal{PB}$ have to be executed. The invariants are shown in Figure 4.18 to 4.21.

The initialization starts with the $\pi(W)$-t-invariant (Figure 4.19). Firing the boundary transitions results in tuples $(0.2, 0.8), (0.1, 0.9), (1.0, 1.0)$ on places R^π, S^π, H^λ, respectively. This indicates that the probabilities of $R, \neg R, S, \neg S$ are $0.2, 0.8, 0.1, 0.9$, respectively, and that there is no evidence to correct the probabilities of H and $\neg H$ (in case they are known or going to be calculated).

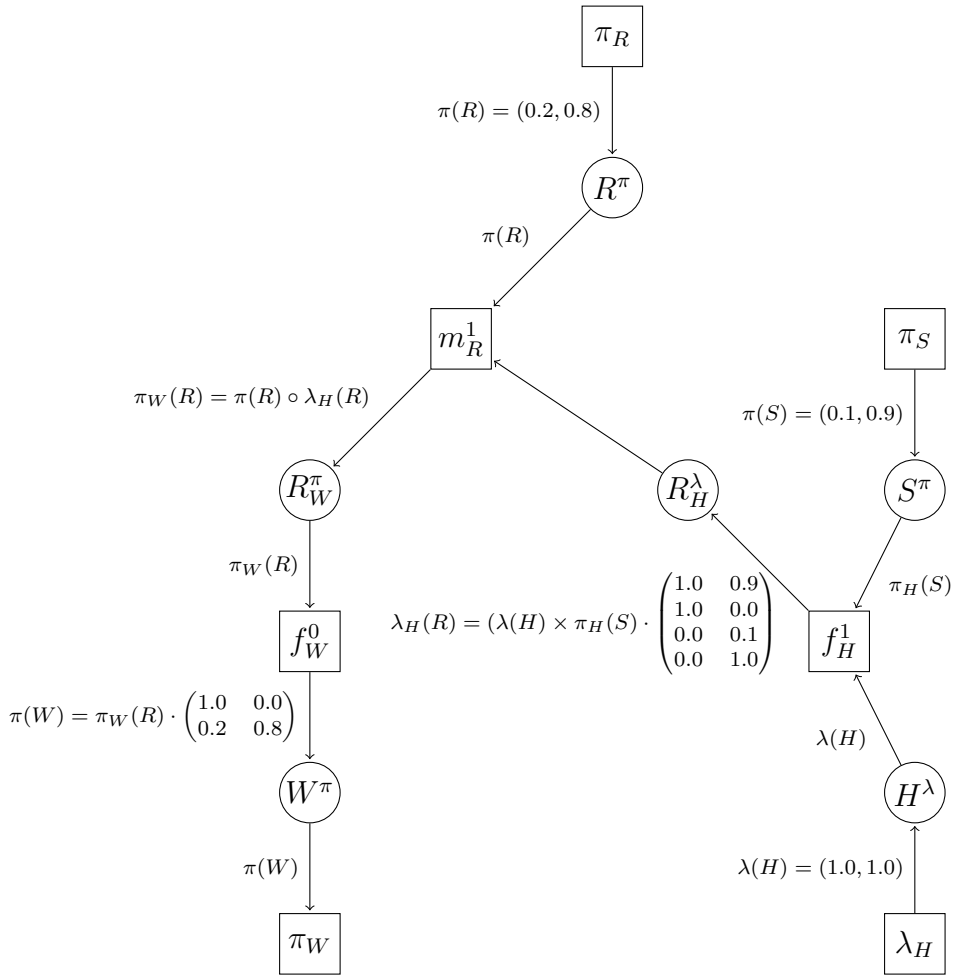


Figure 4.19.: $\pi(W)$-Invariant of Example 4.7

Next, transition f_H^1 fires as follows: The tuples are removed from S^π and H^λ.

$$\lambda_H(R) = (\lambda(H) \times \pi_H(S)) \cdot CPM_H^1 = ((1.0, 1.0) \times (0.1, 0.9)) \cdot CPM_H^1$$
$$= (0.1, 0.9, 0.1, 0.9) \cdot \begin{pmatrix} 1.0 & 0.9 \\ 1.0 & 0.0 \\ 0.0 & 0.1 \\ 0.0 & 1.0 \end{pmatrix} = (1.0, 1.0)$$

is put on place R_H^λ.

Now, transition m_R^1 fires as follows: Tuples $(0.2, 0.8)$ and $(1.0, 1.0)$ are removed

$\pi_R \simeq P(R) =$

	R	
	1	0
	0.2	0.8

$\pi_S \simeq P(S) =$

	S	
	1	0
	0.1	0.9

$f_W^0 \simeq P(W \mid R) =$

	W	
R	1	0
1	1.0	0.0
0	0.2	0.8

$f_W^1 \simeq P^{W \leftarrow R}(W \mid R) =$

	R	
W	1	0
1	1.0	0.2
0	0.0	0.8

$f_H^0 \simeq P(H \mid RS)$

		H	
R	S	1	0
1	1	1.0	0.0
1	0	1.0	0.0
0	1	0.9	0.1
0	0	0.0	1.0

$f_H^1 \simeq P^{R \leftarrow HS}(H \mid RS)$

		R	
H	S	1	0
1	1	1.0	0.9
1	0	1.0	0.0
0	1	0.0	0.1
0	0	0.0	1.0

$f_H^2 \simeq P^{S \leftarrow HR}(H \mid RS)$

		S	
H	R	1	0
1	1	1.0	1.0
1	0	0.9	0.0
0	1	0.0	0.0
0	0	0.1	1.0

Table 4.7.: Transition Functions of Example 4.7

from their respective places R^π and R_H^λ. Then

$$\pi_W(R) = \pi(R) \circ \lambda_H(R) = (0.2, 0.8) \circ (1.0, 1.0) = (0.2, 0.8)$$

is put on the output place R_W^π (see Definition 2.7).

Lastly, f_W^0 fires by taking $(0.2, 0.8)$ from R_W^π and putting

$$\pi(W) = \pi_W(R) \cdot CPM_W^0 = (0.2, 0.8) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.2 & 0.8 \end{pmatrix} = (0.36, 0.64)$$

on place W^π. This tuple is removed in the end, thereby completing the reproduction of the empty marking.

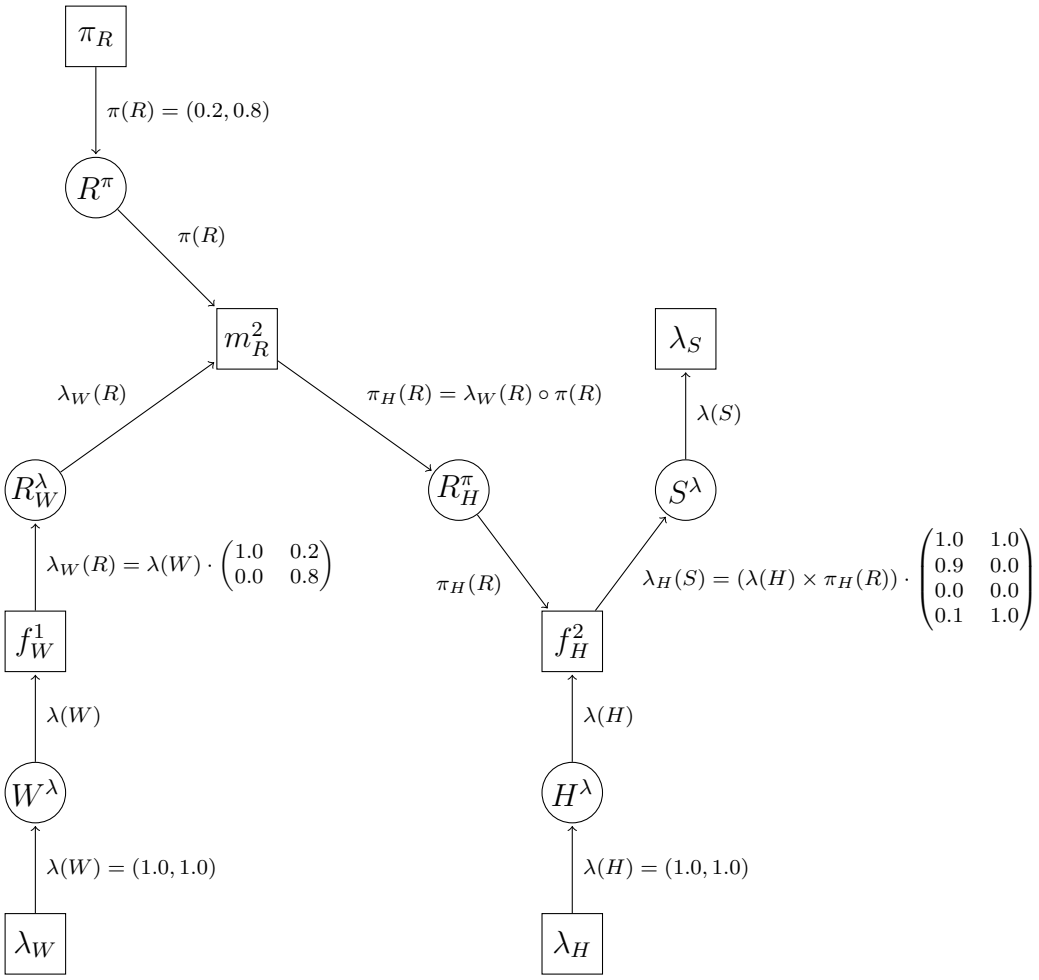


Figure 4.20.: $\lambda(S)$-Invariant of Example 4.7

Altogether,

$$\begin{aligned} BEL(R) &= (0.2, 0.8) \\ BEL(S) &= (0.1, 0.9) \\ BEL(W) &= (0.36, 0.64) \\ BEL(H) &= (0.272, 0.728) \end{aligned}$$

are the beliefs for R, S, W and H after the initialization process. All probabilities and the corresponding beliefs coincide, because all the likelihoods turned out to be $(1.0, 1.0)$. □

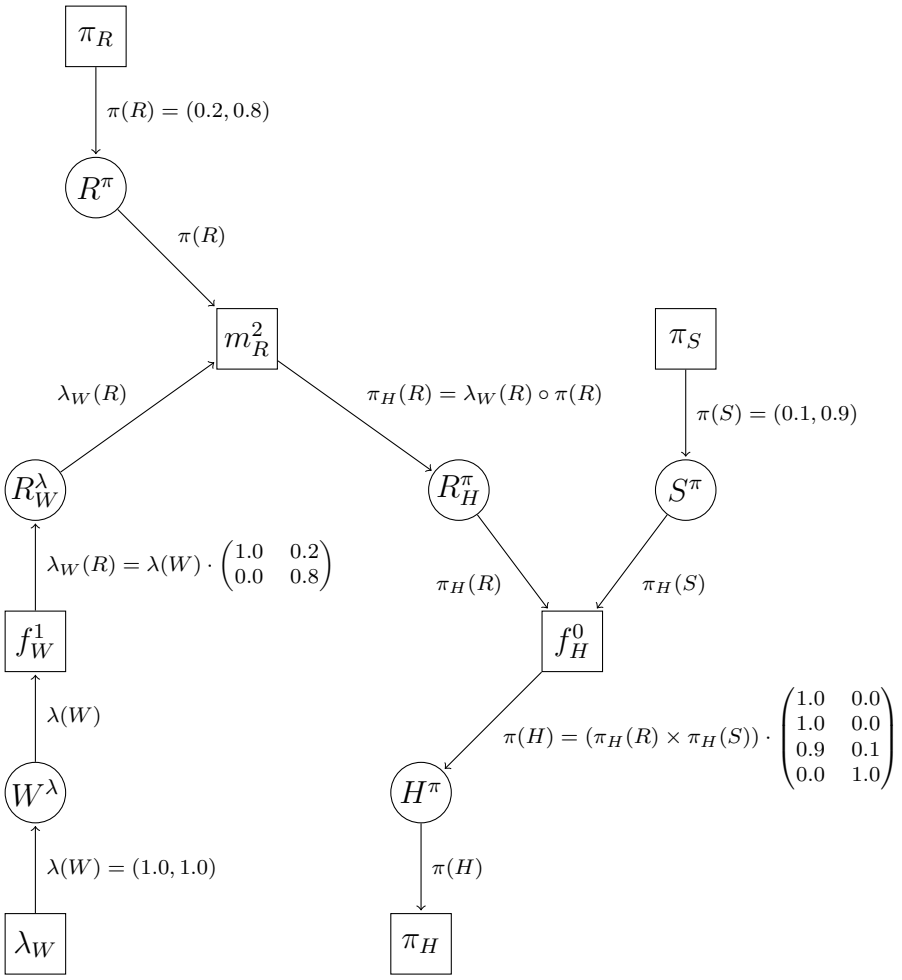


Figure 4.21.: $\pi(H)$-Invariant of Example 4.7

In conclusion it can be pointed out that initializing the probability propagation net is a quite simple task. Simulating every t-invariant suffices to get the initialized values for the variables. If all likelihoods are set to neutral values, even simulating only the π-invariants suffices. Considering that the firing rule is not that complex and that it seems to be straight forward understandable for people who are familiar with Petri nets, the probability propagation net representation of Bayesian networks is a remarkable approach to reveal the complex propagation structure which is hidden in the Bayesian network algorithms. So probability propagation nets make the world of Bayesian networks more transparent. Furthermore, the t-invariants provide means of structure by exactly representing the propagation paths in Bayesian networks and therefore structuring the Petri net in a natural and adequate way. In addition, the

marking concept and arc labels of probability propagation nets provide an appropriate representation of the current situation in the Bayesian network or in the modeled system, which is directly integrated in the Petri net.

The next section shows that also evidence propagation is done straight forward in probability propagation nets, again by simulating corresponding t-invariants.

4.3. Evidence Propagation

Evidence in the context of Bayesian networks means particular observations about the modeled system which influence the certainty of variable states. According to [Jensen96], if the exact state of a variable is known, this is called *hard evidence* or *instantiation of the variable*, if just the tendency for the states of a variable is adjusted, this is called *soft evidence*.

In Bayesian networks this is done by changing the λ- or π-values of the corresponding variables. If for example a variable $X = (x_1, x_2)$ is instantiated for x_1, both λ- and π-values are changed to $(1.0, 0.0)$. Obviously, in the Petri net representation this adjustment can be done by simply changing the corresponding labels of the edges connected to the preset nodes. In this case, the arcs $\{(t, X^\pi) \mid t \in {}^\bullet X^\pi\}$ and $\{(t, X^\lambda) \mid t \in {}^\bullet X^\lambda\}$ are set to $(1.0, 0.0)$. Generally, the new label is a vector whose entries are 0.0 except the entry corresponding to the instantiated value, which is 1.0.

To propagate the new evidence and update the beliefs only the associated t-invariants have to be simulated. This will now be demonstrated by continuing the above examples and bringing in evidence according to the original examples in the cited literature.

Example 4.11 (Burglar Alarm: Evidence Propagation)
Now, it is assumed that Mr. Holmes got the call and knows that his alarm sounds. That means A has to be instantiated for a_1. Consequently, the arc label $(1.0, 1.0)$ of arc (λ_A, A^λ) has to be changed to $(1.0, 0.0)$ in Figure 4.5 and Figure 4.12 to 4.14. In order to calculate Holmes' present beliefs about B (being burglarized) and C (earthquake), the empty marking is being reproduced in the adjusted $\lambda(B)$- and $\lambda(C)$-t-invariants (Figure 4.22 and 4.23).

In Figure 4.22, after firing λ_A and π_C, tuples $(1.0, 0.0)$ and $(0.001, 0.999)$ are lying on places A^λ and C^π, respectively. Now, f_A^1 is activated and fires thereby removing the

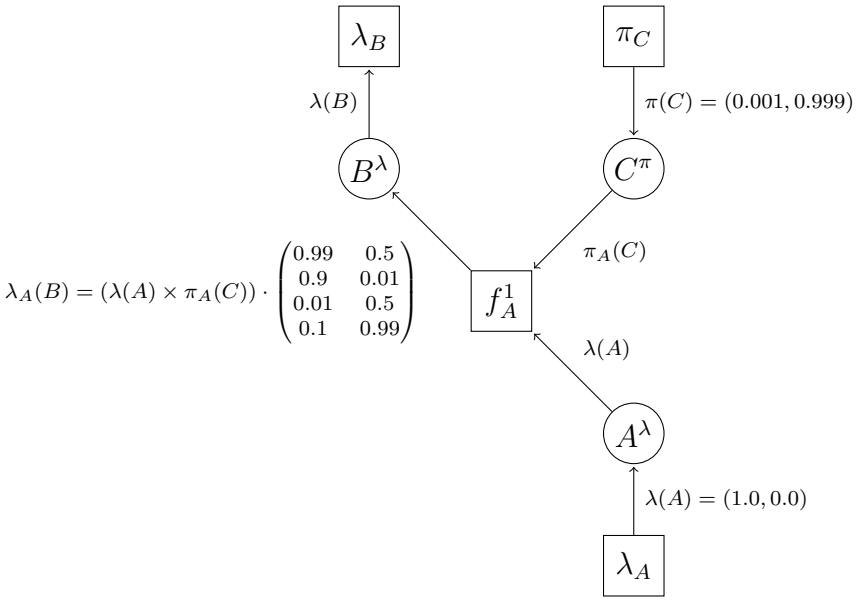


Figure 4.22.: Evidence Adjustment for $\lambda(B)$-t-invariant of Example 4.11

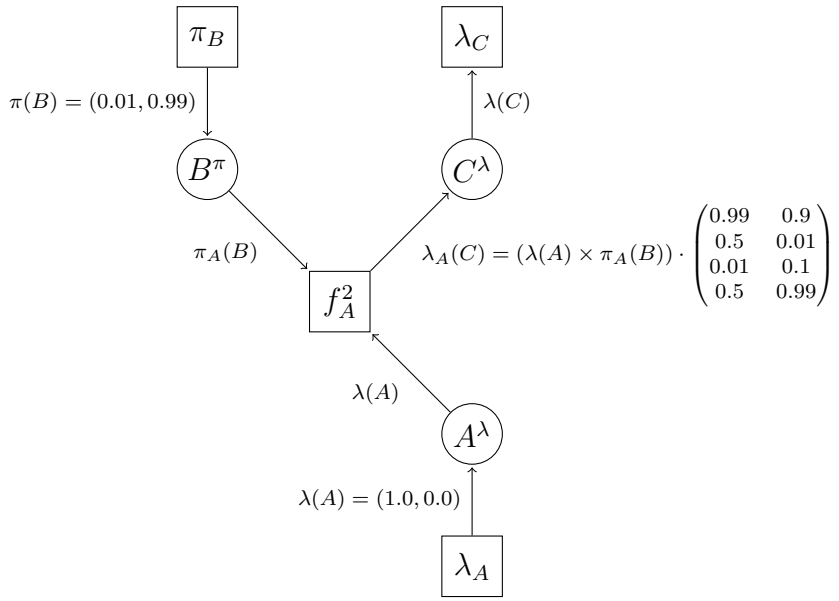


Figure 4.23.: Evidence Adjustment for $\lambda(C)$-t-invariant of Example 4.11

tuples from places A^λ and C^π, and the following tuple is put on place B^λ (operative

formula 1):

$$\begin{aligned}\lambda_A(B) &= (\lambda(A) \times \lambda_A(C)) \cdot \begin{pmatrix} 0.99 & 0.5 \\ 0.9 & 0.01 \\ 0.01 & 0.5 \\ 0.1 & 0.99 \end{pmatrix} \\ &= ((1.0, 0.0) \times (0.001, 0.999)) \cdot \begin{pmatrix} 0.99 & 0.5 \\ 0.9 & 0.01 \\ 0.01 & 0.5 \\ 0.1 & 0.99 \end{pmatrix} \\ &= (0.9, 0.01).\end{aligned}$$

Removing that tuple by λ_B completes the reproduction of the empty marking.

