From the Institute of Information Systems
of the University of Lübeck

Director: Prof. Dr. Ralf Möller

Indirect Causes, Dependencies and Causality in Bayesian Networks

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by
Alexander Motzek
from Rüsselsheim, Germany

Lübeck, 2016

Dedicated to my wife and my parents.

From the Institute of Information Systems
of the University of Lübeck

Director: Prof. Dr. Ralf Möller

Indirect Causes, Dependencies and Causality in Bayesian Networks

Dissertation
for Fulfillment of
Requirements
for the Doctoral Degree
of the University of Lübeck

from the Department of Computer Sciences

Submitted by
Alexander Motzek
from Rüsselsheim, Germany

Lübeck, 2016

First referee: Prof. Dr. Ralf Möller

Second referee: Prof. Dr. Maciej Liśkiewicz

Date of oral examination: October 19, 2016

Approved for printing. Lübeck, October 20, 2016

Abstract

Modeling causal dependencies in complex or time-dependent domains often demands cyclic dependencies. Such cycles arise from local points of views on dependencies where no singular causality is identifiable, i.e., roles of causes and effects are not *universally* identifiable. Modeling causation instead of correlation is of utmost importance, which is why Bayesian networks are frequently used to reason under uncertainty. Bayesian networks are probabilistic graphical models and allow for a causal modeling approach with locally specifiable and interpretable parameters, but are not defined for cyclic graph structures. If Bayesian networks are to be used for modeling uncertainties, cycles are eliminated with dynamic Bayesian networks, eliminating cycles over time. However, we show that eliminating cycles over time eliminates an anticipation of indirect influences as well, and enforces an infinitesimal resolution of time. Without a "causal design," i.e., without representing direct and indirect causes appropriately, such networks return spurious results.

In particular, the main novel contributions of this thesis can be summarized as follows. By considering specific properties of local conditional probability distributions, we show that a novel form of probabilistic graphical models rapidly adapts itself to a specific context at every timestep and, by that, correctly anticipates indirect influences under an unrestricted time granularity, even if cyclic dependencies arise. We show that this novel form of probabilistic graphical models follows familiar Bayesian networks' syntax and semantics, despite being based on cyclic graphs. Throughout this thesis, we show that no external reasoning frameworks are required, no novel calculus needs be introduced, no computational overhead for solving common inference-, query- and learning-problems is introduced, and that familiar algorithmic schemes remain applicable. We feel confident to say that Bayesian networks and dynamic Bayesian networks can be based on cyclic graphs. In effect, we show, for the very first time, that a novel dynamic probabilistic graphical model is an intrinsic representation of a full joint probability distribution over multiple full joint probability distributions.

Zusammenfassung

Betrachtet man kausale Zusammenhänge in komplexen oder zeitlich veränderlichen Domänen, sind oftmals keine *eindeutigen* Kausalitäten erkennbar, d. h. die Rollen von Ursachen und Wirkung sind nicht allgemeingültig bestimmbar. Betrachtet man Zusammenhänge lokal, d. h. aus einer rein kausalen Sicht von Ursachen auf Wirkungen, so werden durch veränderliche kausale Abhängigkeiten in erster Linie zyklische Abhängigkeiten benötigt. Es ist sehr wichtig, Kausalität anstatt von Korrelation zu modelieren, weshalb oftmals Bayes'sche Netze verwendet werden, um unter Untersicherheit Schlüsse zu ziehen. Bayes'sche Netzwerke sind gerichtete probabilistische graphische Modelle, erlauben eine kausale Modellierung von Wahrscheinlichkeitsprozessen und verfügen über lokal spezifizierbare und verständliche Parameter, sind jedoch nicht für zyklische Graphen definiert. Um zyklische Abhängigkeiten in Bayes'schen Netzen zu betrachten, werden oftmals dynamische Bayes'sche Netze verwendet, welche zyklische Abhängigkeiten in mehreren Zeitschritten auflösen. Durch diese zeitliche Auflösung werden jedoch nicht nur Zyklen aufgehoben, sondern es werden auch sämtliche indirekte Einflüsse innerhalb eines Zeitschrittes eliminiert, wodurch sich eine infinitesimal kleine Zeiteinteilung aufzwingt. Ohne eine kausale Modellierung, d. h. ohne direkte und indirekte Zusammenhänge kausal korrekt darzustellen, so zeigen wir in dieser Arbeit, können Modelle "‘kollabieren"’ und entartete Ergebnisse produzieren.

Der größte Beitrag dieser Arbeit lässt sich wie folgt zusammenfassen: Wir zeigen, dass spezielle Eigenschaften von bedingten Wahrscheinlichkeitsverteilungen es probabilistischen graphischen Modellen erlauben, sich rapide an einen speziellen Kontext anzupassen, wodurch sich Einflüsse korrekt modellieren lassen und indirekte Einflüsse beachtet werden, ohne dabei Anforderungen an eine Zeiteinteilung zu stellen, selbst wenn sich zyklische Abhängigkeiten ergeben. Zudem zeigen wir, dass diese gerichteten probabilistischen Modelle der üblichen Syntax und Semantik von Bayes'schen Netzen folgen, obwohl sie auf zyklischen Graphen basieren. Im Laufe dieser Arbeit zeigen wir, dass diese Modelle keine externen Rahmenwerke benötigen, zur Lösung übliche Inferenz-, Anfrage- und Lern-Probleme kein erheblich höherer Rechenaufwand nötigt ist und übliche Strukturen von bekannten Algorithmen erhalten bleiben. Wir sind daher überzeugt, dass Bayes'sche Netze und dynamische Bayes'sche Netze tatsächlich auf zyklischen Graphen basieren können. Schlussendlich ist es mit dieser Arbeit erstmalig möglich ist, mehrere Verbundwahrscheinlichkeiten innerhalb eines probabilistischen graphischen Modells zu repräsentieren.

Contents

Chapter 1

Bayesian Networks, Dependencies and Causality

Bayesian networks are probabilistic graphical models for reasoning under uncertainty from causal relationships between causes and effects. Pearl and Russell (2003) emphasize that Bayesian networks should be a direct representation of the world instead of a reasoning process. Therefore, directions of edges in a probabilistic graphical model shall represent causality and identify cause$\rightarrow$effect relationships, and should not be adjusted to properties and restrictions of a reasoning framework. While edge directions in probabilistic graphical models are irrelevant for the expressivity, i.e., ability to represent full joint probability distributions, only a *causal* edge direction allows for an intuitive, directly understandable, and local parametrization of conditional probability distributions (CPDs) in Bayesian networks. Pearl and Russell (2003) introduce local implications of parameters, i.e., CPDs, in Bayesian networks as their local semantics, to which we accredit the local understandability and direct, intuitive interpretability of CPDs as well. The product of those local parameters defines the global semantics of a Bayesian network without the need of any global normalization factors which allows one to locally and individually interpret these values. We consider these local semantics of CPDs as a highly valuable property of probabilistic graphical models, as they allow one to directly represent knowledge directly obtained from multiple experts from different expertises. For the scope of this thesis and the following definitions, we, therefore, solely consider models in which edge directions represent possible causal relationships.

As outlined throughout this thesis, the world is not necessarily causal if viewed in a too general setting. We show that no matter how acausal a world might seem, a Bayesian network must exactly represent a world or otherwise loses its expressiveness.

This introductory chapter is structured as follows. Section 1.1 introduces preliminaries on probabilistic graphical models and Bayesian networks and outlines an example domain and application of Bayesian networks for reasoning from causal models. Section 1.2 considers the role of causalities and extends a running example towards an emerging problem that motivates the work throughout this thesis, over which an overview is given by Section 1.3 and Section 1.4 describes this thesis' contributions.

1.1 Probabilistic Graphical Models

In general, a probabilistic graphical model (PGM) represents relationships, influences and dependencies between variables. A large variety of PGMs exists in the literature, from which we consider the form of *directed* probabilistic graphical models, and particularly focus on the form of Bayesian networks. We introduce both using the following running example.

Example 1.1 (*Joke-Cry* domain)**.** *Explaining why someone is crying and someone tells a joke at a funeral, why sad persons are crying, why someone cries at a wedding, or why a joke is told at a wedding are easy problems for a human. To brighten up an icing atmosphere and to cheer up crying (sad) persons, more-or-less good anecdotes or jokes are told; persons moved by weddings more-or-less inexplicably start crying; jokes are generally told at weddings and someone is usually happy. One is able to represent all these cause-and-effect relationships in one model as shown in Figure 1.1. Here, edge directions immediately identify causalities between causes and effects. While, from a reasoning view, a told joke provides information about crying persons, capturing a true causality is of utmost importance: Jokes are not told because of a funeral taking place, but jokes are told to partially cheer up persons in deep sorrow at a funeral. Without a directed edge from Cry to Joke, the only explanation (i.e., cause) for a joke being told at a funeral would be the funeral itself.*

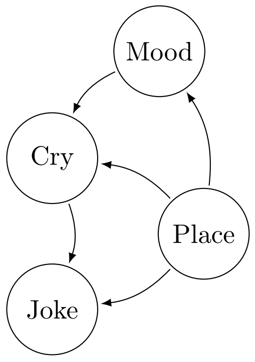

Figure 1.1. *Modeling causal dependencies is of utmost importance. Without a link from Cry to Joke, the only explanation for a Joke being told at a funeral is the funeral itself—usually a quite macabre assumption.*

This example domain, further called the joke-cry domain, is an example for a probabilistic graphical model and, in fact, for a Bayesian network. In this example $Joke$, Cry, $Mood$, $Place$ represent variables assignable to some values, e.g., the $Place$ can be a $funeral$ or a *wedding*. However, it is not certain which value each variable will take, which is why they are called random variables.

Notation 1.1 (Random variables)**.** *Let capital X represent a random variable and let X be assignable to a value of its domain $x \in \mathrm{dom}(X)$, denoted in lowercase. Let $P(X = x)$, or $P(x)$ for brevity, denote the probability of random variable X having x as a value. If $\mathrm{dom}(X) = \{true, false\}$ we write $+x$ for the event $X = true$ and $\neg x$ for $X = false$. If X is unspecified and not fixed by evidence, let $P(X)$ denote the probability distribution of X w.r.t. all possible values in $\mathrm{dom}(X)$.*

Every random variable is associated with a domain of values, to which the random variable are assignable. For the running example, one obtains $\mathrm{dom}(\mathit{Place}) = \{\mathit{funeral}, \mathit{wedding}\}$, $\mathrm{dom}(\mathit{Mood}) = \{\mathit{happy}, \mathit{sad}\}$, and *Cry* and *Joke* are binary domains, representing someone is (not) crying and (some/no)body tells a joke. Naturally, e.g., being in a sad mood does not inevitably lead to crying, which is why, in probabilistic graphical models, every influence is associated with a set of parameters representing a degree of uncertainty in their influence. For the scope of this thesis, we solely consider *directed* probabilistic graphical models which are parametrized using conditional probability distributions. A conditional probability, for example, represents a probability of crying under the consideration of attending a wedding and being happy.