In the same way, the $\lambda(C)$-t-invariant shown in Figure 4.23 returns $\lambda_A(C) = (0.505, 0.019)$. This leads to

$$\begin{aligned}BEL(B) &= \alpha\left(\lambda(B) \circ \pi(B)\right) = \alpha\left((0.9, 0.01) \circ (0.01, 0.99)\right) \\ &= \alpha(0.009, 0.0099) = (0.476, 0.524) \text{ for } \alpha = \tfrac{1}{0.0189} \\ BEL(C) &= \alpha\left(\lambda(C) \circ \pi(C)\right) \\ &= \alpha\left((0.505, 0.019) \circ (0.001, 0.999)\right) \\ &= \alpha(0.000505, 0.018981) = (0.026, 0.974) \\ &\text{for } \alpha = \tfrac{1}{0.019486}\end{aligned}$$

Holmes' belief in being burglarized has increased from 0.01 to 0.476. Even his belief in an earthquake has increased from 0.001 to 0.026. Although compared to his belief in being burglarized the belief in an earthquake is quite small, it has increased by a factor of 26. Yet there is also the possibility of a false alarm which is not modeled here but which can be assumed.

Next, Mr. Homes is assumed to have heard the announcement of an earthquake on the radio. Now, the arc label $(0.001, 0.99)$ of arc (π_c, C^π) has to be substituted with $(1.0, 0.0)$, since C has to be instantiated for c_1. To calculate Mr. Holmes' belief about B (being burglarized), again, the empty marking in the $\lambda(B)$-t-invariant (Figure 4.24) has to be reproduced. After firing λ_A and π_C, tuples $(1.0, 0.0)$ are lying on places A^λ

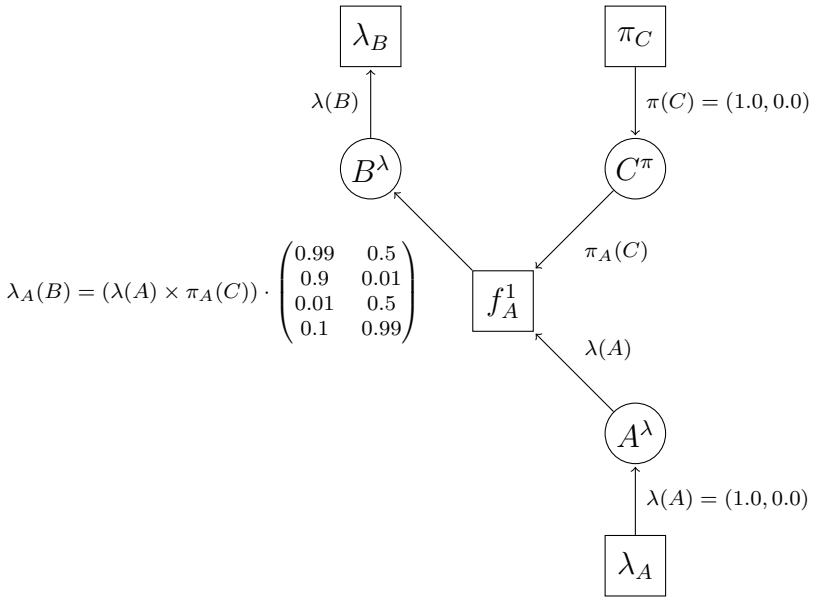


Figure 4.24.: Second Evidence for $\lambda(B)$-t-invariant of Example 4.11

and C^π, respectively. After firing f_A^1 these tuples are removed and the following tuple is put on place B^λ:

$$\begin{aligned}
\lambda_A(B) &= (\lambda(A) \times \pi_A(C)) \cdot \begin{pmatrix} 0.99 & 0.5 \\ 0.9 & 0.01 \\ 0.01 & 0.5 \\ 0.1 & 0.99 \end{pmatrix} \\
&= ((1.0, 0.0) \times (1.0, 0.0)) \cdot \begin{pmatrix} 0.99 & 0.5 \\ 0.9 & 0.01 \\ 0.01 & 0.5 \\ 0.1 & 0.99 \end{pmatrix} \\
&= (1.0, 0.0, 0.0, 0.0) \cdot \begin{pmatrix} 0.99 & 0.5 \\ 0.9 & 0.01 \\ 0.01 & 0.5 \\ 0.1 & 0.99 \end{pmatrix} \\
&= (0.99, 0.5).
\end{aligned}$$

λ_B fires and removes that tuple from B^λ, thus completing the reproduction of the

empty marking. Mr. Holmes' new belief concerning B is

$$\begin{aligned} BEL(B) &= \alpha\,(\lambda(B) \circ \pi(B)) = \alpha\,((0.99, 0.5) \circ (0.01, 0.99)) \\ &= \alpha(0.0099, 0.495) = (0.02, 0.98) \text{ for } \alpha = \tfrac{1}{0.5049}. \end{aligned}$$

So, Holmes' belief in being burglarized has changed from 0.01 via 0.476 to 0.02. Holmes was worried after the phone call and calmed down after he heard the announcement on the radio. □

The values coincide with the values calculated by Bayesian message propagation in [Neap90]. In the Petri net representation it is easy to trace the influences of evidence on other variables by watching the t-invariant simulation, which is yet another facet of transparency added to Bayesian network models by representing them as probability propagation nets.

To exemplify the propagation of evidence in Bayesian networks containing a split, Example 4.9 will now be continued.

Example 4.12 (Cheating Spouse: Evidence Propagation)
Now, spouse was seen dining with another. Hence, B is instantiated: $\lambda(B) = \pi(B) = (1.0, 0.0)$. The consequence for the corresponding Petri nets is quite simple: the variable arc labels $\lambda_C(B)$ and $\pi(B)$ are replaced by the constant tuple $(1.0, 0.0)$ (as an example see Figure 4.27). That means whatever the input values enabling f_B^0 and f_C^1 are, both transitions put $(1.0, 0.0)$ on their respective output places B (operative formulas 2, and 3).

The changes of the beliefs are as follows:

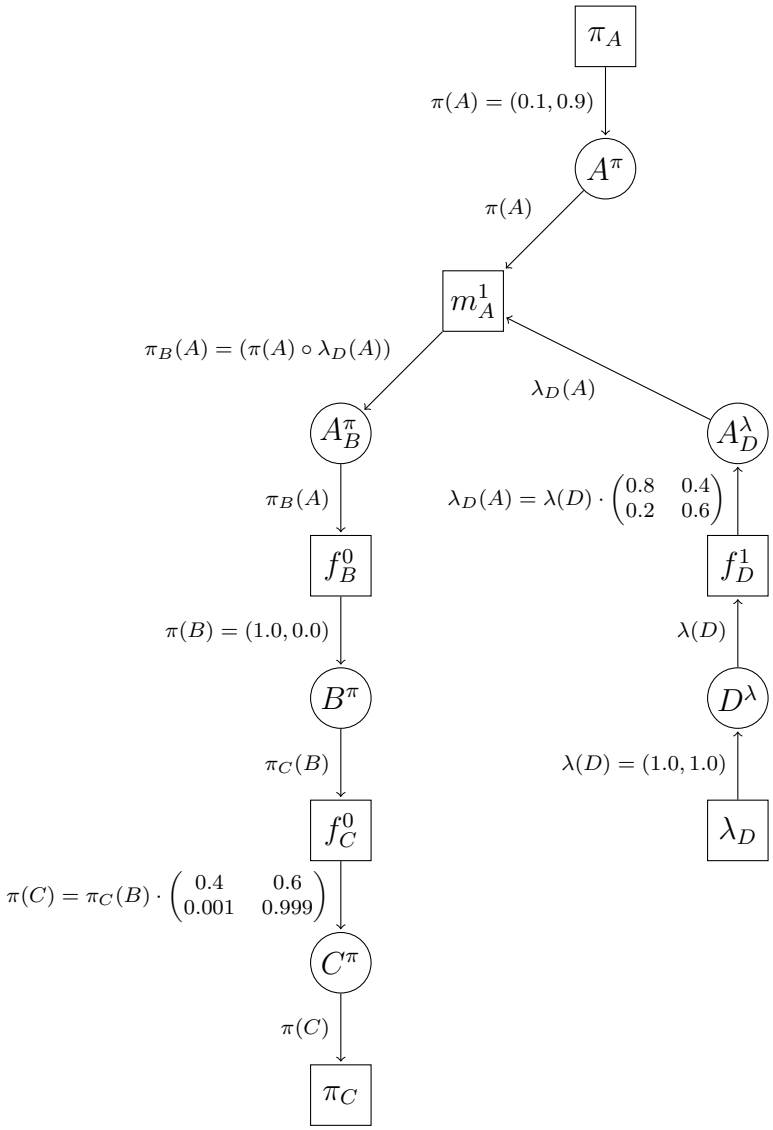


Figure 4.25.: Evidence Adjustment for $\pi(C)$-t-invariant Example 4.9

- For the adjusted $\lambda(A)$-t-invariant (see Figure 4.27):

$$\begin{aligned}
\lambda_D(A) &= (1.0, 1.0) \text{ as before;} \\
\lambda_B(A) &= \lambda(B) \cdot \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} = (1.0, 0.0) \cdot \begin{pmatrix} 0.7 & 0.2 \\ 0.3 & 0.8 \end{pmatrix} \\
&= (0.7, 0.2) \text{ by firing of } f_B^2; \\
\lambda(A) &= \lambda_B(A) \circ \lambda_D(A) = (0.7, 0.2) \circ (1.0, 1.0) \\
&= (0.7, 0.2).
\end{aligned}$$

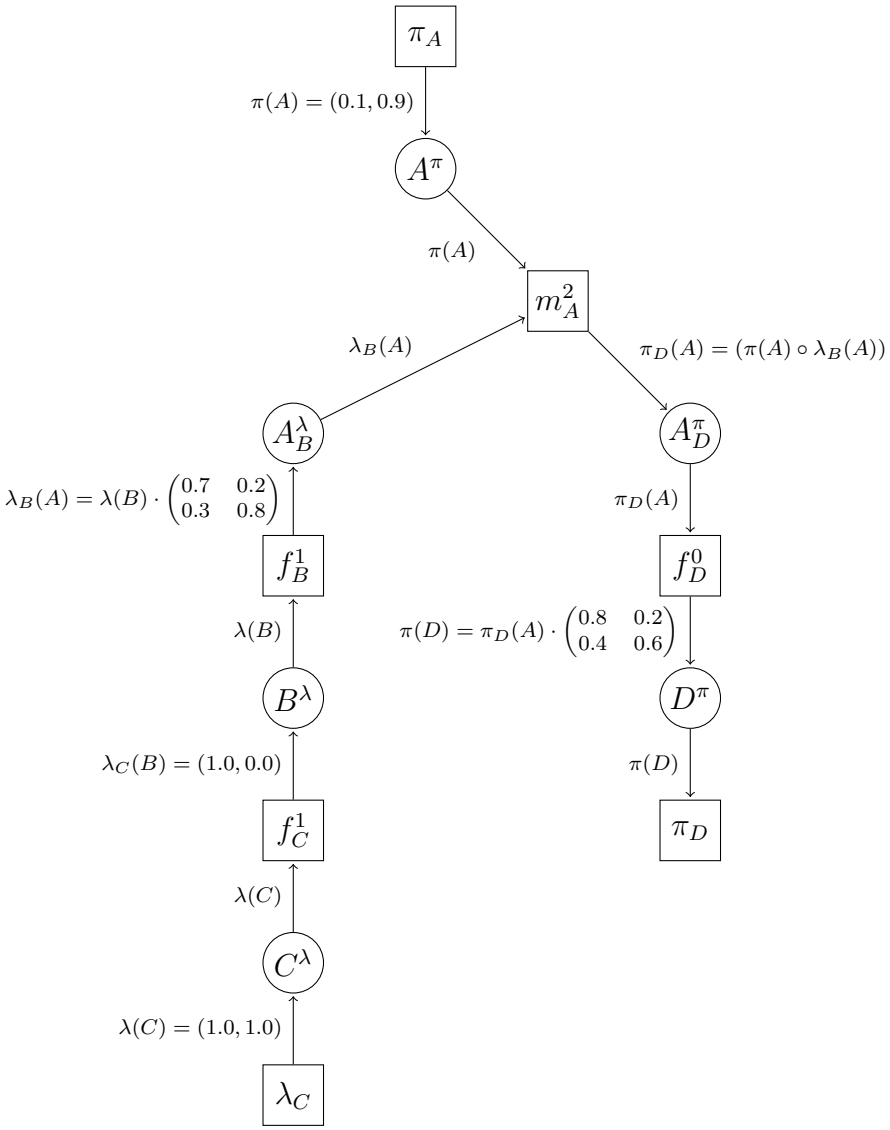


Figure 4.26.: Evidence Adjustment for $\pi(D)$-t-invariant Example 4.9

Therefore,

$$\begin{aligned} BEL(A) &= \alpha\left(\pi(A) \circ \lambda(A)\right) = \alpha\left((0.1, 0.9) \circ (0.7, 0.2)\right) \\ &= \alpha(0.07, 0.18) = (0.28, 0.72) \text{ for } \alpha = \tfrac{1}{0.25}. \end{aligned}$$

- For the adjusted $\pi(C)$-t-invariant (see Figure 4.25), f_B^0 puts the tuple (1.0,0.0)

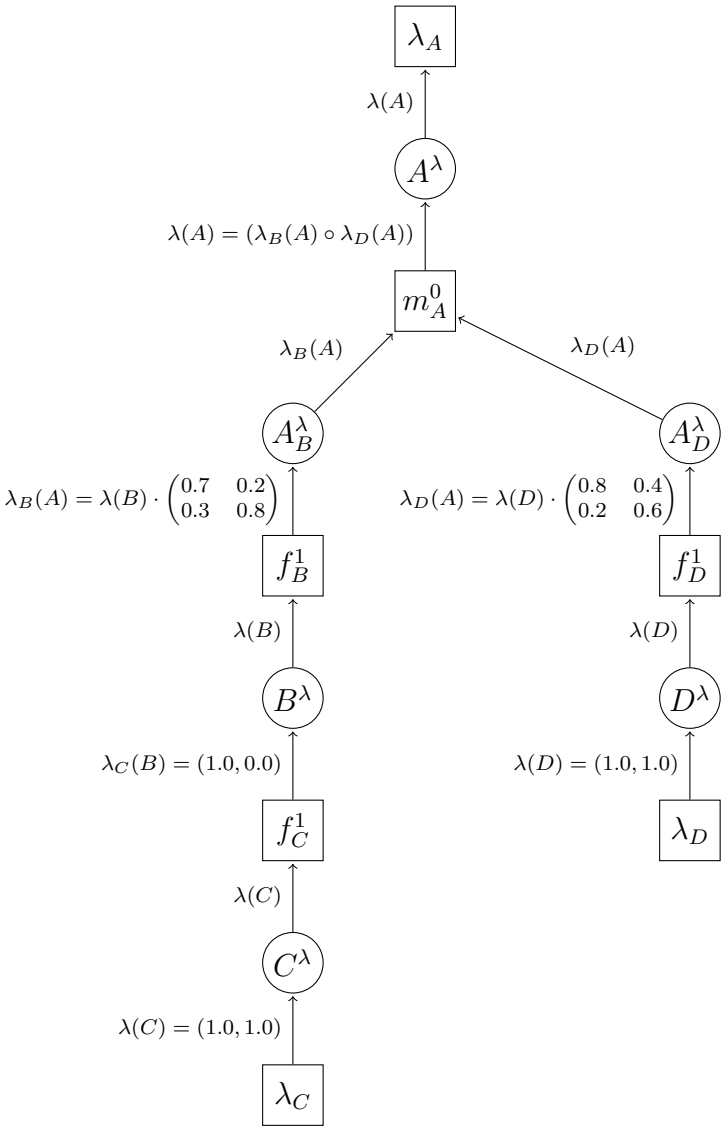


Figure 4.27.: Evidence Adjustment for $\lambda(A)$-t-invariant Example 4.12

on place B; then f_C^0 is enabled and puts

$$\begin{aligned} \pi(C) &= (1.0, 0.0) \cdot \begin{pmatrix} 0.4 & 0.6 \\ 0.001 & 0.999 \end{pmatrix} \\ &= (0.4, 0.6) \end{aligned}$$

on place C. Hence

$$\begin{aligned} BEL(C) &= \alpha\left(\lambda(C) \circ \pi(C)\right) = \alpha\left((1.0, 1.0) \circ (0.4, 0.6)\right) \\ &= (0.4, 0.6) \end{aligned}$$

- For the adjusted $\pi(D)$-t-invariant (see Figure 4.26), $\lambda_B(A) = (0.7, 0.2)$ (see above). Firing of m_A^2 puts

$$\pi_D(A) = \alpha\left(\pi(A) \circ \lambda_B(A)\right) = \alpha\left((0.1, 0.9) \circ (0.7, 0.2)\right) = (0.28, 0.72)$$

on place A; after firing of f_D^0

$$\begin{aligned} \pi(D) &= \pi_D(A) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix} = (0.28, 0.72) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix} \\ &= (0.512, 0.488) \end{aligned}$$

and thus

$$\begin{aligned} BEL(D) &= \alpha\left(\lambda(D) \circ \pi(D)\right) \\ &= \alpha\left((1.0, 1.0) \circ (0.512, 0.488)\right) = (0.512, 0.488) \end{aligned}$$

The interpretation is that after spouse dines with another ($\lambda(B) = (1.0, 0.0)$) the beliefs that

- spouse is cheating
- spouse is reported seen dining with another
- strange man/lady calls on the phone

have increased.