Definition 1.1 (Directed probabilistic graphical model, PGM). *A PGM is syntactically defined as a directed graph $B = \langle V, E\rangle$ with vertices V and edges E. Every vertex represents a random variable $X_i \in \vec{X}$ assignable to one of its values $x_i \in \mathrm{dom}(X_i)$. Every directed edge from vertex X to Y represents a possible direct* influence *of X on Y and a possible direct causal dependence of Y on X. Every absent directed edge from vertex X to Y assures that X does not exert a direct causal influence on Y and that X is not a direct cause of Y. Every random variable is associated a local conditional probability distribution (CPD) $P(X|\vec{par}_X)$, where $\vec{par}_X$ represents the set of random variables that directly causally influence X. If $\vec{par}_X$ is empty, we call X a* prior (random variable). *A PGM's global semantics is defined as the joint probability distribution (JPD) over all random variables $P(\vec{X})$, which is given by some combination, specific to each PGM, of the locally defined CPDs.* ▲

Note that by Definition 1.1 an edge does only represent a possibility for a direct causal influence, but does not guarantee it. Further note that by Definition 1.1 an absent edge from random variable X to Y does *not* assure a global independence between both considering the global semantics of the model; a missing edge from X to Y assures that no direct causal influence is exerted, for which we speak of a *local conditional independence* of Y on X given $\vec{par}_Y$. Further, the complete graph B encodes global conditional independence assertions, i.e., a variable X is globally conditionally independent from its non-descendants given its parents $\vec{par}_X$.

With a specification of a conditional probability distribution for every random variable in the joke-cry domain in addition to Figure 1.1, a PGM is formed. Often, e.g., by Koller and Friedman (2009), Murphy (2012), or by Pearl and Russell (2003), directed probabilistic graphical models are used synonymously with the term *Bayesian network*. During the scope of this thesis we revisit the foundations of Bayesian networks and, therefore, require a careful differentiation and definition of (directed) probabilistic graphical models and Bayesian networks, which are defined as follows.

Theorem 1.1 (Bayesian network). *A Bayesian network B is a directed probabilistic graphical model where the PGM's associated graph $\langle V, E\rangle$ is a directed acyclic graph (DAG). A Bayesian network's (global) semantics is defined as the PGM's semantics and given as the product of all locally defined CPDs, i.e.,*

$$P(\vec{X}) = \prod_{X \in \vec{X}} P(X|\vec{par}_X) \tag{1.1}$$

▲

Figure 1.1 is a DAG and, therefore, the PGM introduced by Example 1.1 for the joke-cry domain represents a Bayesian network. Although, the definition of a Bayesian network's JPD (Eq. 1.1, derived as part of Proof of Theorem 1.1 given in Appendix A) might seem unimpressive, it is its simplicity that characterizes Bayesian networks as a world-representing first class declaration. The global semantics, i.e., the joint probability distribution over all random variables Eq. 1.1, is directly given as the product of all defined conditional probability distributions, i.e., parameters of the PGM, and the JPD does not involve any "normalizing" factors. This means that all defined parameters are directly and locally interpretable in a Bayesian network. Pearl and Russell (2003) introduce local implications of parameters, i.e., CPDs, in Bayesian networks as their local semantics, to which we accredit the local understandability and direct, intuitive interpretability of CPDs as well. While the edge directions in a Bayesian network are irrelevant for the expressivity, i.e., ability to represent full joint probability distributions, only a *causal* edge direction allows for this intuitive, directly understandable, local parametrization of conditional probability distributions (CPDs) in Bayesian networks. If an edge does not represent a direct causal relationship, such an edge must consider transitive and inverse implications of effects on causes which is inherently non-local and hardly overseeable by a human. We, therefore, as mentioned in the beginning, solely consider models where edge directions represent direct causal cause→effect relationships, which, per se, are intensively studied by, e.g., Pearl (2009). The following example demonstrates the intuitive and directly understandable parametrization of Bayesian networks under a causal design.

Example 1.2 (CPDs and local semantics)**.** *Continuing the joke-cry domain from Example 1.1, one needs to specify the CPDs* $P(Cry|Place, Mood)$*,* $P(Mood|Place)$*,* $P(Joke|Cry, Place)$*, and a prior random distribution* $P(Place)$*. In this example, we exemplarily demonstrate the parametrization of* $P(Cry|Place, Mood)$*. To do so, one needs to specify a local conditional probability for every dependance instantiation, i.e., every instantiation of* $Place, Mood$*, namely* $P(+cry|wedding, happy)$*,* $P(+cry|funeral, happy)$*,* $P(+cry|wedding, sad)$*,* $P(+cry|funeral, sad)$*. A respective counter part, e.g.,* $P(\neg cry|wedding, happy)$*, is then directly given by* $P(\neg cry|wedding, happy) = 1 - P(+cry|wedding, happy)$*. It is quite usual to specify CPDs in Bayesian networks by using conditional probability tables (CPTs), as demonstrated in Table 1.1.*

Each probability value of the CPD $P(Cry|Place, Mood)$ *is interpretable as itself and, moreover, is locally interpretable. This means that each defined conditional probability is independent of the use case of the complete model. For example, the specified probability of a happy person crying at a wedding is assessed to be* 70%[1] *and represents the same probability of, e.g., tossing less than 7 given one tosses a ten-sided dice. Further, one is not urged to consider backwardly-directed implications of* $Joke$*, as only direct* causes *need to be considered. Therefore, this CPD remains valid even if, say, one extends the domain with a deep and complex Bayesian network reasoning from individual spoken words if a joke was told or not.*

The example shows that CPDs are locally interpretable, i.e., demonstrates the local semantics of a Bayesian network. Moreover, all parameters are interpretable

[1]We use % in its mathematical meaning "per hundred," i.e., to represent a *number* and not a ratio of a whole, e.g., 70% is identical to 0.7.

Table 1.1. *Example for a CPD $P(Cry|Place, Mood)$ represented as a CPT for the joke-cry domain.*

Place	**Mood**	$\mathbf{P}(+cry|Place, Mood)$
wedding	*happy*	70%[1]
wedding	*sad*	20%
funeral	*happy*	1%
funeral	*sad*	80%

and specifiable without a need to consider the big picture. This means that every CPD is defined locally by considering its direct parents, i.e., its direct causes, and one is neither urged to consider transitive effects of grandparents nor urged to consider causally-inverse implications of descendants. Using the direct interpretation of CPDs, one is able to validate all CPDs separately and locally. This is especially useful for applications where one must rely on (subjective) expert assumptions and is not able to validate parameters and results against ground truth.

Example 1.2 uses a conditional probability table (CPT) to represent CPD $P(Cry|Place, Mood)$ in Table 1.1, but please note that CPTs are just one form for representing CPDs, and CPDs can as well be represented by three dimensional graphs, lists of probabilities, or, in some cases, as decision trees. We believe it is best to see a CPD $P(X|\vec{Z})$ as a function associated with random variable X returning a value between 0 and 1, depending on arguments of X's value x and an instantiation $\vec{z}$ of X's dependencies $\vec{Z}$. This view is as well taken by Koller and Friedman (2009).

To obtain results from PGMs, i.e., to reason about possible values of random variables, a PGM, seen as a probabilistic knowledge base, is queried for results of conditional probabilities. For example, to obtain knowledge where one is located, given one sees someone cry and hears a joke being told, one queries for $P(Place|+cry, +joke)$. Finding an answer to this query poses a probabilistic inference problem solvable by Bayes' theorem and marginalization from the JPD, as the following example outlines. Common queries and associated answering problems are discussed more deeply in Section 2.3.

Example 1.3 (Global semantics and answering a query)**.** *Continuing Example 1.2 with a specification of all CPDs, the BN's global semantics is simply given by the product of all locally defined CPDs, i.e.,*

$$\begin{aligned} &P(Place, Mood, Cry, Joke) = \\ &\quad P(Joke|Cry, Place) \cdot P(Cry|Place, Mood) \cdot P(Mood|Place) \cdot P(Place) \qquad (1.2) \end{aligned}$$

Based on this BN, say, one intends to obtain knowledge about the probability of being at a wedding while seeing crying persons and hearing jokes being told, i.e., $P(wedding|+cry, +joke)$. An exact answer is obtained by marginalization over all unobserved random variables from the JPD as

$$P(wedding|+cry, +joke) = \frac{\sum_{Mood} P(wedding, Mood, +cry, +joke)}{\sum_{Place} \sum_{Mood} P(Place, Mood, +cry, +joke)} .$$

Often, it is more interesting to obtain a distribution over possible values of a random variable, e.g., $P(Place|+cry, +joke)$. When calculating distributions, *one commonly avoids an explicit calculation of the denominator by using a normalizing factor α s.t. $\sum_{Place} P(Place|{+}cry, {+}joke) = 1$. Further, one usually avoids the explicit calculation of $P(Place, Mood, Cry, Joke)$ by using the product representation (i.e., the right side of Eq. 1.2), and one factors out CPDs in the summation.*

To answer this query from a semantic point of view, one needs to incorporate that +cry and +joke deliver information about a potential Place (abduction), and that +cry and some Place deliver information about Joke (prediction). Therefore, one obtains some information about Place that leads to information about Mood (prediction), but, +cry delivers information about Mood (abduction) as well, i.e., information from two "sides" must be merged.

Furthermore, say, to represent that a wound makes people cry as well, one models that a Wound influences Cry additionally. Then, if one observes crying persons, beliefs in +wound, sad, and funeral (these are all modeled causes for crying) increase (abduction). But if one additionally knows that people attend a funeral, the belief in +wound suddenly lowers, i.e., $P({+}wound|funeral, {+}cry) < P({+}wound|{+}cry)$: A Bayesian network can "explain causes away," although no information about the cause is directly given and its effect is in place.

Obtaining exact answers to queries in PGMs based on exact probabilistic inference is computationally expensive, and often approximate inference techniques are used. Still, the direct incorporation of prediction, abduction and explaining away in BNs is an extremely valuable reasoning mechanism, which, as demonstrated in Example 1.3 and stated by Pearl and Russell (2003), is otherwise difficult to implement. Moreover, due to the local semantics of parameters, once all parameters are validated, every result derived from this model is validated as well, as no global normalization factor is involved.

To further accentuate the need of local and global semantics, consider the following counter-example.

Example 1.4 (Counter-example of non-local semantics)**.** *Say, one intends to mimic reasoning capabilities of Bayesian networks, as outlined in Example 1.3, in a new reasoning framework. Then, considering Figure 1.1, at first, uncertainty degrees must be specified somehow. Say, one specifies the uncertainty from $Place = wedding \rightarrow Mood = happy$ by 5. However, the value 5 of this parameter has no meaning as itself and does not bear any (local) semantics. 5 first gains a degree of meaning, once one specifies a range of possible values, e.g., $[0, 15]$. Still, 5 on a scale from 0 to 15 only bears a very limited degree of interpretation, as it is unclear how this range is scaled. In order to judge this scale, other parameters must be considered. Say, one knows it is highly unlikely that jokes are told at a funeral where no one cries, which is parametrized by 4. 5 now seems to be more unlikely, but one has to consider distant, unrelated relationships as well to perform a judgment. In fact, one is first able to judge the likeliness of a parameter 5 by considering* all *specified parameters. In order to mimic the reasoning capabilities, one now needs to combine all parameters by an algorithm, answering a query for, e.g., the likeliness of being at a wedding, given one hears a joke and people are crying. Still, as parameters do not bear any local semantics, and no global semantics are defined for this framework, this algorithm must be validated against ground truth. In essence, for every domain that is supposed to be modeled by this framework and algorithm, large datasets must be present against which an algorithm can be validated and can be trained from. In the end,* *one would obtain an answer of, say,* 13*. However, one now requires large sets of reference values to judge this likeliness or must deeply understand the complete framework and algorithms.*