As a second evidence, it is additionally assumed that no strange man/lady calls on the phone ($\lambda(D) = (0.0, 1.0)$). So, the constant arc weight $(1.0, 1.0)$ at the output arc of transition λ_D has to be changed to $(0.0, 1.0)$[2]. This does not change $BEL(B)$ and

[2]The updated invariants are not shown anymore, because the principle of adjusting the arc labels should be clear by now.

$BEL(C)$. It changes $BEL(D)$ and $BEL(A)$:

$$\begin{aligned} BEL(D) &= \alpha\left(\lambda(D) \circ \pi(D)\right) = \alpha\left((0.0, 1.0) \circ (0.512, 0.488)\right) \\ &= \alpha(0.0, 4.88) = (0.0, 1.0) \text{ for } \alpha = \tfrac{1}{0.488} \end{aligned}$$

The anew adjusted $\lambda(A)$-t-invariant puts

$$\lambda_D(A) = \lambda(D) \cdot CPM_C^1 = (0.0, 1.0) \cdot \begin{pmatrix} 0.8 & 0.4 \\ 0.2 & 0.6 \end{pmatrix} = (0.2, 0.6)$$

on place A after firing of transition f_D^1. Then, after firing of transition m_A^0

$$\lambda(A) = \lambda_B(A) \circ \lambda_D(A) = (0.7, 0.2) \circ (0.2, 0.6) = (0.14, 0.12)$$

is put on A. Thus,

$$\begin{aligned} BEL(A) &= \alpha\left(\pi(A) \circ \lambda(A)\right) = \alpha\left((0.1, 0.9) \circ (0.14, 0.12)\right) \\ &= \alpha(0.014, 0.108) = (0.1148, 0.8852) \text{ for } \alpha = \tfrac{1}{0.122}. \end{aligned}$$

So, the belief that spouse is cheating decreased. □

Evidence propagation in the wet grass example is done the same way as described in the two examples above. Hence, Example 4.10 is continued in terms of a short summary.

Example 4.13 (Wet Grass: Evidence Propagation)
Now, Mr. Homes realizes that his grass is wet. This new evidence gives rise to slightly change the net $\mathcal{PB}$ of Figure 4.11 and the net representations of the t-invariants: $\lambda(H) = (1.0, 1.0)$ is changed to $\lambda(H) = (1.0, 0.0)$ thereby saying that Mr. Holmes' grass is wet, indeed. As only one arc label has changed, the adjusted net and its t-invariants are not displayed again. The resulting changes of the beliefs are put down

in the second column of Table 4.5 (the first column shows the initialization results):

$$BEL(r_1) = BEL(R = 1) = 0.2 \text{ changes to } 0.735$$
$$BEL(w_1) = BEL(W = 1) = 0.36 \text{ changes to } 0.788$$
$$BEL(s_1) = BEL(S = 1) = 0.1 \text{ changes to } 0.338.$$

Even the probability that the sprinkler was the cause for Mr. Holmes' grass being wet (s_1) increases. But it is more probable that rain was the cause and that as a consequence Dr. Watson's grass is also wet (w_1).

Finally, Mr. Holmes discovers that Dr. Watson's grass is also wet (w_1). Again, the new evidence results in a change of the net: $\lambda(W) = (1.0, 1.0)$ is changed to $\lambda(W) = (1.0, 0.0)$. Now, the beliefs change as follows:

$$BEL(r_1) = BEL(R = 1) = 0.735 \text{ changes to } 0.93$$
$$BEL(s_1) = BEL(S = 1) = 0.338 \text{ changes to } 0.16.$$

	Initialization	Evidence 1	Evidence 2
$\lambda(R)$	$(1.0, 1.0)$	$(1.0, 0.09)$	$(1.0, 0.018)$
$\pi(R)$ **given**	$(0.2, 0.8)$	$(0.2, 0.8)$	$(0.2, 0.8)$
$BEL(R)$	$(0.2, 0.8)$	$(0.735, 0.265)$	$(0.93, 0.07)$
$\lambda(S)$	$(1.0, 1.0)$	$(0.92, 0.2)$	$(0.956, 0.56)$
$\pi(S)$ **given**	$(0.1, 0.9)$	$(0.1, 0.9)$	$(0.1, 0.9)$
$BEL(S)$	$(0.1, 0.9)$	$(0.338, 0.662)$	$(0.16, 0.84)$
$\lambda(W)$ **given**	$(1.0, 1.0)$	$(1.0, 1.0)$	$(\mathbf{1.0}, \mathbf{0.0})$ **new ev.!**
$\pi(W)$	$(0.36, 0.64)$	$(0.788, 0.212)$	$(0.788, 0.212)$
$BEL(W)$	$(0.36, 0.64)$	$(0.788, 0.212)$	$(1.0, 0.0)$
$\lambda(H)$ **given**	$(1.0, 1.0)$	$(\mathbf{1.0}, \mathbf{0.0})$ **new ev.!**	$(1.0, 0.0)$
$\pi(H)$	$(0.272, 0.728)$	$(0.272, 0.728)$	$(0.272, 0.728)$
$BEL(H)$	$(0.272, 0.728)$	$(1.0, 0.0)$	$(1.0, 0.0)$

Table 4.8.: Simulation Results of Example 4.7 after Initialization and after Evidence Propagations

As a result, the probability of r_1 increases again (to 0.93). Now, it is virtually sure that rain was the cause for Mr. Holmes' grass being wet. On the other hand, the belief of s_1 decreases nearly to its initial value 0.1. It changes from 0.338 to 0.16

which is absolutely consequential. □

Altogether, it has to be pointed out that the changes of probabilities and beliefs induced by observations and evidences are directly propagated via the t-invariants in the probability propagation net. Thereby, it is easy to follow how changes of one random variable (for example by instantiation) influence the other beliefs. By that, the Petri net approach adds transparency to the Bayesian network approach of message propagation by integrating the propagation algorithms into the graph structure and into the graph's dynamics. The considerable complexity of the resulting Petri nets indicates that the propagation algorithms – which are hidden in the Bayesian network representation – are by far not trivial. Particularly, when model parameters have to be adjusted or fine-tuned, the understanding of the propagation processes, dependabilities, and mutual influences is essential. Because of the advantages mentioned above, the Petri net approach could be a good choice for the process of fine-tuning.

For some applications these information may not be important. Perhaps the models are given and just the beliefs after initialization or after evidence propagation are of interest. In such cases, the Petri net representation at first sight does not add any benefit to the Bayesian network approach at all. There may be a benefit if the higher expressiveness of Petri nets is used (which will be sketched later on) but if that is not needed as well, the most convenient way in this case probably is to use a Bayesian network tool to build the model and to calculate the desired results.

Yet, as some Petri net classes are a perfect means to describe (technical) processes, it may be interesting to couple the abilities of probability propagation nets with Petri net concepts for technical systems. As an example, fault tree analysis can be done with the Petri net approach via the Bayesian network representation of fault trees and the transformation of these Bayesian networks into probability propagation nets. This will be shown in chapter 5.

Before that, the representation of conditioned Bayesian networks with probability propagation nets is described in the next section. Another folding of the high-level Petri net representation leads to matrices as tokens instead of vectors. By that, the different instantiations for the cut variables can be evaluated in one pass through the t-invariants.

4.4. Representing Conditioned Bayesian Networks with High-Level Probability Propagation Nets

The metastatic cancer example (see Examples 2.11 and 2.12) will now be used to show how an additional (kind of) folding of high-level probability propagation nets is capable of representing conditioned Bayesian networks. This section is just to sketch the potential of probability propagation nets and since the basic idea is quite plausible, there will be no formal definition or proof given in this section. Yet, it has to be mentioned that if there are lots of loops in the Bayesian network, the conditioning approach leads to complex structures which then are represented by big matrices in this kind of probability propagation net.

For the understanding of the propagation process in this probability propagation net, the special matrix products of Definition 2.8 are needed.

Example 4.14 (Metastatic Cancer: Probability Propagation Net)
The probability propagation net representing the conditioned Bayesian network(s) is shown in Figure 4.28. Table 4.9 contains the transition functions of the functional transitions. Note that transitions π_{A_1}, π_{A_2}, λ_D and λ_E create matrices instead of pairs when firing. This is due to the folding of two conditioned Bayesian networks into one representation. The upper row of the matrix represents the Bayesian network in which A is instantiated for 1, the lower row represents A being instantiated for 0. As a consequence, matrices instead of pairs flow through the net. By that, both instantiations can be evaluated simultaneously in one pass through the t-invariants.

In order to show how to work with the probability propagation net, the empty marking in the $\pi(D)$-t-invariant (see Figure 4.29) is reproduced. The (constant) matrix $\left(\begin{smallmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{smallmatrix}\right)$ is the arc label of both arcs (π_{A_1}, A_1^π) and (π_{A_2}, A_2^π). So, firing both transitions π_{A_1} and π_{A_2} puts $\pi(A_i) = \left(\begin{smallmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{smallmatrix}\right)$ on places A_1^π and A_2^π. This is the beginning of a simultaneous evaluation of both Bayesian networks in Figure 2.5. Similarly, firing λ_E puts $\lambda(E) = \left(\begin{smallmatrix}1.0 & 1.0\\ 1.0 & 1.0\end{smallmatrix}\right)$ on place E^λ. This indicates in both evaluations that no evidence or observation is known that forces $\lambda(E)$ to be adjusted. Next, firing of f_B^0, f_C^0, f_E^1 removes $\pi_B(A) = \left(\begin{smallmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{smallmatrix}\right), \pi_C(A) = \left(\begin{smallmatrix}1.0 & 0.0\\ 0.0 & 1.0\end{smallmatrix}\right), \lambda(E) = \left(\begin{smallmatrix}1.0 & 1.0\\ 1.0 & 1.0\end{smallmatrix}\right)$ from

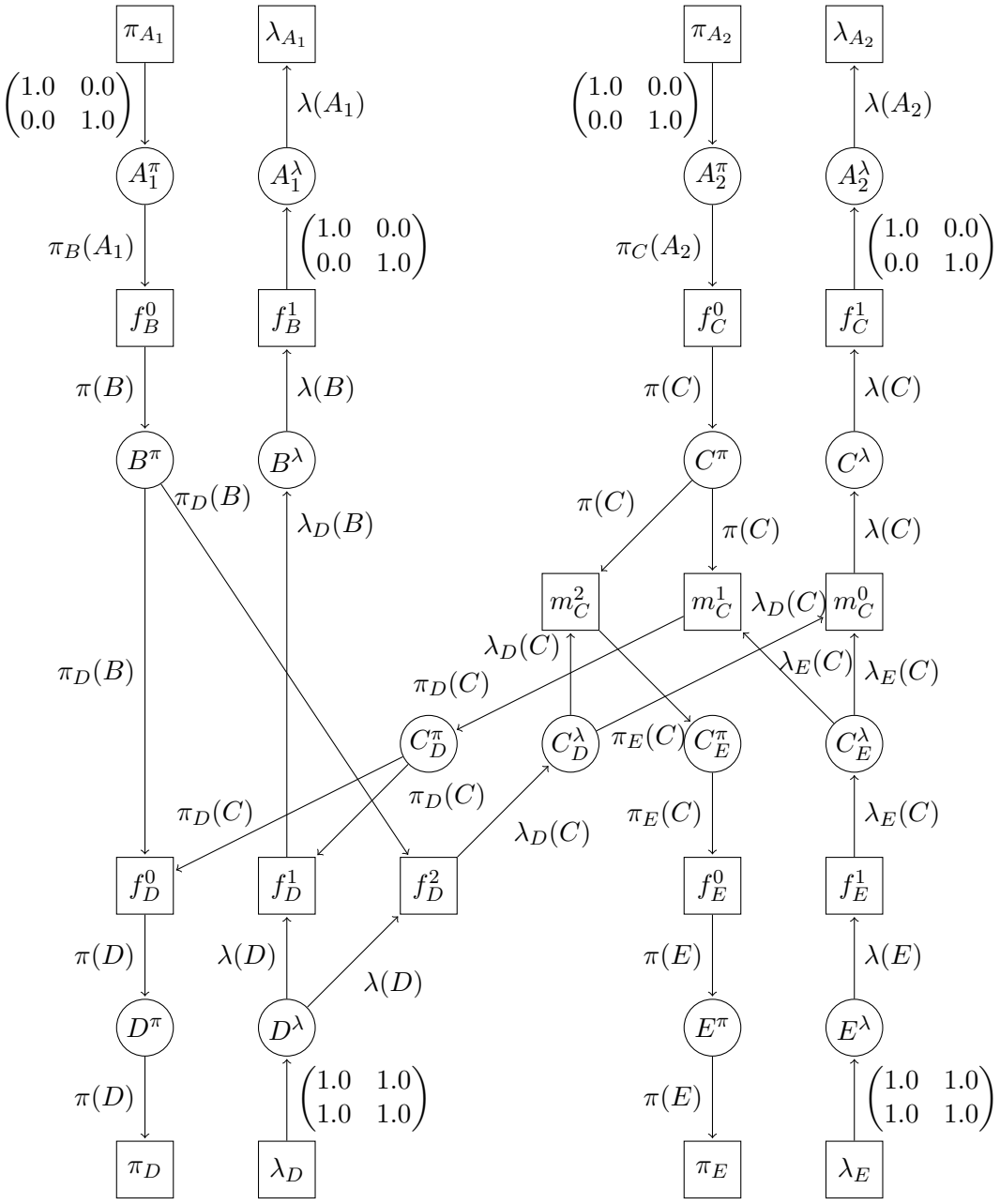


Figure 4.28.: Probability Propagation Net for the Metastatic Cancer Example 4.14

$A_1^\pi, A^2\pi, E^\lambda$ and puts

$$\pi(B) = \pi_B(A) \cdot \begin{pmatrix}0.8 & 0.2\\0.2 & 0.8\end{pmatrix} = \begin{pmatrix}1.0 & 0.0\\0.0 & 1.0\end{pmatrix} \cdot \begin{pmatrix}0.8 & 0.2\\0.2 & 0.8\end{pmatrix} = \begin{pmatrix}0.8 & 0.2\\0.2 & 0.8\end{pmatrix},$$

$f_B^0 \simeq P(B \mid A_1) =$

	B	
A_1	1	0
1	0.8	0.2
0	0.2	0.8

$f_C^1 \simeq P^{A_2 \leftarrow C}(C \mid A_2) =$

	A_2	
C	1	0
1	0.2	0.05
0	0.8	0.95

$f_B^1 \simeq P^{A_1 \leftarrow B}(B \mid A_1) =$

	A_1	
B	1	0
1	0.8	0.2
0	0.2	0.8

$f_D^0 \simeq P(D \mid BC) =$

		D	
B	C	1	0
1	1	0.8	0.2
1	0	0.8	0.2
0	1	0.8	0.2
0	0	0.05	0.95

$f_C^0 \simeq P(C \mid A_2) =$

	C	
A_2	1	0
1	0.2	0.8
0	0.05	0.95

$f_D^1 \simeq P^{B \leftarrow CD}(D \mid BC) =$

		B	
C	D	1	0
1	1	0.8	0.8
1	0	0.2	0.2
0	1	0.8	0.05
0	0	0.2	0.95

$f_E^0 \simeq P(E \mid C) =$

	E	
C	1	0
1	0.8	0.2
0	0.6	0.4

$f_E^1 \simeq P^{C \leftarrow E}(E \mid C) =$

	C	
E	1	0
1	0.8	0.6
0	0.2	0.4

$f_D^2 \simeq P^{C \leftarrow BD}(D \mid BC) =$

		C	
B	D	1	0
1	1	0.8	0.8
1	0	0.2	0.2
0	1	0.8	0.05
0	0	0.2	0.95

Table 4.9.: Transition Functions of Example 4.14

$$\pi(C) = \pi_C(A) \cdot \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \cdot \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} = \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix},$$

and

$$\lambda_E(C) = \lambda(E) \cdot \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix}$$

on B^π, C^π and C_E^λ, respectively.

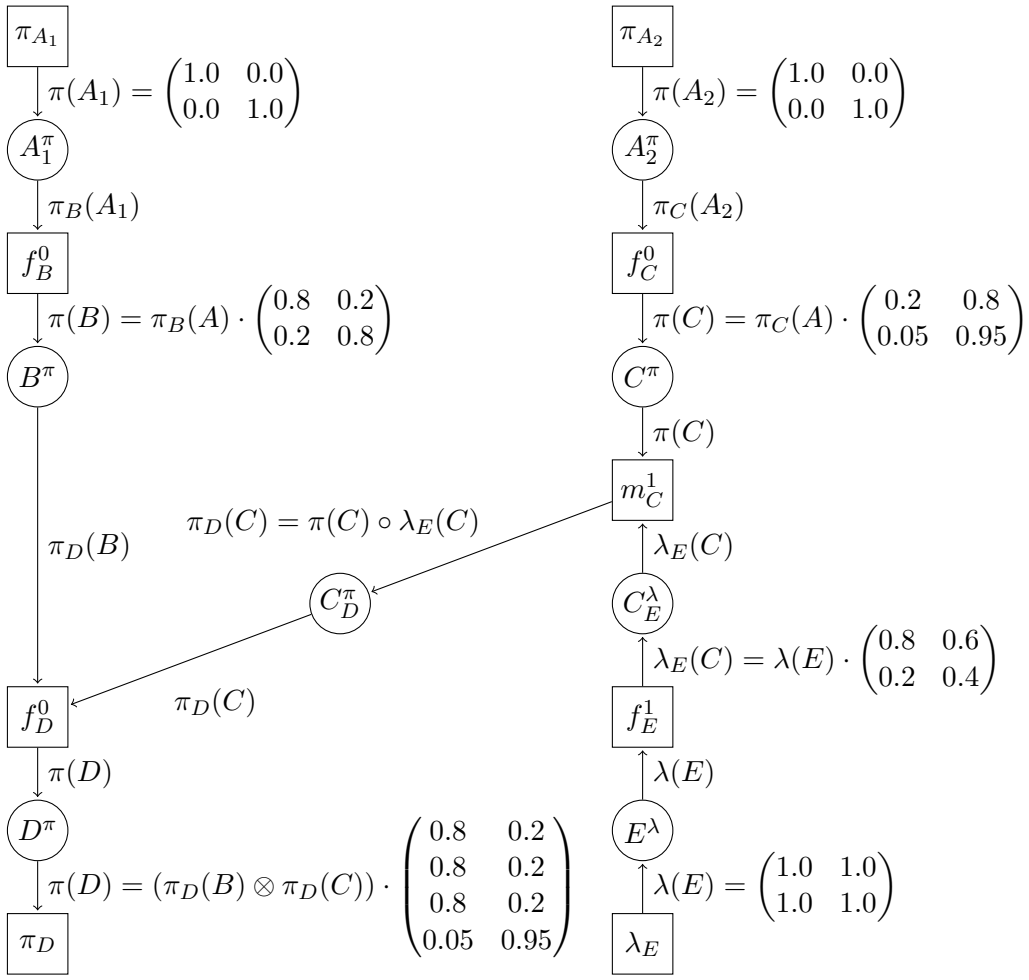


Figure 4.29.: $\pi(D)$-Invariant of Example 4.14

Now, transition m_C^1 is enabled, removes $\pi(C)$ and $\lambda_E(C)$ from places C^π and C_E^λ, and puts

$$\begin{aligned}\pi_D(C) = \pi(C) \circ \lambda_E(C) &= \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \circ \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \\ &= \begin{pmatrix} 0.2 \cdot 1.0 & 0.8 \cdot 1.0 \\ 0.05 \cdot 1.0 & 0.95 \cdot 1.0 \end{pmatrix} = \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix}\end{aligned}$$

on place C_D^π, thus enabling transition f_D^0.

Firing f_D^0 consists of clearing the input places B^π and C_D^π and putting $\pi(D)$ on D^π

which is calculated as follows:

$$
\begin{aligned}
\pi(D) &= (\pi_D(B) \otimes \pi_D(C)) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \left(\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \otimes \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \right) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \begin{pmatrix} 0.8 \cdot 0.2 & 0.8 \cdot 0.8 & 0.2 \cdot 0.2 & 0.2 \cdot 0.8 \\ 0.2 \cdot 0.05 & 0.2 \cdot 0.95 & 0.8 \cdot 0.05 & 0.8 \cdot 0.95 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \begin{pmatrix} 0.16 & 0.64 & 0.04 & 0.16 \\ 0.01 & 0.19 & 0.04 & 0.76 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \begin{pmatrix} 0.68 & 0.32 \\ 0.23 & 0.77 \end{pmatrix}.
\end{aligned}
$$

In completing the **0**-reproduction, transition π_D empties the net representation of the t-invariant. The results so far are the probabilities $\pi(B), \pi(C), \pi(D)$. By **0**-reproduction in the remaining three t-invariants, the results listed in Table 4.14 are calculated.

As a last step the beliefs are computed. They have to be adjusted by the actual values for the cut variables. In this case, A was instantiated which led to the matrices instead of vectors flowing through the Petri net. When computing the beliefs these matrices are "flattened" by multiplying them with the current "belief-vector" of A:

$$\pi(A) = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \qquad \pi(B) = \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix}$$
$$\pi(C) = \begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \qquad \pi(D) = \begin{pmatrix} 0.68 & 0.32 \\ 0.23 & 0.77 \end{pmatrix}$$
$$\pi(E) = \begin{pmatrix} 0.64 & 0.36 \\ 0.61 & 0.39 \end{pmatrix}$$
$$\lambda(A) = \lambda(B) = \lambda(C) = \lambda(D) = \lambda(E) = \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix}$$

Table 4.10.: Probability Propagation Net Initialization Results of the Metastatic Cancer Example

$$\begin{aligned} BEL(B) &= BEL(A) \cdot (\pi(B) \circ \lambda(B)) \\ &= (0.2, 0.8) \cdot \left(\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \circ \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \right) \\ &= (0.2, 0.8) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \\ &= (0.32, 0.68) \end{aligned}$$

$$\begin{aligned} BEL(C) &= BEL(A) \cdot (\pi(C) \circ \lambda(C)) \\ &= (0.2, 0.8) \cdot \left(\begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \circ \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \right) \\ &= (0.08, 0.92) \end{aligned}$$

$$\begin{aligned} BEL(D) &= BEL(A) \cdot (\pi(D) \circ \lambda(D)) \\ &= (0.2, 0.8) \cdot \left(\begin{pmatrix} 0.68 & 0.32 \\ 0.23 & 0.77 \end{pmatrix} \circ \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \right) \\ &= (0.32, 0.68) \end{aligned}$$

$$
\begin{aligned}
BEL(E) &= BEL(A) \cdot (\pi(E) \circ \lambda(E)) \\
&= (0.2, 0.8) \cdot \left(\begin{pmatrix} 0.64 & 0.36 \\ 0.61 & 0.39 \end{pmatrix} \circ \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \right) \\
&= (0.616, 0.384).
\end{aligned}
$$

That means: Given the probability for metastatic cancer $P(A) = (0.2, 0.8)$, the beliefs (probabilities) for increased total serum calcium, brain tumor, coma and severe headaches are $BEL(B) = (0.32, 0.68)$, $BEL(C) = (0.08, 0.92)$, $BEL(D) = (0.32, 0.68)$ and $BEL(E) = (0.616, 0.384)$, respectively. □

Remark 4.10

For this propagation, the normalization of vectors or matrices was not needed. In analogy to the normalizing constant α for vectors, when bringing in evidence, some matrix products need to be normalized (see Remark 4.6). In this case, normalizing a matrix means multiplying each row r_i separately with a corresponding constant α_i such that the components of r_i sum up to 1.0. This can be done by multiplying the matrix with the diagonal matrix

$$
\begin{pmatrix} \alpha_1 & 0 & \dots & 0 \\ 0 & \alpha_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_n \end{pmatrix}
$$

with $\alpha_i := \frac{1}{\sum_{j=1}^{n} r_{ij}}$. ◇

Example 4.15 (Metastatic Cancer: Evidence Propagation)

Now, the patient moans about severe headaches. Hence, E has to be instantiated for 1. In the probability propagation net this is done by changing the labels of the arc (λ_E, E^{λ}) into $\left(\begin{smallmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \end{smallmatrix}\right)$, which is not shown in picture. As a consequence, the belief of A has to be adjusted as follows:

$$\pi(E) = \begin{pmatrix} P(E=1 \mid A=1) & P(E=0 \mid A=1) \\ P(E=1 \mid A=0) & P(E=0 \mid A=0) \end{pmatrix} = \begin{pmatrix} 0.64 & 0.36 \\ 0.61 & 0.39 \end{pmatrix},$$
$$P(A) = (0.2, 0.8) = BEL(A)$$

and thus

$$\begin{aligned} BEL(A \mid E=1) &= \alpha \cdot P(E=1 \mid A) \circ P(A) = \alpha \cdot (0.64 \cdot 0.2, 0.61 \cdot 0.8) \\ &= \alpha \cdot (0.128, 0.488) \\ &= (0.208, 0.792) \end{aligned}$$

Hence, $BEL'(A) = (0.208, 0.792)$.

The propagation is done by reproducing the empty marking in the $\pi(D)$-invariant. After firing π_{A_1}, π_{A_2}, f_B^0 and f_C^0, place B^π is marked with $\left(\begin{smallmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{smallmatrix}\right)$ and C^π is marked with $\left(\begin{smallmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{smallmatrix}\right)$. But in contrast to the initialization shown above, λ_E now puts the token $\left(\begin{smallmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \end{smallmatrix}\right)$ on place E^λ.

Then, f_E^1 fires and puts the matrix

$$\lambda_E(C) = \lambda(E) \cdot \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix}$$

on place C_E^λ, which enables m_C^1.

After firing of m_C^1 the matrix

$$\begin{aligned} \pi_D(C) &= \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot (\pi(C) \circ \lambda_E(C)) \\ &= \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \left(\begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \circ \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix} \right) \\ &= \begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \begin{pmatrix} 0.16 & 0.48 \\ 0.04 & 0.57 \end{pmatrix} = \begin{pmatrix} 0.25 & 0.75 \\ 0.066 & 0.934 \end{pmatrix} \end{aligned}$$

is put on place C_D^{π}. Now, transition f_D^0 fires and puts the token

$$\begin{aligned}
\pi(D) &= (\pi_D(B) \otimes \pi_D(C)) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \left(\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \otimes \begin{pmatrix} 0.25 & 0.75 \\ 0.066 & 0.934 \end{pmatrix} \right) \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \begin{pmatrix} 0.2 & 0.6 & 0.05 & 0.15 \\ 0.0132 & 0.1868 & 0.0528 & 0.7472 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.8 & 0.2 \\ 0.05 & 0.95 \end{pmatrix} \\
&= \begin{pmatrix} 0.6875 & 0.3125 \\ 0.2396 & 0.7604 \end{pmatrix}
\end{aligned}$$

on place D^{π} from where it is removed by firing of transition π_D, thus completing the **0**-reproduction in the $\pi(D)$-invariant.

So, the new belief of D is:

$$\begin{aligned}
BEL'(D) &= BEL'(A) \cdot (\pi(D) \circ \lambda(D)) \\
&= (0.208, 0.792) \cdot \left(\begin{pmatrix} 0.6875 & 0.3125 \\ 0.2396 & 0.7604 \end{pmatrix} \circ \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \right) \\
&= (0.208, 0.792) \cdot \begin{pmatrix} 0.6875 & 0.3125 \\ 0.2396 & 0.7604 \end{pmatrix} = (0.3328, 0.6672).
\end{aligned}$$

In conclusion the belief in a coma (D) is almost unchanged. Simulating the $\lambda(A_1)$-invariant also does not influence the belief in increased total serum (B), so it will not be demonstrated. In contrast to this, the $\lambda(A_2)$-invariant shown in Figure 4.30 influences the belief in a brain tumor (C).

To reproduce the empty marking in the $\lambda(A_2)$-invariant, π_{A_1}, f_B^0, λ_D and f_D^2 fire.

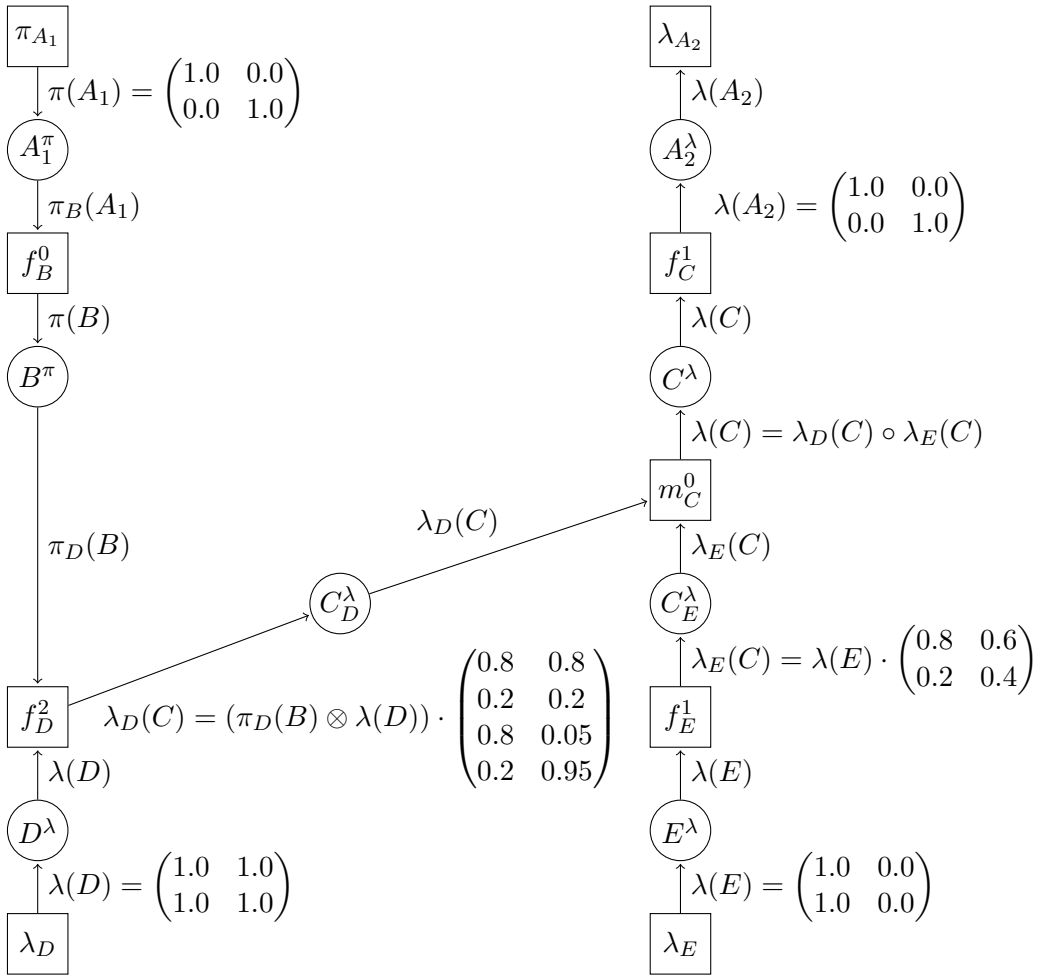


Figure 4.30.: $\lambda(A_2)$-Invariant of Example 4.14 with Evidence

This results in token

$$\lambda_D(C) = (\pi_D(B) \otimes \lambda(D)) \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix}$$

$$= \left(\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \otimes \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \right) \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix}$$

$$= \begin{pmatrix} 0.8 & 0.8 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.8 & 0.8 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} = \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix}$$

put on C_D^λ by firing of f_D^2. To enable m_C^0, transitions λ_E and f_E^1 have to fire in that order. This yields the tuple

$$\lambda_E(C) = \lambda(E) \cdot \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.4 \end{pmatrix} = \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix}$$

on place C_E^λ.

Now, m_C^0 fires and puts

$$\lambda(C) = \lambda_D(C) \circ \lambda_E(C) = \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} \circ \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix} = \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix}$$

on place C^λ. Firing f_C^1 and λ_{A_2} completes the **0**-reproduction but does not change any values of A_2, because A_2 is instantiated.

So the belief of C has to be adjusted and is now:

$$\begin{aligned} BEL'(C) &= BEL'(A) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot (\pi(C) \circ \lambda(C)) \right) \\ &= (0.208, 0.792) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \left(\begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \circ \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix} \right) \right) \\ &= (0.208, 0.792) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \begin{pmatrix} 0.16 & 0.48 \\ 0.04 & 0.57 \end{pmatrix} \right) \\ &= (0.208, 0.792) \cdot \begin{pmatrix} 0.25 & 0.75 \\ 0.0656 & 0.9344 \end{pmatrix} \\ &= (0.1040, 0.8960). \end{aligned}$$

To sum it up, observing severe headaches causes the belief in a brain tumor (C) to increase from 0.08 to 0.1040.

Now, as the patient moans about severe headaches, it seems obvious that he did not

fall into a coma. So to conclude the example, D has to be instantiated for $\left(\begin{smallmatrix}0.0 & 1.0\\ 0.0 & 1.0\end{smallmatrix}\right)$. The corresponding arc label (λ_D, D^λ) is adjusted properly. To check the influence of these evidences on B and C, the $\lambda(A_1)$- and $\lambda(A_2)$-invariants have to be simulated.