In contrast, in a Bayesian network, local parameters, e.g., $P(happy|wedding) = 89\%$*, immediately bear a meaning and are interpretable. For every random variable only the* direct *causes must be considered, and locally independent random variables (e.g., Joke) are completely irrelevant for the parametrization of* $P(Mood|Place)$*. Note that under consideration of the global semantics of this Bayesian networks, Joke and Mood are* not *independent, and an observation of one random variable will deliver information about the other. Still, due to the local views taken in CPDs, one need neither consider these backwardly-directed implications and nor consider the global interdependencies of all variables for specifiying the local CPD* $P(Mood|Place)$*. Moreover, every result obtained from a Bayesian network (e.g.,* $P(wedding|{+}cry, {+}joke) = 66\%$*) is immediately interpretable by itself, does not require any reference values, and is seen as validated once an expert trusts or assures that local CPDs are reasonable from a subjective perspective. Moreover, in order to interpret this value, an expert need not even understand probabilistic inference.*

This example accentuates—and exaggerates—the advantages and needs of well-defined local and global semantics. The need for these properties are further outlined and applied in a real world use case involving experts' assessments in Chapter 6.

To summarize, in a Bayesian network, one is able to capture true causalities from local point of views, experts' assessments for degrees of uncertainty are directly integrable without a need to consider a big picture, and obtained results are defined to be correct once experts' assessments are correct. The combination of these properties allow a Bayesian network to be a first-class representation of the world, i.e., a Bayesian network *is* everything required to reason under uncertainty and must not be hidden inside some probabilistic reasoning framework.

In the following section we extend the joke-cry domain with further places and encounter that capturing true causalities in Bayesian networks from local points of views quickly leads to significant problems in Bayesian network formalisms.

1.2 Cyclic Dependencies and Causality

To repeat, Pearl and Russell (2003) emphasize that Bayesian networks should be a direct representation of the world instead of a reasoning process. Therefore, directions of edges in a probabilistic graphical model shall represent causality and identify cause→effect relationships, and should not be adjusted to properties and restrictions of a reasoning framework. The following example shows that it is not easy to adhere to this principle. To represent the world more precisely in our joke-cry domain, we extend the domain in the following example with further places where one usually hears jokes and/or needs to cry.

Example 1.5 (Human interaction and emotions)**.** *Cause and effects of human interactions are often not uniquely identifiable without further context and play a major role in Winograd challenges (see, e.g., Levesque, Davis, & Morgenstern, 2012). Crying need not always be emotionally triggered by being moved,* *but can also be triggered from pure joy, e.g., hearing the best joke ever told. In this particular situation, for example at a comedy festival or party, a told (very good) joke might make someone cry, i.e., Joke $\rightarrow$ Cry. To integrate this new information into our world-representing model from Figure 1.1, we extend* dom(*Place*) *by* $\langle comedy - festival, party \rangle$ *and, from a pure local point of view, now, Cry is dependent on Joke and one obtains Figure 1.2. However, Figure 1.2 is not a DAG anymore, as an allegedly directed cycle is evident from the graph. Therefore, one does not obtain a Bayesian network anymore, by which one loses all local semantics and the global semantics is not given. Still, all dependency- and influence considerations originate from local point of views from direct causes. What one has to note is that a causality is not singly identifiable, but is only known in a further context, e.g., if a funeral or a party is observed. However, the context is a random variable, is part of our domain itself, cannot be known in advance, and we intend to reason over it as well.*

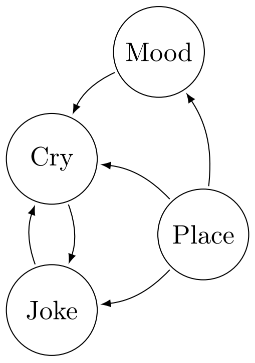

Figure 1.2. *Modeling causal dependencies often requires cyclic dependencies. If one is supposed to reason over Cry, potential influences* of *Cry or* on *Joke must be considered. Without a link from Joke to Cry, the only possible explanation for crying at a party is that the person is Mood=sad or that a person always cries at parties—usually a wrong assumption. And without a link from Cry to Joke, the only explanation for a Joke being told at a funeral is the funeral itself—usually a quite macabre assumption.*

Example 1.5 represents an example for a cyclic dependency created due to local views on dependencies, and one must conclude that dependencies cannot be modeled in a causally correct way. In essence, no straightforward solution in a Bayesian network formalism exists: Removing both edges of the alleged cyclic dependencies between *Cry* and *Joke* destroys causality, as the essential causes and effect relationships are removed: Jokes are not told at a funeral, just because of the funeral taking place; jokes are told at a funeral to possibly cheer up crying, sad persons. Further, someone (usually) does not cry at a comedy event because of being at a comedy event; a person is crying at a comedy event because jokes (which are usually told at comedy events) were good and made a person cry. One could mimic an intended joint probability distribution by a single edge between *Cry* and *Joke*, e.g., only *Cry* $\rightarrow$ *Joke*, which, however, destroys the local semantics: The local CPD of *Joke* must then be designed w.r.t. an intended influence of *Joke* $\rightarrow$ *Cry*, i.e., a CPD is not solely designed with respect to causal cause$\rightarrow$effect relationships from direct parents, but also with an inverse "reasoning" view on effect$\rightarrow$cause relationships. By moving away from directed PGMs, one is able to represent the outlined domain as a completely undirected PGM or as a partially directed PGM. For example, by sacrificing the identification of causality, the allegedly cyclic edges between *Joke* and *Cry* could be represented by one undirected edge associated with a factor for uncertainty. However, such factors, sometimes called "potentials," bear no local meaning and are not easily interpretable. Moreover, local semantics are lost, as only a global normalization factor on the complete model enables a certain degree of interpretation. Note that, such allegedly cyclic dependencies need not be evident directly between two variables, but could emerge from a constellation of many random variables in large domains.