First, because of the new evidence, the belief in A has to be updated:

$$\begin{aligned} P(A \mid E=1, D=0) &= \alpha P(D=0 \mid A, E=1) \circ P(A \mid E=1) \\ &= \alpha(0.3125 \cdot 0.208, 0.7604 \cdot 0.792) \\ &= \alpha(0.065, 0.602) = (0.0975, 0.9025). \end{aligned}$$

Hence, $BEL''(A) = (0.0975, 0.9025)$ is the new weight for the beliefs of B and C.

After firing π_{A_2}, f_C^0, λ_E, f_E^1, m_C^1 and λ_D, place C_D^π is marked with $\left(\begin{smallmatrix}0.25 & 0.75\\ 0.066 & 0.934\end{smallmatrix}\right)$ and place D^λ is marked with $\left(\begin{smallmatrix}0.0 & 1.0\\ 0.0 & 1.0\end{smallmatrix}\right)$.

Now transition f_D^1 fires putting

$$\begin{aligned} \lambda_D(B) &= (\pi_D(C) \otimes \lambda(D)) \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} \\ &= \left(\begin{pmatrix} 0.25 & 0.75 \\ 0.066 & 0.934 \end{pmatrix} \otimes \begin{pmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \right) \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} \\ &= \begin{pmatrix} 0.0 & 0.25 & 0.0 & 0.75 \\ 0.0 & 0.066 & 0.0 & 0.934 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} \\ &= \begin{pmatrix} 0.2 & 0.7625 \\ 0.2 & 0.9005 \end{pmatrix} \end{aligned}$$

on place B^λ.

The remaining firings for completing the $\lambda(A_2)$-invariant do not change any values. As a last step, the $\lambda(A_2)$-invariant has to be simulated to check the influence on C.

To save space, the beginning of the **0**-reproduction is not shown in detail. The

interesting part starts with firing of f_D^2 putting

$$\begin{aligned}\lambda_D(C) &= (\pi_D(B) \otimes \lambda(D)) \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} \\ &= \left(\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \otimes \begin{pmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \right) \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} \\ &= \begin{pmatrix} 0.0 & 0.8 & 0.0 & 0.2 \\ 0.0 & 0.2 & 0.0 & 0.8 \end{pmatrix} \cdot \begin{pmatrix} 0.8 & 0.8 \\ 0.2 & 0.2 \\ 0.8 & 0.05 \\ 0.2 & 0.95 \end{pmatrix} = \begin{pmatrix} 0.2 & 0.35 \\ 0.2 & 0.8 \end{pmatrix}\end{aligned}$$

on place C_D^λ.

As C_E^λ by firing of λ_E and f_E^1 is marked with $\left(\begin{smallmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{smallmatrix}\right)$, m_C^0 fires next:

$$\begin{aligned}\lambda(C) = \lambda_D(C) \circ \lambda_E(C) &= \begin{pmatrix} 0.2 & 0.35 \\ 0.2 & 0.8 \end{pmatrix} \circ \begin{pmatrix} 0.8 & 0.6 \\ 0.8 & 0.6 \end{pmatrix} \\ &= \begin{pmatrix} 0.16 & 0.21 \\ 0.16 & 0.48 \end{pmatrix}\end{aligned}$$

is placed on C^λ.

Firing of f_C^1 and λ_{A_2} completes the **0**-reproduction but does not influence the values for A_2 as A_2 is instantiated.

Altogether, the given evidence leads to new beliefs for B and C:

$$\begin{aligned}
BEL''(B) &= BEL''(A) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot (\pi(B) \circ \lambda(B)) \right) \\
&= BEL''(A) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \left(\begin{pmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{pmatrix} \circ \begin{pmatrix} 0.2 & 0.7625 \\ 0.2 & 0.9005 \end{pmatrix} \right) \right) \\
&= (0.0975, 0.9025) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \begin{pmatrix} 0.16 & 0.1525 \\ 0.04 & 0.7204 \end{pmatrix} \right) \\
&= (0.0975, 0.9025) \cdot \begin{pmatrix} 0.512 & 0.488 \\ 0.0526 & 0.9474 \end{pmatrix} \\
&= (0.0974, 0.9026)
\end{aligned}$$

$$\begin{aligned}
BEL''(C) &= BEL''(A) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot (\pi(C) \circ \lambda(C)) \right) \\
&= BEL''(A) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \left(\begin{pmatrix} 0.2 & 0.8 \\ 0.05 & 0.95 \end{pmatrix} \circ \begin{pmatrix} 0.16 & 0.21 \\ 0.16 & 0.48 \end{pmatrix} \right) \right) \\
&= (0.0975, 0.9025) \cdot \left(\begin{pmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{pmatrix} \cdot \begin{pmatrix} 0.032 & 0.168 \\ 0.008 & 0.456 \end{pmatrix} \right) \\
&= (0.0975, 0.9025) \cdot \begin{pmatrix} 0.16 & 0.84 \\ 0.0172 & 0.9828 \end{pmatrix} \\
&= (0.0311, 0.9689).
\end{aligned}$$

Finally, it is discovered that the new evidence decreased the belief in a brain tumor (C) by about two third. Also the belief in increased serum calcium (B) is way smaller given the evidence that the patient has severe headaches (E) but has not fallen into a coma (D). Additionally, compared to the initial probability, the belief in metastatic cancer (A) was halved, which is absolutely consequential. □

As stated above, the structure of Bayesian networks is a bit meager and the steps of probability propagation are "hidden" in the algorithms. In contrast to that, the Petri net representation shows the probability propagation in detail, and the algorithms

are distributed over the net such that each transition's share is of manageable size.[3]

On the one hand, in spite of the exactness of the Petri net representation, one might consider the size of the net as a disadvantage. On the other hand, the size of the probability propagation nets might indicate that Bayesian networks are indeed a little under-structured. Moreover, the specific structure of probability propagation nets causes an absolute appropriate partition into propagation processes. The minimal t-invariants (precisely their net representations) exactly describe the paths of probabilities and likelihoods. The t-invariants do not have to be calculated on the higher level, necessarily. It is sufficient to calculate them on the "black" net (that means the underlying place/transition net without arc labels).

Additionally, even conditioned Bayesian networks can be represented with minimal adaption of the firing rule allowing matrices instead of tuples to flow through the net, whereby the Petri net structure is almost completely maintained. By that, probability propagation nets are a perfect means to describe causality or mutual dependencies and influence. Since Petri nets are widely-used to model technical systems or processes which are influenced by risks, the Petri net representation of Bayesian networks could be integrated with those Petri nets, thus joining the worlds of technical processes and Bayesian networks.

[3]The proof that high-level probability propagation nets completely represent the Bayesian message propagation is sketched in Appendix B.

5. Representing Fault Trees with High-Level Probability Propagation Nets

In section 2.4.2 it was described how fault trees can be translated into Bayesian networks. Hence, they can also be represented by probability propagation nets. As fault trees may contain loops, the Bayesian network translation of the fault tree has to be made loop-free before being transformed into a probability propagation net. In the first instance, the Petri net representation does not add any expressiveness compared to the Bayesian network approach. But it will be suggested later on that they can easily be extended.

5.1. Representing the Fault Tree Semantics

Fault trees can be represented by probability propagation nets with the intermediate step of transforming them into Bayesian networks which, once the loops have been eliminated for example by conditioning, are the input to generate the high-level probability propagation nets. The Bayesian network representation of fault trees already adds expressiveness to the models, as stated and exemplified in [Bobbio*99, PorBob99]. Yet, the missing transparency of the Bayesian network message propagation may be considered as a disadvantage. Thus, the probability propagation net representation of fault trees can be used to increase transparency and to better adjust the parameters. This will be shown by continuing Example 2.15.

Example 5.1 (Multiprocessor System: Conditioned Bayesian Network)
As the Bayesian network of Figure 2.11 – which results from applying the transforma-

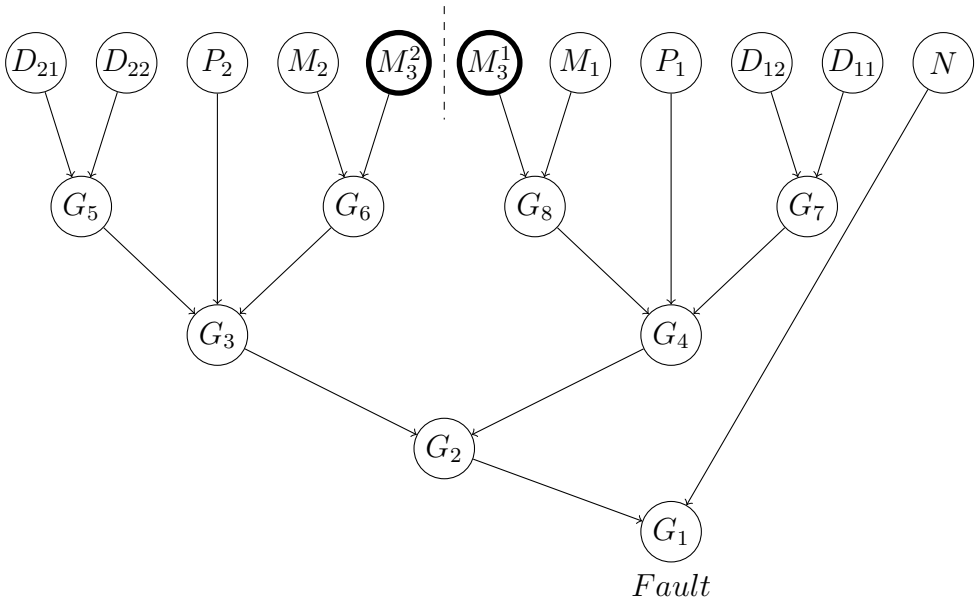


Figure 5.1.: Conditioned Bayesian Network for the Multiprocessor System

tion rules given in section 2.4.2 to the fault tree shown in Figure 2.7 – contains a loop, and this loop has to be eliminated first. Again this is done by conditioning (see section 2.3.3). As proposed above, the loop is cut at the root element. Figure 5.1 shows the modified Bayesian network. The node M_3 has been cut and is now represented as M_3^1 linked to the first subsystem and M_3^2 linked to the second one. Hence, the conditional probability tables of G_6 and G_8 must be slightly changed by substituting M_3 for M_3^2 and for M_3^1, respectively. Except for this quite simple modification, the conditional probability tables are preserved as given in Table 2.7. □

Remark 5.1

When propagating the messages in the conditioned Bayesian network, M_3 (and thus M_3^1 and M_3^2) has to be instantiated for all of its elementary events. For every instantiation, the complete message propagation has to be executed (cf. section 2.3.3). In contrast to the simultaneous evaluation of the instantiations by propagating matrices instead of vectors as shown in section 4.4, this example demonstrates the separate evaluation of the instantiations. ◇

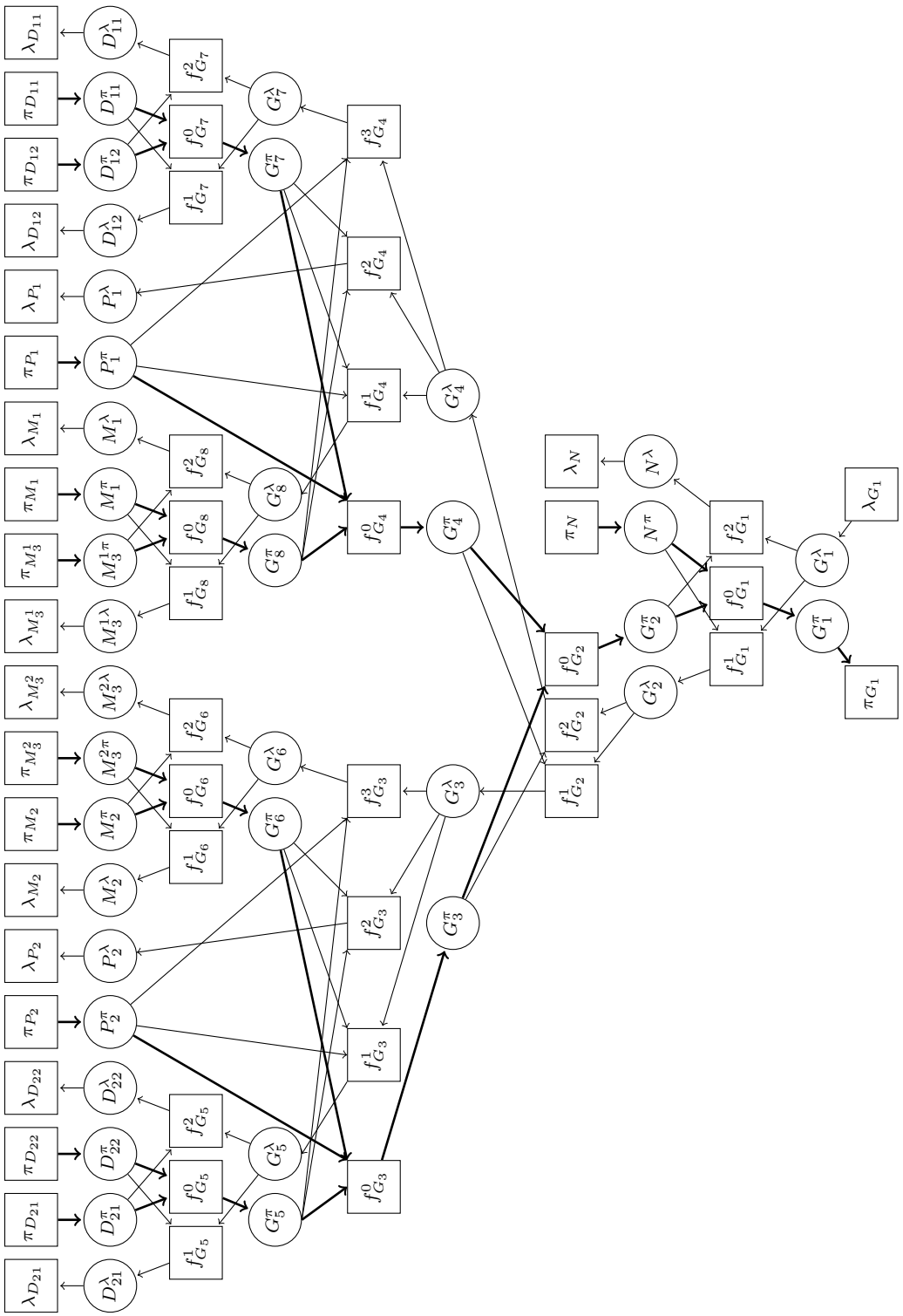


Figure 5.2.: Multiprocessor System Probability Propagation Net

Now that the Bayesian network has been made free of loops, it can be translated into the corresponding high-level probability propagation net by applying the rules defined in section 4.1. Similar to the Bayesian network message propagation, the initial simulation of minimal t-invariants has to be executed for every instantiation of M_3 separately and the weighted simulation results have to be summed up in a final step. This is demonstrated by continuing Example 5.1.

Example 5.2 (Multiprocessor System: Probability Propagation Net)
Figure 5.2 shows the high-level probability propagation net for the multiprocessor system example. To save space, the arc labels are not included in the figure. The corresponding transition functions are listed in extracts in Table 5.1. As the transition functions are very simple, the different sortings of the corresponding matrices are omitted in the table (that means $f^1_{G_1}, f^2_{G_1}, f^1_{G_2}, \ldots, f^2_{G_8}$ are missing).

In order to calculate the failure probability of the system it suffices to simulate the only one minimal π-t-invariant (which is highlighted in Figure 5.2 by bold edges), because – as no observations or evidences are given – all likelihoods are set to the neutral value $(1.0, 1.0)$. As the original Bayesian network is loopy and thus the probability propagation net is the representation of a conditioned Bayesian network, the cut variable must be instantiated for its elementary events and separate simulations have to be executed as described in section 2.3.3.

Instantiation: Faulty Global Memory

The simulation with M_3 (M^1_3, M^2_3) being instantiated for 1, i.e $\pi(M_3) = \pi(M^1_3) = \pi(M^2_3) = (1.0, 0.0)$, looks as follows:

First, Transitions $\pi_{D_{21}}$, $\pi_{D_{22}}$, π_{P_2}, π_{M_2}, $\pi_{M^2_3}$, $\pi_{M^1_3}$,π_{M_1}, π_{P_1}, $\pi_{D_{11}}$, $\pi_{D_{12}}$ and π_N fire, putting tuples $(0.32968, 0.67032)$ on places D^π_{21}, D^π_{22}, D^π_{11}, and D^π_{12}, respectively, tuples $(0.0025, 0.9975)$ on places P^π_2 and P^π_1, tuples $(0.00015, 0.99985)$ on places M^π_2 and M^π_1, tuples $(1.0, 0.0)$ on $M^{2\pi}_3$ and $M^{1\pi}_3$, and furthermore tuple $(0.00001, 0.99999)$ on place N^π.