In summary, cyclic dependencies are required from a causal perspective on Bayesian networks and arise from local views on dependencies, as envisioned in Bayesian networks. Without cyclic dependency structures in a Bayesian network, local interpretation of parameters and causalities are lost. However, cyclic dependencies in Bayesian network are fundamentally forbidden, as every definition of a Bayesian network begins with "a Bayesian network is a directed *acyclic* graph.", which, as we show in the remainder of this thesis, is not necessarily true. We continue the joke-cry example as an example for a novel well-defined PGM, introduced in the following chapters, using familiar Bayesian network syntax and semantics in Section 5.2 on Page 88.

1.3 Overview over the Remainder of this Thesis

The remainder of this thesis is structured as follows. Chapter 2 discusses preliminaries on dynamic probabilistic graphical models (DPGMs) and shows that the inability to model dependencies in a causally correct way poses a compelling problem in dynamic Bayesian networks (DBNs). Namely, we show that in certain domains DBNs are unable to anticipate indirect influences and enforce an infinitesimal resolution of time. By considering properties of local conditional probability distributions (CPDs) associated with random variables, we introduce a novel, directed (dynamic) probabilistic graphical model that permits locally seen cyclic dependencies, allowing for a causal design, anticipating indirect influences on an unbound time granularity. In fact, we show and prove that the introduced (dynamic) probabilistic graphical model is remarkable similar to a (D)BN and preserves all attributed desired properties, which is why we call them Activator (D)BNs (ADBNs). We discuss and derive solutions to commonly known query answering problems and, further, that ADBNs do not introduce any computational overhead, compared to its closest relative in classic DBN formalisms. To reduce computational complexity of finding solutions to common query answering problems in ADBNs, we introduce, derive and discuss approximate inference techniques for ADBNs and show that familiar approaches remain applicable. As this thesis covers a broad field of research areas, we discuss related work on each chapter separately in each chapter.

While Chapter 2 shows that ADBNs, as a novel form of DPGMs, enable a causal parametrization and creation of models from local perspectives, an expert might now always be present or must be assisted to do so. Therefore, Chapter 3 discusses and introduces a learning approach towards ADBNs based on an EM-algorithm. We discuss that classical approaches such as structural EM algorithms and learning approaches for non-stationary DBNs are not applicable towards learning ADBNs, as ADBNs pose a novel challenge: A structure is not and cannot be known in advance. That structures could be known in advance is so far assumed for all DPGM learning approaches. We, further, continue the discussion of ADBNs' closest relatives in classic DBN formalisms. Chapter 2 has shown that classic DBN formalisms are not parametrizable and creatable from causal, local perspectives, leading to a spurious anticipation of indirect influences in certain domains. Now, Chapter 3 discusses if classic DBN models can learn from data in these domains to anticipate indirect influences in an unbound time granularity. In fact, we empirically evaluate that not even the most general form of DBNs is able to learn an anticipation of indirect influences in these particular domains.

In Chapter 4 we focus more deeply on specific properties of local CPDs in (D)PGMs and formalize an often mentioned, but yet unformalized property of CPDs, which we call innocuousness. In particular, it is often assumed that "a false dependence does not cause any harm," and an arc in a graphical model can be left out if a potential cause is known to be non-harmful. Chapter 4 formalizes a general form of innocuousness in the form of innocuousness-contexts and exploits their properties of vacuous arcs in ADBNs. While, semantically, often accredited to commonly known Boolean combination functions for creating CPDs, e.g., noisy-or assumptions, ADBNs enable one to formalize this property for the very first time.

Where the previous chapters introduce and discuss ADBNs as a novel DPGMs with softer, but still existing, restrictions on observations and instantiations, Chapter 4 shows that these restrictions can be significantly relaxed. In Chapter 5 we discuss the rationale behind such restrictions and show that, by adequate parametrization and modeling approaches, ADBNs are able to represent multiple structures in one model and are able to intrinsically adapt to a specific context at every timestep with neither restrictions on observations or instantiations, nor introduction of external reasoning frameworks. In effect, we show, for the very first time, that ADBNs are (D)PGMs that representat multiple joint probability distributions in one model, i.e., are a graphical representation of a joint probability distribution of joint probability distributions. In fact, we show that the joke-cry domain introduced by Example 1.5 and Figure 1.2 from Chapter 1 is already such a well-defined probabilistic graphical model.

In Chapter 6 we discuss and introduce a pertinacious problem in the field of business informatics and cyber security. Large industrial companies become more and more dependent on large and complex IT infrastructures supporting critical operations inside the company. Any defect, attack, or executed action, i.e., any *impact* on some resource might transitively lead to a causal chain of failures affecting a critical operation of a company and therefore a company itself. Assessing these eventualities is a novel field of research often referred to as a mission impact assessment, and Chapter 6 shows that local and global semantics of (dynamic) probabilistic graphical model are highly beneficial for these assessments. The local interpretation of locally defined parameters provides a direct integration of expert knowledge. Further, PGMs permit validating parameters, instead of holistically validating obtained results against ground truth. Such large amounts of ground truth are often not available in such domains. We discuss a dynamic mission impact assessment providing real-time and forensic analyses of potential impacts caused inside largely scaled networks based on a DPGM and show that ADBNs are inevitably required and directly evident from the problem's domain.

We conclude, summarize and discuss results obtained throughout this thesis in Chapter 7.

1.4 Scientific Contribution

The scientific contributions of this thesis can be summarized as follows: We show that the restriction of acyclic graphs of classic DBN formalisms enforces an infinitesimal resolution of time in certain domains, or, otherwise, leads to an ignorance of indirect influences. By considering DPGMs in which some random variables show to have an activator nature, we prove that such DPGMs, called ADBNs, are subject to a different acyclicity constraint and can be based on cyclic graph structures. We show that ADBNs intrinsically are able to rapidly adapt to contexts, without requiring an infinitesimal resolution of time, anticipating indirect influences on a solid mathematical basis using familiar Bayesian network semantics. This is beneficial for applications where causal models arise naturally and cyclic dependencies arise through local views on (in)dependencies, e.g., automatic learning of causal influences from coarse observation sets and—as a long-term goal—finding causally correct explanations and relations in (temporally uncertain) knowledge bases requiring context-aware interpretations and anticipations of causal chains, as required for, e.g., DeepQA (Ferrucci et al., 2010) or the Knowledge Vault (Dong et al., 2014).

Moreover, by proposing a learning approach for ADBNs for problems where effective structures are not known in advance, are rapidly changing over time, and are not evident from data while learning, we effectively fuse structure learning with parameter learning into one atomic task to learn DBNs from incomplete data under unknown structures. On top of that, we question whether it might be possible that "diagonal" (A)DBNs can be learned, whose parameters do not bear a local semantics, but which deliver satisfying inference results anticipating indirect influences. We show that not even the most general form of diagonal (A)DBNs can be trained to anticipate indirect influences.

Furthermore, by formalizing a yet unexpressed innocuousness property in CPDs, expressiveness of local semantics of CPDs is increased, and acyclicity restrictions on (A)DBNs are relaxed. Namely, we unveil that in ADBNs one is able to novelly formalize that "a deactive nodes does not cause any harm," and an arc could be left out in a model. Based on graph enumeration techniques we quantitatively explore new relaxations of syntactic restrictions of graphical models for (A)DBNs.

In addition, based on a consideration of specific modeling approaches and a constraint of local CPDs, we show that ADBNs form (dynamic) probabilistic graphical models where partial structural information can be missing completely. In effect, we prove that ADBNs are well-defined DPGMs intrinsically representing multiple structures, and, for the very first time, are a representation of multiple joint probability distributions. This means that one is able to represent joint probability distributions of joint probability distributions. In conclusion we show that all local and global semantics, i.e., locally interpretable and parametrizable parameters of causal models remain without a need of global normalization factors in ADBNs. We show that familiar schemes for learning, exact inference, and approximate inference in ADBNs are evident, that no computational overhead is introduced, and that one remains in a classical and familiar calculus. By showing that, through adequate modeling approaches, any restrictions on ADBNs are in fact removable, we feel confident to say that (dynamic) Bayesian networks can directly be based on cyclic graphs.

Finally, we contribute to an emerging research area in cyber security and business informatics by introducing a well-defined dynamic probabilistic graphical model for dynamic mission impact assessment, which is predestined for realtime and forensic analyses of mission impacts. We reduce impact assessment onto a probabilistic inference problem, which allows validating results at data level, and does not require deep training of experts to understand and validate holistic results. By the use of local semantics of ADBNs one is able to integrate widespread knowledge from different expertise into a single sound model. This is useful for applications where qualitative assessments are required and perpendicular views from multiple experts onto a problem must be brought inline and require a view over time.

The following papers and articles have been peer-reviewed, accepted and published as part of this thesis.

Motzek, A., Möller, R. Indirect Causes in Dynamic Bayesian Networks Revisited. In *IJCAI 2015: 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015*, pp. 703–709.

Motzek, A., Möller, R. Exploiting Innocuousness in Bayesian Networks. In *AI 2015: 28th Australasian Joint Conference on Artificial Intelligence, Canberra, Australia, November 30 - December 4, 2015*, pp. 411–423.

Motzek, A., & Möller, R. Context- and Bias-Free Probabilistic Mission Impact Assessment. *Computers & Security, ISSN 0167-4048, Vol. 65, 2017*, pp. 166–186.

Motzek, A., Möller, R., Lange, M., Dubus, S. Probabilistic Mission Impact Assessment based on Widespread Local Events. In *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015*, pp. 16–22.

Motzek, A., Geick, C., Möller, R. Semantic Normalization and Merging of Business Dependency Models. In *CBI 2016: 18th IEEE Conference on Business Informatics, Paris, France, August 29 - September 1, 2016*, pp. 7–15.

Motzek, A., Möller, R. Probabilistic Mission Defense and Assurance. In *NATO IST-148: STO Symposium on Cyber Defence Situation Awareness, STO-MP-IST-148, Sofia, Bulgaria, October 3-4, 2016*, pp. 4-1–4-18.

Grenadillo, G. G., Motzek, A., Garcia-Alfaro, J., Debar, H. Selection of Mitigation Actions Based on Financial and Operational Impact Assessments. In *ARES 2016: 11th International Conference on Availability, Reliability and Security, Salzburg, Austria, August 31 - September 2, 2016*, pp. 137–146.

Grenadillo, G. G., Alvarez, E., Motzek, A., Merialdo, M., Garcia-Alfaro, J., Debar, H. Towards an Automated and Dynamic Risk Management Response System. In *NORDSEC 2016: 21st Nordic Conference on Secure IT Systems, Oulu, Finland, November 2-4, 2016*, pp. 37–53.

A digital edition of this PhD thesis, electronic versions of published articles and papers, and information about ongoing research are available on `http://adbn.motzek.org`. The website further provides supplementary material to this thesis, including an open-source implementation for inference in ADBNs and learning of ADBNs, and data sets for all performance-evaluation used throughout this thesis.

This work has been partly supported by the Seventh Framework Programme (FP7) of the European Commission as part of the PANOPTESEC integrated research project (GA 610416).

Chapter 2

Indirect Causes in Dynamic Bayesian Networks Revisited

Dynamic Bayesian networks (DBNs) are an extension to Bayesian networks motivated from two perspectives, on the one hand as a manifestation of cyclic dependencies over time, closely related to Markov models (Murphy, 2002), on the other hand as a stationary process repeated over time in fixed timeslices (Glesner & Koller, 1995) to reason about historical and future evolutions of processes. Considering Pearl and Russell (2003) who emphasize that Bayesian networks should be a direct representation of the world instead of a reasoning process, Murphy's and Glesner's views seem to be conflicting. A model repeated over time with cyclic dependencies would expand to infinity already for one timeslice. Therefore, e.g., Jaeger (2001) or Glesner and Koller (1995) use a strict order of dependencies s.t. state variables of time t are only dependent of states at $t-1$. Unfortunately, this means that evidence at a certain time point does not affect states at this time point, but one slice later. We argue and show that this limits the causal expressiveness of Bayesian networks.