$\pi_{D_{ij}} \simeq P(D_{ij}) =$

D_{ij}	
1	0
0.32968	0.67032

$\pi_{P_i} \simeq P(P_i) =$

P_i	
1	0
0.0025	0.9975

$\pi_{M_k} \simeq P(M_k) =$

M_k	
1	0
0.00015	0.99985

$\pi_N \simeq P(N) =$

N	N	
	1	0
	0.00001	0.99999

$i, j \in \{1, 2\}$, $k \in \{1, 2, 3\}$

$f^0_{G_1} \simeq P(G_1) =$

		G_1	
G_2	N	1	0
1	1	1.0	0.0
1	0	1.0	0.0
0	1	1.0	0.0
0	0	0.0	1.0

$f^0_{G_2} \simeq P(G_2) =$

		G_2	
G_3	G_4	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

$f^0_{G_3}[f^0_{G_4}] \simeq P(G_3)[P(G_4)] =$

			$G_3\,[G_4]$	
$P_2\,[P_1]$	$G_5\,[G_7]$	$G_6\,[G_8]$	1	0
1	1	1	1.0	0.0
1	1	0	1.0	0.0
1	0	1	1.0	0.0
1	0	0	1.0	0.0
0	1	1	1.0	0.0
0	1	0	1.0	0.0
0	0	1	1.0	0.0
0	0	0	0.0	1.0

$f^0_{G_5}[f^0_{G_7}] \simeq P(G_5)[P(G_7)] =$

		$G_5\,[G_7]$	
D_{l1}	D_{l2}	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

$f^0_{G_6}[f^0_{G_8}] \simeq P(G_6)[P(G_8)] =$

		$G_6\,[G_8]$	
M_l	M_3	1	0
1	1	1.0	0.0
1	0	0.0	1.0
0	1	0.0	1.0
0	0	0.0	1.0

$l \in \{1, 2\}$

Table 5.1.: Transition Functions Extract of the Multiprocessor Example

Now, transition $f^0_{G_5}$ is activated and puts

$$\begin{aligned}
\pi(G_5) &= (\pi(D_{21}) \times \pi(D_{22})) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= ((0.32968, 0.67032) \times (0.32968, 0.67032)) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= (0.10868890, 0.22099110, 0.22099110, 0.44932890) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= (0.10868890, 0.89131110)
\end{aligned}$$

on place G^π_5. The same tuple is being placed on G^π_7 by firing of transition $f^0_{G_7}$.

Transitions $f^0_{G_6}$ and $f^0_{G_8}$ fire symmetrically, too and put

$$\begin{aligned}
\pi(G_6) = \pi(G_8) &= \left(\pi(M_2) \times \pi(M^2_3)\right) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= \left(\pi(M_1) \times \pi(M^1_3)\right) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= ((0.00015, 0.99985) \times (1.0, 0.0)) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= (0.00015, 0.0, 0.99985, 0.0) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= (0.00015, 0.99985)
\end{aligned}$$

on places G^π_6 and G^π_8, respectively.

After that, transitions $f^0_{G_3}$ and $f^0_{G_4}$ are enabled and fire:

$$\begin{aligned}
\pi(G_3) &= (\pi(G_5) \times \pi(P_2) \times \pi(G_6)) \cdot CPM^0_{G_3} \\
&= ((0.10868890, 0.89131110) \times (0.0025, 0.0075) \times (0.00015, 0.99985)) \\
&\quad \cdot \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \\
&= (0.11105054, 0.88894946)
\end{aligned}$$

Due to the symmetry, this tuple is placed on G^π_3 by firing of $f^0_{G_3}$ as well as on G^π_4 by firing of $f^0_{G_4}$.

Subsequently, $f^0_{G_2}$ fires, marking place G^π_2 with the tuple

$$\begin{aligned}\pi(G_2) &= (\pi(G_3) \times \pi(G_4)) \cdot CPM^0_{G_2} \\ &= ((0.11105054, 0.88894946) \times (0.11105054, 0.88894946)) \cdot CPM^0_{G_2} \\ &= ((0.01233222), (0.09871832), (0.09871832), (0.79023114)) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix} \\ &= (0.01233222, 0.98766778).\end{aligned}$$

By that, $f^0_{G_1}$ can fire putting

$$\begin{aligned}\pi(G_1) &= (\pi(G_2) \times \pi(N)) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} \\ &= \ldots = (0.01234210, 0.98765790)\end{aligned}$$

on place G^π_1, which is removed by π_{G_1}, thus completing the **0**-reproduction.

As all likelihoods are neutral, the belief of G_1 with M_3 being instantiated as faulty is

$$BEL_{M_3=1}(G_1) = (0.01234210, 0.98765790).$$

Instantiation: Functional Global Memory

Similarly, the simulation of the π-invariant is executed with M_3 (M^1_3, M^2_3) being instantiated for 0, that means $\pi(M_3) = \pi(M^1_3) = \pi(M^2_3) = (0.0, 1.0)$. This results in

$$BEL_{M_3=0}(G_1) = (0.01231250, 0.98768750)$$

being the belief for G_1 if M_3 is operational.

Altogether, by weighting these beliefs with the actual probability of M_3 (M^1_3, M^2_3) and summing up,

$$\begin{aligned}BEL(G_1) &= 0.00015 \cdot BEL_{M_3=1}(G_1) + 0.99985 \cdot BEL_{M_3=0}(G_1) \\ &= 0.00015 \cdot (0.01234210, 0.98765790) \\ &\quad + 0.99985 \cdot (0.01231250, 0.98768750) \\ &\approx (0.01231250, 0.98768750)\end{aligned}$$

is the overall probability for a faulty system. This value coincides with the values calculated by the binary decision diagram method (see Example 2.14) and the Bayesian network tool (see Example 2.15) with respect to the accuracy of the result display. □

Until now, this approach does not add any benefit compared to the fault tree approach besides the fact that the t-invariants structure propagations and influences, thus providing a reasonable representation of causality. In [PorBob99, Bobbio*99] it is described how using Bayesian networks to represent fault tree analysis can be enhanced by taking advantage of the higher expressiveness of Bayesian networks. This applies to the probability propagation net representation of fault trees, too, which will be shown in the next section.

5.2. Adding Expressiveness by Bayesian Network Methods

Static fault trees are somehow "binary": Their variables can either be `true` or `false` and the gates are binary as well. In contrast to that, Bayesian networks are more flexible. Especially expressing causality and dependability can be fine-tuned by adjusting the parameters of the conditional probability tables.

This can be of interest if a fault tree abstracts from certain very unlikely interrelationships. Consider a system relying on two redundant components. If at least one component is functional the system works. Yet, there may be a cable connecting the components which is not modeled in the fault tree and if this cable fails, the whole system fails although the two redundant components work. This could of course be modeled in the fault tree but real-life examples usually are governed by some level of granularity and abstraction.

By adjusting the parameters of the Bayesian network conditional probability tables and thus slightly adapting the "gate functionality" (for example the *noisy or*-gate, see [HecBre96]), these unlikely interrelationships can easily be integrated into the model. There are many approaches to quantify the parameters which is referred to as an aspect of "learning" in the context of Bayesian networks (cf. [MeeHec97, Neap03]).

Not only the modification of gate functions may be of interest but also the evidence propagation and the concept of situation in a Bayesian network. It is possible to eval-

uate different scenarios of failure, for instance certain components can be assumed as not functional and the failure probability for the whole system can then be calculated by message propagation. By that, different constellations can be evaluated by means of the Bayesian network propagation technique and for example critical components or dependencies can be detected.

The Petri net representation of fault trees (as probability propagation nets corresponding to the respective Bayesian networks) can add transparency and a clear structure, as mentioned above. This will be shown by continuing the multiprocessor example.

Example 5.3 (Multiprocessor System: Evidence Scenario)
Assume it has been detected that the first disk of the first subsystem is faulty, that means $P(D_{11}) = \pi(D_{11}) = \lambda(D_{11}) = (1.0, 0.0)$. Simulation of the π-invariant of the probability propagation net shown in Figure 5.2 with respect to the conditioning of variable M_3 results in

$$BEL'(G_1) = (0.037, 0.963).$$

Although the disk has a mirrored backup (D_{12}) and the first subsystem itself has a backup (namely the second subsystem), a failure of the first disk increases the risk of a faulty system by a factor of three.

Now, additionally the second disk of the first subsystem D_{12} fails. A new simulation of the π-invariant yields

$$BEL''(G_1) = (0.111, 0.889),$$

that means the risk of a faulty system is again increased by a factor of three.

Other combinations can be evaluated similarly. By that, different scenarios can be analyzed by means of evidence propagation in Bayesian networks. As mentioned before, the probability propagation net representation thereby helps to make the propagations and influences transparent. This becomes even clearer when likelihoods are propagated.

Assume for example that the system is known to have failed but no other observations have been made. In this case G_1 has to be instantiated for $\pi(G_1) = \lambda(G_1) =$

$(1.0, 0.0)$ and all λ-invariants have to be simulated to update the λ-values of the input variables and thus the corresponding beliefs. As an example, the reproduction of the empty marking in the $\lambda(M_1)$-invariant is shown in a compacted way:

First, M_3 is instantiated for 1. λ_{G_1} and π_N fire and put $(1.0, 0.0)$ on place G_1^λ and $(0.00001, 0.99999)$ on place N^π. Now, $f_{G_1}^1$ fires and puts $\lambda_{G_1}(G_2) = (1.0, 0.00001)$ on place G_2^λ. For transition $f_{G_2}^2$ to be enabled, G_3^π must be marked. $\pi(G_3) = (0.11105054, 0.88894946)$ can be calculated as shown at the initialization.

Now, $f_{G_2}^2$ fires putting

$$\lambda_{G_2}(G_4) = (\lambda(G_2) \times \pi(G_3)) \cdot \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 1.0 \\ 1.0 & 1.0 \end{pmatrix} = (0.11105943, 0.00001)$$

on place G_4^λ.

After generating $\pi(P_1) = (0.0025, 0.9975)$ and $\pi(G_7) = (0.1086889, 0.891311)$ (see initialization), $f_{G_4}^1$ fires and puts

$$\lambda_{G_4}(G_8) = (\lambda(G_4) \times \pi(P_1) \times \pi(G_7)) \cdot \begin{pmatrix} 1.0 & 1.0 \\ 1.0 & 1.0 \\ 1.0 & 1.0 \\ 1.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \\ 0.0 & 1.0 \end{pmatrix} = (0.11105943, 0.80266019)$$

on place G_8^λ.

Now, $\pi_{M_3^1}$ fires thus enabling $f_{G_8}^2$ which in turn fires and puts

$$(0.11105943, 0.80266019)$$

on place M_1^λ, which is removed by λ_{M_1}, thus completing the **0**-reproduction.

The whole propagation has to be repeated with M_3 being instantiated for 0 (that means the global memory is functional). The resulting likelihood of M_1 is (0.80276397, 0.80276397). Note that the π-value for M_1 still is $(0.00015, 0.99985)$. The current weight with respect to the conditioning of the Bayesian network is calculated as

$$\begin{aligned} BEL'''(M_3) &= BEL'''(M_3 \mid G_1 = 1) = \alpha \cdot P(G_1 = 1 \mid M_3) \circ P(M_3) \\ &= \alpha \cdot (0.01234210 \cdot 0.00015, 0.01231250 \cdot 0.99985) \\ &= \alpha \cdot (0.00000185, 0.01231065) = (0.00015025, 0.99984975). \end{aligned}$$

After simulating the other t-invariants, the following beliefs are calculated:

$$\begin{aligned} BEL'''(D_{ij}) &= (0.984, 0.016), \quad i,j \in \{1,2\} \\ BEL'''(P_1) &= BEL'''(P_2) = (0.023, 0.977) \\ BEL'''(M_k) &= (0.0002, 0.9998), \quad k \in \{1,2,3\} \\ BEL'''(N) &= (0.001, 0.999). \end{aligned}$$

Note that the beliefs partially changed dramatically. The belief in a faulty bus (N) even increased by the factor of one hundred. But in relation to the other variables, especially the disk failure probability, it still is very small. A technical engineer may conclude that a running disk array seems to be the most critical component in the system, because of the high failure probability given that the system is faulty. In order to optimize the reliability of the whole system, first the reliability of the disk arrays should be increased. □

In conclusion, the evidence propagation in Bayesian networks can be used to detect critical components and thus to improve the system's reliability.

Concluding Remarks

Besides using instantiations of likelihoods and probabilities to evaluate different scenarios, the interpretation of likelihoods as postulations can contribute to the failure analysis of technical systems, too. Especially instantiating the top event for the `false`-value, thus expressing that the system is operational, can help to check for absolute essential components of the system. If a component X is vital for the system to work, the likelihood propagation would result in $X = (0.0, 1.0)$ (or similar pairs in case the likelihood is not normalized), that means X must not be faulty. Moreover, for variables which are not absolute essential but which contribute to an operational system, the corresponding likelihoods change thus indicating a "quantity of contribution". The tendency of the induced changes of each input variable can be an indicator which components should be optimized best to improve the reliability of the system.

Since this interpretation of likelihoods has similar facets compared to evaluating scenarios, an example is left out at this place. Basically, the two notions of likelihood differ in the direction of the assumptions. Postulating a functional system and then

looking at the effects on the input variables usually is part of the first interpretation, whereas instantiating variables of input components and checking the effects on the root event (that means the output variable) or intermediate gates may rather be considered as evaluation of scenarios.

At this point, it should be repeated that the probability propagation nets do not add functionality compared to the Bayesian network representation of fault trees. Yet, by structuring the propagation paths and integrating the message flows into the graph structure, the advantages of probability propagation nets mentioned for example in chapter 4 can be completely transferred to fault tree analysis.

On top of that, since other kinds of Petri nets are used to model or to control technical systems or processes (for example [HumFen05]), it is conceivable to combine them with probability propagation nets. Depending on the values of some set of input variables, for example different decisions could be triggered in the controlling part of the Petri net. Hence, probability propagation nets offer possibilities to couple probabilistic issues with technical processes in one homogeneous representation.

6. Conclusion and Outlook

Summary of Research Results

This thesis introduced a new class of Petri nets, namely probability propagation nets, which are suited to model probabilistic propagations on different levels. Probabilistic Horn abduction can be represented by low-level probability propagation nets, which allow straight-forward analysis and diagnosis by exploiting structural properties of the underlying p/t-nets.

A first folding of these Petri nets leads to higher-level probability propagation nets. Their structure is compacted transferring the complexity into the edge labels and the markings. Hence, pairs instead of one-tuples flow through the Petri net. Nevertheless, the central means of structural analysis – namely t-invariants – are calculated on the p/t-net level.

The higher-level probability propagation nets have proven to be suited to represent propagations of π-messages in Bayesian networks. A second extension, which adds likelihood propagations, leads to high-level probability propagation nets which are capable of representing the complete message propagation process of Bayesian networks. On the one side, these nets are more complex than the underlying Bayesian networks, which may be considered as a disadvantage. On the other side, the probability propagation nets reveal the true complexity of Bayesian message propagations and thus make the propagation processes in Bayesian networks transparent. Again, structural analysis is done on the p/t-net level, whereby the t-invariants are an excellent means for structuring the message flows, because each invariant exactly corresponds to one propagation path in the Bayesian network. By that, probability propagation nets show how probabilities and evidences influence each other in Bayesian networks, which is a precise representation of causality that eases the understanding of models and may give hints for diagnosis and optimization.

A precondition for the transformation of Bayesian networks into probability propa-

gation nets is that the Bayesian networks are singly-connected which means that they do not contain any loop. Since this is fundamental for correct propagation results, different methods to eliminate loops have been developed. In contrast to many existing implementations, the conditioning method was used in this thesis, because the resulting loop-free Bayesian networks are structurally closer to the original networks than the graphs calculated for example by the popular junction tree method.

As a consequence of using the conditioning method, a further folding of high-level probability propagation nets allows simultaneous evaluation of all instantiated conditioned versions of the corresponding Bayesian network. This is achieved by transferring the structural complexity once more into the edge labels and the markings. Originally, for each instantiation of conditioned variables, a single instance of the Bayesian network had to be evaluated. By the folding, these single instances are molten together and matrices instead of vectors flow through the probability propagation net. Thereby, a row of a matrix corresponds to the vector of the respective conditioned Bayesian network instance. Again, the folding does not affect the simplicity of structural analysis which is done on the p/t-net level as before.

Altogether, probability propagation nets are well-suited for representing Bayesian network message propagations, because

- they integrate the dynamics into the graph structure,
- they reveal the true complexity of message propagations, thus increasing the transparency of the Bayesian propagation algorithms,
- they clearly structure the propagation flows in Bayesian networks by means of t-invariants,
- and they precisely represent causality.

Once more it has to be pointed out that probability propagation nets are not intended to improve Bayesian networks but they are an adequate representation of Bayesian networks which eases their understanding, in the first place.

In addition to that, it was described how fault trees can be transformed into probability propagation nets via the corresponding Bayesian networks such that the above mentioned advantages of the Petri net representations are transferred to fault tree analysis, too. The expressiveness of Bayesian networks thereby allows sophisticated

possibilities regarding the analysis and diagnosis of technical systems. To look forward, the idea of coupling probabilistic Petri nets with established Petri nets modeling or controlling technical systems or processes was mentioned. This leads to the topics which could not be treated in this thesis but which should be addressed to in future research.

Future Work

First of all, a tool support for modeling, simulating and analyzing probability propagation nets including a generator creating the corresponding probability propagation nets out of Bayesian networks would be desirable. A supporting tool is essential for examples which are more complex and closer to real-life than the examples given in this thesis. Especially in the context of biological and technical applications it would be interesting to evaluate the potential of probability propagation nets. It is planned to extend the tool *NeMo* [PhPiRa05, Phil*06, Pinl*07] with support for probability propagation nets and a corresponding generator.

Additionally, it would be interesting to see if the learning in Bayesian networks (cf. [Heck99, Neap03]) applied to probability propagation nets can be profitable for Petri nets in general. Perhaps some aspects of structure and parameter learning apply to other kinds of Petri nets, too. This could be of interest for example in technical applications to adjust Petri nets based on collected data.

Another interesting question is if there is a kind of duality (cf. [Laut03a, Müller04]) which can be found in probability propagation nets. First research in this context indicates that there is a non-trivial duality between π- and λ-structures. This topic will be addressed to in a forthcoming paper.

There is also an early but quite promising approach in extending probability propagation nets to represent mass function propagations according to the Dempster-Shafer theory [Demp68, Shafer76, Spie86, Pearl88, Pearl90, YaKaFe94]. The Dempster-Shafer theory differentiates probability and plausibility and thus allows modeling in a finer way. It is subject to future research if a probability propagation net representation for mass functions can be found, which increases transparency the same way as applied to Bayesian networks. A request for a research project in this context is pending, at the moment.

To sum it up, probability propagation nets may be considered a reasonable extension of the Petri net world, because they allow for probabilities to be directly

propagated in the Petri net representation, they have proven to be useful at representing established approaches such as fault trees, probabilistic Horn abduction and Bayesian networks, and a high potential with respect to applications in biological and technical contexts as well as Dempster-Shafer theory can be assumed.

A. Bayesian Network of the Multiprocessor System

```
<?xml version="1.0" encoding="US-ASCII"?>
<!--
Bayesian network in XMLBIF v0.3 (BayesNet Interchange Format)
Produced by BNJ 3.0 (http://bndev.sourceforge.net/
-->
    <!-- DTD for the XMLBIF 0.3 format -->
<!DOCTYPE BIF [
<!ELEMENT BIF ( NETWORK )*>
<!ATTLIST BIF VERSION CDATA #REQUIRED>
<!ELEMENT NETWORK ( NAME, ( PROPERTY | VARIABLE |
  DEFINITION )* )>
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT VARIABLE ( NAME, ( OUTCOME |  PROPERTY )* ) >
  <!ATTLIST VARIABLE TYPE (nature|decision|utility) "nature">
<!ELEMENT OUTCOME (#PCDATA)>
<!ELEMENT DEFINITION ( FOR | GIVEN | TABLE | PROPERTY )* >
<!ELEMENT FOR (#PCDATA)>
<!ELEMENT GIVEN (#PCDATA)>
<!ELEMENT TABLE (#PCDATA)>
<!ELEMENT PROPERTY (#PCDATA)>
]>
<BIF VERSION="0.3">
<NETWORK>
<NAME>bn</NAME>
  <VARIABLE TYPE="nature">
```

```
    <NAME>D21</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (50,50)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>D22</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (150,50)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>P2</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (250,50)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>M2</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (350,50)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>M3</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (450,50)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>M1</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (550,50)</PROPERTY>
```

```
</VARIABLE>
<VARIABLE TYPE="nature">
  <NAME>P1</NAME>
  <OUTCOME>True</OUTCOME>
  <OUTCOME>False</OUTCOME>
  <PROPERTY>position = (650,50)</PROPERTY>
</VARIABLE>
<VARIABLE TYPE="nature">
  <NAME>D12</NAME>
  <OUTCOME>True</OUTCOME>
  <OUTCOME>False</OUTCOME>
  <PROPERTY>position = (750,50)</PROPERTY>
</VARIABLE>
<VARIABLE TYPE="nature">
  <NAME>D11</NAME>
  <OUTCOME>True</OUTCOME>
  <OUTCOME>False</OUTCOME>
  <PROPERTY>position = (850,50)</PROPERTY>
</VARIABLE>
<VARIABLE TYPE="nature">
  <NAME>N</NAME>
  <OUTCOME>True</OUTCOME>
  <OUTCOME>False</OUTCOME>
  <PROPERTY>position = (950,50)</PROPERTY>
</VARIABLE>
<VARIABLE TYPE="nature">
  <NAME>G5</NAME>
  <OUTCOME>True</OUTCOME>
  <OUTCOME>False</OUTCOME>
  <PROPERTY>position = (100,150)</PROPERTY>
</VARIABLE>
<VARIABLE TYPE="nature">
  <NAME>G6</NAME>
  <OUTCOME>True</OUTCOME>
```

```
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (400,150)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>G8</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (500,150)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>G7</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (800,150)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>G3</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (250,250)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>G4</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (650,250)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
    <NAME>G2</NAME>
    <OUTCOME>True</OUTCOME>
    <OUTCOME>False</OUTCOME>
    <PROPERTY>position = (450,350)</PROPERTY>
  </VARIABLE>
  <VARIABLE TYPE="nature">
```

```
  <NAME>G1</NAME>
  <OUTCOME>True</OUTCOME>
  <OUTCOME>False</OUTCOME>
  <PROPERTY>position = (650,450)</PROPERTY>
</VARIABLE>
<DEFINITION>
  <FOR>D21</FOR>
  <TABLE>0.32968 0.67032     </TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>D22</FOR>
  <TABLE>0.32968 0.67032     </TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>P2</FOR>
  <TABLE>0.0025 0.9975     </TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>M2</FOR>
  <TABLE>1.5E-4 0.99985     </TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>M3</FOR>
  <TABLE>1.5E-4 0.99985     </TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>M1</FOR>
  <TABLE>1.5E-4 0.99985     </TABLE>
</DEFINITION>
<DEFINITION>
  <FOR>P1</FOR>
  <TABLE>0.0025 0.9975     </TABLE>
</DEFINITION>
<DEFINITION>
```

```
    <FOR>D12</FOR>
    <TABLE>0.32968 0.67032     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>D11</FOR>
    <TABLE>0.32968 0.67032     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>N</FOR>
    <TABLE>1.0E-5 0.99999     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G5</FOR>
    <GIVEN>D21</GIVEN>
    <GIVEN>D22</GIVEN>
    <TABLE>1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G6</FOR>
    <GIVEN>M2</GIVEN>
    <GIVEN>M3</GIVEN>
    <TABLE>1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G8</FOR>
    <GIVEN>M3</GIVEN>
    <GIVEN>M1</GIVEN>
    <TABLE>1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G7</FOR>
    <GIVEN>D12</GIVEN>
    <GIVEN>D11</GIVEN>
    <TABLE>1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0     </TABLE>
```

```
  </DEFINITION>
  <DEFINITION>
    <FOR>G3</FOR>
    <GIVEN>P2</GIVEN>
    <GIVEN>G5</GIVEN>
    <GIVEN>G6</GIVEN>
    <TABLE>
      1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 1.0
    </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G4</FOR>
    <GIVEN>P1</GIVEN>
    <GIVEN>G8</GIVEN>
    <GIVEN>G7</GIVEN>
    <TABLE>
      1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0 1.0
    </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G2</FOR>
    <GIVEN>G3</GIVEN>
    <GIVEN>G4</GIVEN>
    <TABLE>1.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0     </TABLE>
  </DEFINITION>
  <DEFINITION>
    <FOR>G1</FOR>
    <GIVEN>N</GIVEN>
    <GIVEN>G2</GIVEN>
    <TABLE>1.0 0.0 1.0 0.0 1.0 0.0 0.0 1.0     </TABLE>
  </DEFINITION>
</NETWORK>
</BIF>
```

B. Proof: Representation of Message Propagation in Probability Propagation Nets

Theorem B.1 The introduced high-level probability propagation nets, built according to Definition 4.8, completely represent the message propagation according to the Bayesian network propagation algorithm given in section 2.3.2.

Proof (sketch)
To prove this theorem, the propagation formulas (cf. [Neap90]) are inspected one by one and it is shown how the respective formula is represented in the probability propagation net.

Operative Formulas

1. *Bayesian network operative formula:*

   If B is a child of A, B has k possible values, A has m possible values, and B has one other parent D, with n possible values, then for $1 \leq j \leq m$ the λ message from B to A is given by

   $$\lambda_B(a_j) = \sum_{p=1}^{n} \pi_B(d_p)\big(\sum_{i=1}^{k} P(b_i \mid a_j, d_p)\lambda(b_i)\big).$$

   This kind of structure is a Bayesian network join (see Definition 4.1). In the probability propagation net, this operational formula is represented by the firing

of a functional transition f_B^l with $l \geq 1$ and $f_B^{l\bullet} = A^\lambda$. The arc label thereby is

$$
\begin{aligned}
\lambda_B(A) &= (\lambda(B) \times \pi_B(D)) \cdot CPM_B^l \\
&= (\lambda(b_1) \cdot \pi_B(d_1), \ldots, \lambda(b_1) \cdot \pi_B(d_n), \ldots, \\
&\quad \lambda(b_k) \cdot \pi_B(d_1), \ldots, \lambda(b_k) \cdot \pi_B(d_n)) \cdot CPM_B^l \\
&= (\lambda(b_1) \cdot \pi_B(d_1) \cdot P(b_1 \mid a_1, d_1) + \ldots \\
&\quad + \lambda(b_1) \cdot \pi_B(d_n) \cdot P(b_1 \mid a_1, d_n) + \ldots \\
&\quad + \lambda(b_k) \cdot \pi_B(d_1) \cdot P(b_k \mid a_1, d_1) + \ldots \\
&\quad + \lambda(b_k) \cdot \pi_B(d_n) \cdot P(b_k \mid a_1, d_n), \ldots, \\
&\quad \lambda(b_1) \cdot \pi_B(d_1) \cdot P(b_1 \mid a_m, d_1) + \ldots \\
&\quad + \lambda(b_1) \cdot \pi_B(d_n) \cdot P(b_1 \mid a_m, d_n) + \ldots \\
&\quad + \lambda(b_k) \cdot \pi_B(d_1) \cdot P(b_k \mid a_m, d_1) + \ldots \\
&\quad + \lambda(b_k) \cdot \pi_B(d_n) \cdot P(b_k \mid a_m, d_n)) \\
\Rightarrow \lambda_B(a_j) &= (\lambda(b_1) \cdot \pi_B(d_1) \cdot P(b_1 \mid a_j, d_1) + \ldots \\
&\quad + \lambda(b_k) \cdot \pi_B(d_n) \cdot P(b_k \mid a_j, d_n)) \\
&= \sum_{p=1}^{n} \pi_B(d_p) \cdot \big(\lambda(b_1) \cdot P(b_1 \mid a_j, d_p) + \ldots \\
&\quad + \lambda(b_k) \cdot P(b_k \mid a_j, d_p)\big) \\
&= \sum_{p=1}^{n} \pi_B(d_p) \cdot \left(\sum_{i=1}^{k} \lambda(b_i) \cdot P(b_i \mid a_j, d_p) \right),
\end{aligned}
$$

which is equal to the result of the above operative formula, as the input variables are bound analogously.

2. *Bayesian network operative formula:*

   If B is a child of A and A has m possible values, then for $1 \leq j \leq m$ the π message from A to B is given by

$$
\pi_B(a_j) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_j \\ 0 & \text{if } A \text{ is instantiated, but not for } a_j \\ \frac{P'(a_j)}{\lambda_B(a_j)} & \text{if } A \text{ is not instantiated,} \end{cases}
$$

where $P'(a_j)$ is defined to be the current conditional probability of a_j based on the variables thus far instantiated.

The corresponding arc label in the probability propagation net simply is $\pi_B(A)$, which binds the token on A^π or, in case of a split, on A^π_B. If A is not instantiated, A's current π-value in a normalized form is used (see Remark 4.6). With respect to the definition of the conditional probability (namely $P'(a_j) = \alpha\lambda_B(a_j)\pi_B(a_j)$), this π-value is equal to $\frac{P'(a_j)}{\lambda_B(a_j)}$.

If A is instantiated for a_j, according to section 4.3 the edge labels are changed to constant vectors. Their entries are all 0.0 *except the entry for a_j which is* 1.0. *This is in accordance with the above operative formula.*

3. *Bayesian network operative formula:*

   If B is a variable with k possible values, $s(B)$ is the set of B's children, then for $1 \leq i \leq k$ the λ value of B is given by

$$\lambda(b_i) = \begin{cases} \prod_{C \in s(B)} \lambda_C(b_i) & \text{if } B \text{ is not instantiated} \\ 1 & \text{if } B \text{ is instantiated for } b_i \\ 0 & \text{if } B \text{ is instantiated, but not for } b_i. \end{cases}$$

This kind of structure is a Bayesian network split (see Definition 4.2). Let n be the number of B's children, that means $n = |s(B)|$ and let be $s(B) = \{C_1, \ldots, C_n\}$. According to Definition 4.7 – which applies if B has not been instantiated – the edge label calculating $\lambda(B)$ is defined as follows:

$$\begin{aligned} \lambda(B) &= \lambda_{C_1}(B) \circ \cdots \circ \lambda_{C_n}(B) \\ &= \left(\prod_{j=1}^{n} \lambda_{C_j}(b_1), \ldots, \prod_{j=1}^{n} \lambda_{C_j}(b_m) \right) \\ \Rightarrow \lambda(b_i) &= \prod_{j=1}^{n} \lambda_{C_j}(b_i) = \prod_{C \in s(B)} \lambda_C(b_i). \end{aligned}$$

If B is instantiated for b_i, according to section 4.3 the corresponding λ-vector is a vector whose entries are 0.0 *except the entry corresponding to the instantiated value, which is* 1.0. *So, the λ-value of B in the probability propagation net is*

equal to the one calculated by operative formula 3.

4. Bayesian network operative formula:

If B is a variable with k possible values and exactly two parents, A and D, A has m possible values, and D has n possible values, then for $1 \leq i \leq k$ the π value of B is given by

$$\pi(b_i) = \sum_{j=1}^{m} \sum_{p=1}^{n} P(b_i \mid a_j, d_p)\pi_B(a_j)\pi_B(d_p).$$

This kind of structure is a Bayesian network join (see Definition 4.1). In the probability propagation net, this operational formula is represented by the firing of the join's first functional transition f_B^0. The arc label thereby is $\pi(B) = (\pi(A) \times \pi(D)) \cdot CPM_B^0$. Now it will be shown that the calculation induced by the firing of f_B^0 yields the same result as the above operational formula (the variables and respective numbers of possible values are given above):

$$\begin{aligned}
\pi(B) &= (\pi_B(A) \times \pi_B(D)) \cdot CPM_B^0 \\
&= (\pi_B(a_1) \cdot \pi_B(d_1), \ldots, \pi_B(a_1) \cdot \pi_B(d_n), \ldots, \\
&\quad \pi_B(a_m) \cdot \pi_B(d_1), \ldots, \pi_B(a_m) \cdot \pi_B(d_n)) \cdot CPM_B^0 \\
&= (\pi_B(a_1) \cdot \pi_B(d_1) \cdot P(b_1 \mid a_1, d_1) + \ldots \\
&\quad + \pi_B(a_1) \cdot \pi_B(d_n) \cdot P(b_1 \mid a_1, d_n) + \ldots \\
&\quad + \pi_B(a_m) \cdot \pi_B(d_1) \cdot P(b_1 \mid a_m, d_1) + \ldots \\
&\quad + \pi_B(a_m) \cdot \pi_B(d_n) \cdot P(b_1 \mid a_m, d_n), \ldots, \\
&\quad \pi_B(a_1) \cdot \pi_B(d_1) \cdot P(b_k \mid a_1, d_1) + \ldots \\
&\quad + \pi_B(a_1) \cdot \pi_B(d_n) \cdot P(b_k \mid a_1, d_n) + \ldots \\
&\quad + \pi_B(a_m) \cdot \pi_B(d_1) \cdot P(b_k \mid a_m, d_1) + \ldots \\
&\quad + \pi_B(a_m) \cdot \pi_B(d_n) \cdot P(b_k \mid a_m, d_n))
\end{aligned}$$

$$
\begin{aligned}
\Rightarrow \pi(b_i) = &(\pi_B(a_1) \cdot \pi_B(d_1) \cdot P(b_i \mid a_1, d_1) + \ldots \\
&+ \pi_B(a_1) \cdot \pi_B(d_n) \cdot P(b_i \mid a_1, d_n) + \ldots \\
&+ \pi_B(a_m) \cdot \pi_B(d_1) \cdot P(b_i \mid a_m, d_1) + \ldots \\
&+ \pi_B(a_m) \cdot \pi_B(d_n) \cdot P(b_i \mid a_m, d_n)) \\
= &\sum_{j=1}^{m} \sum_{p=1}^{n} \pi_B(a_j) \cdot \pi_B(d_p) \cdot P(b_i \mid a_j, d_p).
\end{aligned}
$$

Since the variables $\pi_B(A), \pi_B(D)$ are bound at the incoming edges and the current π-values of the corresponding nodes are assigned to these variables, the resulting tuple after firing of f_B^0 is equal to the π-message calculated by the above operative formula.

5. *Bayesian network operative formula:*

   If B is a variable with k possible values, then for $1 \leq i \leq k$, $P'(B_i)$, the conditional probability of b_i based on the variables thus far instantiated, is given by

   $$P'(b_i) = \alpha\lambda(b_i)\pi(b_i).$$

The conditional probability, which is also called the belief, is not represented in the structure of the probability propagation net but this formula is used to calculate the beliefs based on the current λ- and π-messages as described in Definition 2.22 and in section 4.2.

Initialization

1. *Bayesian network initialization rule:*

   Set all λ values, λ messages and π messages to 1.

The initialization of λ-values is done by setting the arc labels of outgoing boundary λ-transitions to the current λ-values of the corresponding variables as described in Definition 4.8. Since at the initialization phase no evidence is given, all λ-values are neutral (that means $(1.0, \ldots, 1.0)$). Messages are in principle represented by tokens flowing through the net representations of the t-invariants.

Until now, only the structure is set and no simulation is started in the probability propagation net.

2. *Bayesian network initialization rule:*

   For all roots A, if A has m possible values, then for $1 \leq j \leq m$, set $\pi(a_j) = P(a_j)$.

   In the probability propagation net, this is also represented by the arc labels. In this case the labels of the edges (π_A, A^π) are initialized with the respective prior probabilities (see Definition 4.8).

3. *Bayesian network initialization rule:*

   For all roots A for all children B of A do
   Post a new π message to B using operative formula 2. (A propagation flow will then begin due to updating procedure C.)

   This rule is represented by starting the reproduction of the empty marking in the π-invariants. This is always the first dynamical step of the initialization process in probability propagation nets as mentioned in section 4.2.

Updating

When a variable is instantiated, or a λ or π message is received by a variable, one of the following updating procedures is used:

Instantiation in probability propagation nets is done by changing the corresponding arc labels generating the π- and λ-values. Message reception is represented by the marking which changes when transitions fire. In turn, this may enable other transitions. So, receiving a message is represented by the enabling of a transition under the respective marking (that means enabling by tokens or vectors on the input places), sending a message is represented by the firing of the respective transition, which consumes the "old" tokens and generates the corresponding new tokens on the output places. The details are described below.

- *Updating procedure A.*

  If a variable B is instantiated for b_j, then

  1. Set $P'(b_j) = 1$ and for $i \neq j$, set $P'(b_i) = 0$;

2. Compute $\lambda(B)$ using operative formula 3;
3. Post new λ messages to all B's parents using operative formula 1;
4. Post new π messages to all B's children using operative formula 2.

As mentioned above, the instantiation of variables in the probability propagation net is done by setting the arc labels to the respective values. In this case, the arcs $\{(B^{\pi}, t \mid t \in B^{\pi\bullet})\}$ *and* $\{(t, B^{\lambda} \mid t \in {}^{\bullet}B^{\lambda})\}$ *are set to the vector which entries are* 0.0 *except the entry of* b_j *which is* 1.0 *as described in section 4.3.*

- *Updating procedure B.*

  If a variable B receives a new λ message from one of its children, then if B is not already instantiated,

  1. Compute the new value of $\lambda(B)$ using operative formula 3;
  2. Compute the new value of $P'(B)$ using operative formula 5;
  3. Post λ messages to all B's parents using operative formula 1;
  4. Post new π messages to B's other children using operative formula 2.

  The new value of $\lambda(B)$ *is computed by the arc label as described above. The new belief is not represented in the Petri net itself. It has to be calculated with the help of the current* π*- and* λ*-values according to operative formula 5. The propagation of the new* λ*- and* π*-messages is again done by simulating the different minimal t-invariants.*

- *Updating procedure C.*

  If a variable B receives a new π message from a parent, then if B is not already instantiated,

  1. Compute the new value of $\pi(B)$ using operative formula 4;
  2. Compute the new value of $P'(B)$ using operative formula 5;
  3. Post new π messages to all B's children using operative formula 2;

  else if $\lambda(B) \neq (1, 1, \ldots, 1)$, then

  4. Post new λ messages to B's other parents using operative formula 1.

This case is similar to the propagation of λ-messages described in the previous step. Once more, the simulation of t-invariants provides for the propagation to be executed.

In conclusion, the properties of the Bayesian propagation algorithm are completely represented in the corresponding probability propagation net. Since to each edge in the Bayesian network the π- as well as the λ-direction in the probability propagation exists, the t-invariants completely cover the Bayesian network's propagation paths. ■

Bibliography

[Alex00] **Carol Alexander**. *Bayesian Methods for Measuring Operational Risk*. Research Paper, ISMA Centre (The Business School for Financial Markets), 2000.

[Bayes63] **Thomas Bayes**. *An Essay towards solving a Problem in the Doctrine of Chances*. Philosophical Transactions of the Royal Society of London, 53:370–418, 1763.

[Bobbio*99] **Andrea Bobbio, Luigi Portinale, Michele Minichino and Ester Ciancamerla**. *Comparing Fault Trees and Bayesian Networks for Dependability Analysis*. In *SAFECOMP '99: Proceedings of the 18th International Conference on Computer Computer Safety, Reliability and Security*, pp. 310–322, London, UK, 1999. Springer-Verlag.

[Comm06] **International Electrotechnical Commission**. *IEC 61025: 2006 (Fault tree analysis)*. Standard, 2006.

[Cooper84] **Gregory F. Cooper**. *NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge*. PhD thesis, Department of Computer Science, Stanford University, Stanford, California, USA, 1984.

[Demp68] **Arthur P. Dempster**. *A generalization of Bayesian inference*. Journal of the Royal Statistical Society, Series B, 30:205–247, 1968.

[DoyDug95] **Stacy A. Doyle and Joanne Bechta Dugan**. *Dependability Assessment using Binary Decision Diagrams (BDDs)*. In *The Twenty-Fifth International Symposium on Fault-Tolerant Computing (FTCS)*, pp. 249–258. IEEE Computer Society, 1995.

[Dugan01] **Joanne Bechta Dugan**. *Reliability Analysis of Computer-based Systems Using Dynamic Fault Trees.* Tutorial presented to the Annual Reliability and Maintainability Symposium, 2001.

[DugTri89] **Joanne Bechta Dugan and Kishor S. Trivedi**. *Coverage Modeling for Dependability Analysis of Fault-Tolerant Systems.* IEEE Trans. Comput., 38(6):775–787, 1989.

[Frie*00] **Nir Friedman, Michal Linial, Iftach Nachman and Dana Pe'er**. *Using Bayesian networks to analyze expression data.* In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology, RECOMB 2000*, pp. 127–135, ACM, Tokyo, Japan, 2000.

[GenLau79] **Hartmann J. Genrich and Kurt Lautenbach**. *The Analysis of Distributed Systems by Means of Predicate/Transition-Nets.* Lecture Notes in Computer Science: Semantics of Concurrent Computation, 70:123–146, 1979.

[GenLau83] **Hartmann J. Genrich and Kurt Lautenbach**. *S-Invariance in Predicate/Transition Nets.* In **Anastasia Pagnoni and Grzegorz Rozenberg** (editors): *Application and Theory of Petri Nets — Selected Papers from the Third European Workshop on Application and Theory of Petri Nets, Varenna, Italy, September 27–30, 1982*, Volume 66 of *Informatik Fachberichte*, pp. 98–111. Springer, 1983.

[HecBre96] **David Heckerman and John S. Breese**. *Causal independence for probability assessment and inference using Bayesian networks.* IEEE Transactions on Systems, Man and Cybernetics, Part A, 26(6):826–831, 1996.

[Heck99] **David Heckerman**. *A Tutorial on Learning with Bayesian Networks.* Learning in Graphical Models, 1999.

[Hsu04] **William H. Hsu**. *Bayesian Network tools in Java (BNJ).* online: http://bnj.sourceforge.net/, 2004.

[HumFen05] **Thorsten Hummel and Wolfgang Fengler**. *Design of Embedded Control Systems Using Hybrid Petri Nets and Time Interval Petri Nets.*

In **Marian A. Adamski, Andrei Karatkevich and Marek Wegrzyn** (editors): *Design of Embedded Control Systems*, pp. 141–154, Springer, Berlin, Heidelberg, New York, 2005.

[Ireson88] **William G. Ireson**. *Handbook of Reliability Engineering and Management*. McGraw-Hill Professional, New York, 1988.

[Jensen81] **Kurt Jensen**. *Coloured Petri Nets and the Invariant-Method*. Theoretical Computer Science, 14:317–336, 1981.

[Jensen96] **Finn V. Jensen**. *Introduction to Bayesian Networks*. UCL Press, London, 1996.

[Kind02] **Ekkart Kindler**. *Petrinetze*. Lecture Notes for the 'Course on Petri Nets', University of Paderborn, Germany, 2002.

[LaPhPi06] **Kurt Lautenbach, Stephan Philippi and Alexander Pinl**. *Bayesian Networks and Petri Nets*. In *Proceedings of the workshop "Entwurf komplexer Automatisierungssysteme" (EKA) 2006*, Braunschweig, 2006.

[LauPin05] **Kurt Lautenbach and Alexander Pinl**. *Probability Propagation in Petri Nets*. Fachberichte Informatik 16–2005, Universität Koblenz-Landau, Universität Koblenz-Landau, Institut für Informatik, Universitätsstr. 1, D-56070 Koblenz, 2005.

[LauPin07] **Kurt Lautenbach and Alexander Pinl**. *Probability Propagation Nets*. Arbeitsberichte aus dem Fachbereich Informatik 20–2007, Universität Koblenz-Landau, Institut für Informatik, Universitätsstr. 1, D-56070 Koblenz, 2007.

[LauSpi88] **Steffen L. Lauritzen and David J. Spiegelhalter**. *Local computations with probabilities on graphical structures and their application to expert systems*. Journal of the Royal Statistical Society, 50(2):157–224, 1988.

[Laut02] **Kurt Lautenbach**. *Reproducibility of the Empty Marking*. In **Javier Esparza and Charles Lakos** (editors): *Applications and Theory of*

Petri Nets 2002, 23rd International Conference, ICATPN 2002, Adelaide, Australia, June 24–30, 2002, Proceedings, Volume 2360 of *Lecture Notes in Computer Science*, pp. 237–253. Springer, 2002.

[Laut03a] **Kurt Lautenbach**. *Duality of Marked Place/Transition Nets.* Fachberichte Informatik 18–03, Universität Koblenz, 2003.

[Laut03b] **Kurt Lautenbach**. *Logical Reasoning and Petri Nets.* In **Wil M. P. van der Aalst and Eike Best** (editors): *Applications and Theory of Petri Nets 2003, 24th International Conference, ICATPN 2003, Eindhoven, Netherlands, June 23–27, 2003, Proceedings*, Volume 2679 of *Lecture Notes in Computer Science*, pp. 276–295. Springer, 2003.

[MalTri95] **Manish Malhotra and Kishor S. Trivedi**. *Dependability Modeling Using Petri-Nets.* IEEE Transactions on Reliability, 44(3):428–440, 1995.

[MeeHec97] **Christopher Meek and David Heckerman**. *Structure and Parameter Learning for Causal Independence and Causal Interaction Models.* In **Dan Geiger and Prakash P. Shenoy** (editors): *UAI '97: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pp. 366–375. Morgan Kaufmann, 1997.

[Müller04] **Jörg Müller**. *Dualität und Analyse von Formalen Modellen — Prädikat/Transitions-Netze und ihr Bezug zur Linearen Algebra.* PhD thesis, Universität Koblenz-Landau, 2004.

[Neap90] **Richard E. Neapolitan**. *Probabilistic reasoning in expert systems: theory and algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 1990.

[Neap03] **Richard E. Neapolitan**. *Learning Bayesian Networks.* Prentice Hall, Upper Saddle River, NJ, 2003.

[Need*06] **Chris J. Needham, James R. Bradford, Andrew J. Bulpitt and David R. Westhead**. *Inference in Bayesian Networks.* Nature Biotechnology, 24(1):51–53, January 2006.

[Need*07] **Chris J. Needham, James R. Bradford, Andrew J. Bulpitt and David R. Westhead**. *A Primer on Learning in Bayesian Networks for Computational Biology.* PLoS Computational Biology, 3(8):1409–1416, August 2007.

[Pearl88] **Judea Pearl**. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[Pearl90] **Judea Pearl**. *Reasoning with belief functions: An analysis of compatibility.* International Journal of Approximate Reasoning, 4(5-6):363–389, 1990.

[PenReg86] **Yun Peng and James A. Reggia**. *Plausibility of Diagnostic Hypotheses: The Nature of Simplicity.* In *AAAI '86: Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 140–147, Philadelphia, 1986.

[Petri62] **Carl A. Petri**. *Kommunikation mit Automaten.* Technical Report, Schriften des Institutes für instrumentelle Mathematik, Bonn, 1962.

[PetSis06] **G. W. Peters and S. A. Sisson**. *Bayesian Inference, Monte Carlo Sampling and Operational Risk.* Research Paper, School of Mathematics and Statistics, University of New South Wales, Sydney, Australia, October 2006.

[Phil*06] **Stephan Philippi, Alexander Pinl, Jörg R. Müller and Roman Slovák**. *Towards tool support for the formally based analysis of safety-critical systems with Petri-Nets.* In **Daniel Moldt** (editor): *Bericht 276, 13. Workshop "Algorithmen und Werkzeuge für Petrinetze" (AWPN 2006)*, pp. 69–74, Hamburg, 2006.

[PhPiRa05] **Stephan Philippi, Alexander Pinl and Gerrit Rausch**. *A first View on a Generalised Modelling Toolkit for Graph-based Languages.* In **Karsten Schmidt and Christian Stahl** (editors): *12. Workshop "Algorithmen und Werkzeuge für Petrinetze" (AWPN 2005)*, pp. 25–30, Berlin, 2005.

[Pinl*07] **Alexander Pinl, Markus Pinl, Jan Poganski and Stephan Philippi**. *Aspekte der Modularität und Flexibilität in NeMo (ToMASEn).* In **Stephan Philippi and Alexander Pinl** (editors): *Proceedings des 14. Workshop Algorithmen und Werkzeuge für Petri-Netze*, pp. 14–19, Universität Koblenz-Landau, 2007.

[Poole93a] **David Poole**. *Logic programming, abduction and probability: a top-down anytime algorithm for estimating prior and posterior probabilities.* New Generation Computing, 11(3-4):377–400, 1993.

[Poole93b] **David Poole**. *Probabilistic Horn abduction and Bayesian networks.* Artificial Intelligence, 64(1):81–129, 1993.

[PorBob99] **Luigi Portinale and Andrea Bobbio**. *Bayesian Networks for Dependability Analysis: an Application to Digital Control Reliability.* In **Kathryn B. Laskey and Henri Prade** (editors): *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30-August 1, 1999*, pp. 551–558. Morgan Kaufmann, 1999.

[PorTor97] **Luigi Portinale and Pietro Torasso**. *A Comparative Analysis of Horn Models and Bayesian Networks for Diagnosis.* In **Maurizio Lenzerini** (editor): *Advances in Artificial Intelligence 1997, Rome, Italy*, Volume 1321 of *Lecture Notes in Computer Science*, pp. 254–265. Springer, 1997.

[RaArGh05] **Shilpa Ramamurthy, Harpreed Arora and Anirbid Ghosh**. *Operational Risk and Probabilistic Networks — An Application to Corporate Actions Processing.* Research Paper, Infosys Technologies Limited, November 2005.

[Robi03] **Gary Robinson**. *A Statistical Approach to the Spam Problem.* online: http://www.linuxjournal.com/article/6467, 2003.

[RolMor90] **Harold E. Roland and Brian Moriarty**. *System safety engineering and management.* Wiley, New York, 1990.

[Sahami*98] **Mehran Sahami, Susan Dumais, David Heckerman and Eric Horvitz**. *A Bayesian Approach to Filtering Junk E-Mail.* AAAI'98 Workshop on Learning for Text Categorization, 1998.

[Shafer76] **Glenn Shafer**. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, 1976.

[SheSha88] **Prakash P. Shenoy and Glenn Shafer**. *Axioms for probability and belief-function proagation.* In **Ross D. Shachter, Tod S. Levitt, Laveen N. Kanal and John F. Lemmer** (editors): *UAI '88: Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 169–198, 1988.

[SheWüt06] **P. V. Shevchenko and M. V. Wüthrich**. *The structural modelling of operational risk via Bayesian inference: combining loss data with expert opinions.* Journal of Operational Risk, 1(3):3–26, 2006.

[Spie85] **David J. Spiegelhalter**. *Probabilistic Reasoning in Predictive Expert Systems.* In **Laveen N. Kanal and John F. Lemmer** (editors): *UAI '85: Proceedings of the First Annual Conference on Uncertainty in Artificial Intelligence, Rome, New York, USA*, pp. 47–68. Elsevier, 1985.

[Spie86] **David J. Spiegelhalter**. *A statistical View of Uncertainty in Expert Systems.* Artificial Intelligence and Statistics, pp. 17–55, 1986.

[SpiLau90] **David J. Spiegelhalter and Steffen L. Lauritzen**. *Techniques for Bayesian analysis in expert systems.* Annals of Mathematics and Artificial Intelligence, 2:353–364, 1990.

[VanBil05] **Somsak Vanit-Anunchai and Jonathan Billington**. *Modelling Dynamic Influence Diagrams using Coloured Petri Nets.* In *Design, Analysis and Simulation of Distributed Systems Symposiumm, The Spring Multiconference, San Diego, USA, April 3–7, 2005*, pp. 98–106, 2005.

[Vesely*81] **William E. Vesely, Francine F. Goldberg, Norman H. Roberts and David F. Haasl**. *Fault Tree Handbook.* Handbook NUREG-0492, U.S. Nuclear Regulatory Commission, Washington, DC, USA, 1981.

[WaChCl07] **Mingyi Wang, Zuozhou Chen and Sylvie Cloutier**. *A hybrid Bayesian network learning method for constructing gene networks*. Computational Biology and Chemistry, 31(5-6):361–372, 2007.

[WagLew99] **Lee W. Wagenhals and Alexander H. Lewis**. *Converting Influence Nets with Timing Information to a Discrete Event System Model, A Coloured Petri Net*. In *Second Workshop on Practical Uses of Coloured Petri Nets and Design/CPN, Aarkus, Denmark, Proceedings*, 1999.

[YaKaFe94] **Ronald R. Yager, Janusz Kacprzyk and Mario Fedrizzi**. *Advances in the Dempster-Shafer theory of evidence*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[Yoon03] **Yew Khuen Yoon**. *Modelling Operational Risk in Financial Institutions using Bayesian Networks*. Master Thesis, Cass Business School, The City of London, 2003.

Index

Curriculum vitae

Personal Details	
Name	Alexander Pinl
Address	Im Pühlchen 2
	56072 Koblenz
Date of Birth	19th July 1979

Education	
1985–1989	Grundschule Mayen-Hausen (elementary school)
1989–1998	Megina Gymnasium Mayen (high school)
1998	Abitur (high school diploma)

Study of Computer Science and further Qualifications	
1998–2004	Diploma study of computer science at the University of Koblenz-Landau
2004	University degree (Diplom-Informatiker)
07/2004–11/2004	Project manager at the Chamaeleon AG, Montabaur
since 12/2004	Scientific assistant at the research group of Prof. Dr. Lautenbach, University of Koblenz-Landau (Some of the scientific results of the DFG-funded project ToMASEn under grant LA-1042/7 flew into this PhD thesis.)