In the extreme form of a directed dynamic probabilistic model, each random variable is locally and causally seen as being directly dependent on every other random variable of one timestep, inevitably creating cyclic dependencies. When using DBNs to allow for local specifications and interpretation of probability distributions, there exists no option to causally correctly leave all dependencies in the same timestep as dependencies create cycles. Therefore, some random variables can only be dependent on variables from a previous timestep. However, this poses serious conflicts in causality, as (i) the temporal causality is simply inaccurate and (ii) no indirect effects are considered at a particular timestep, enforcing an infinitesimal resolution of time adjusted to a reasoning process, instead of a time modeled for a world representation. Both restrictions (i) and (ii) severely limit the usage of DBNs.

To circumvent this problem, basically two options are available. As investigated by Boutilier, Friedman, Goldszmidt, and Koller (1996), variables might be independent in certain contexts, which would allow a causally correct network generation from rules such as those presented by Glesner and Koller (1995), Ngo, Haddawy, and Helwig (1995), or by Ngo and Haddawy (1997). Still, then rules often need to be designed with a procedural view, degrading a BN to a procedural tool in a reasoning process, rather than designing it as a first-class declarative representation. Further, such rules would inherently be cyclic and might cause additional problems as stated by Ngo et al. (1995). A second option would be to heavily restrict a DBN to specialized observation sets, e.g., to “single observations at a time” as done by, e.g., Jaeger (2001) or Sanghai, Domingos, and Weld (2005), s.t. no indirect causes need to be considered. Obtaining only single observations during one timeslice again implies that observations are made at an infinitesimal resolution of time.

The contribution of this chapter can be summarized as follows: By considering DBNs in which some random variables have an activator nature, we prove that such DBNs, called ADBNs, are subject to a different acyclicity constraint by rapidly adapting to different contexts at every timestep, do not enforce an infinitesimal resolution of time, and anticipate indirect influences on a solid mathematical basis using familiar Bayesian network semantics. This is beneficial for applications where causal models arise naturally and cyclic dependencies arise through local views on (in)dependencies, e.g., automatic learning of causal influences from coarse observation sets and—as a long-term goal—finding causally correct explanations and relations in (temporally uncertain) knowledge bases requiring context-aware interpretations and anticipations of causal chains, e.g., DeepQA (Ferrucci et al., 2010) or the Knowledge Vault (Dong et al., 2014).

This chapter is structured as follows. We discuss preliminaries on DBNs and context-specific independencies as introduced by Boutilier et al. (1996) and Haddawy, Helwig, Ngo, and Krieger (1995) in Section 2.1 and identify an eminent problem of causality in DBNs by the use of a running example. By considering DBNs in which some random variables have an activator nature, we introduce Activator Dynamic Bayesian Networks (ADBNs) in Section 2.2. We investigate common queries and associated problems in ADBNs such as filtering and smoothing in Section 2.3 and provide experimental results for exact inference. Section 2.4 is dedicated to an introduction and derivation of an approximate inference technique for (A)DBNs. We discuss our results and relations to previous work in Section 2.5. Section 2.6 gives an intermediate conclusion an outlook to following chapters.

2.1 Dynamic Bayesian Networks: Preliminaries

The following definitions, propositions and theorems on dynamic probabilistic graphical models and dynamic Bayesian networks follow familiar notations and definitions. However, painstaking definitions are needed to shed light on marginal details which lead to a significant problem of anticipating indirect influences in DBNs and limit the expressiveness of (D)BNs as we show in the following section.

A PGM models influences and dependencies between random variables, e.g., “a wet lawn is caused by rain,” where rain is a prior random variable. However, seeing rain as a fixed prior random variable is not necessarily correct, as it is easy to see that it is more likely for it to rain tomorrow, if it rained for the past three weeks. Still, the dependence between lawn and rain remains structurally the same over time. Dynamic (directed) probabilistic graphical models (DPGMs) are used to represent such models, where some dependencies depend on previous states of random variables. Therefore, a DPGM models dependencies between random variables where random variables exist in fixed, discrete timeslices and are dependent on random variables from the same or from previous timeslices. Random variables that depend on their direct predecessor are stateful, which is why we call them state variables.

Notation 2.1 (State variables)**.** *Let X_i^t be a random variable. X_i^t is the i^{th} state variable X_i at time t, where time t is represented by the t^{th} discrete timeslice and t represents some wall-clock time t. Every X_i^t is assignable to a value $x_i \in \mathrm{dom}(X_i^t)$, where $\mathrm{dom}(X_i^t) = \mathrm{dom}(X_i)$ for all t. Let $\vec{X}^t$ be the vector of all n state variables at time t s.t.*

$$\vec{X}^t = \left(X_1^t, \ldots, X_n^t\right)^\intercal .$$

A random variable X_i^t is a state variable if it bears a history, i.e., is at least dependent on X_i^{t-1}. Let $P(X_i^t = x_i)$ (or $P(x_i^t)$ for brevity) denote the probability of state X_i having x_i as a value at time t. If $\mathrm{dom}(X) = \{true, false\}$ we write $+x^t$ for the event $X^t = true$ and $\neg x^t$ for $X^t = false$. If X_i^t is unspecified and not fixed by evidence, $P(X_i^t)$ denotes the probability distribution of X_i^t w.r.t. all possible values in $\mathrm{dom}(X_i)$.

To represent influences and dependencies between state variables, a DPGM is specified by one initial model and a dependency pattern between variables from consecutive timeslices. It is assumed that dependencies and influences remain constant, and a modeled pattern is repeated for multiple timeslices forming an ever expanding model. For the scope of this thesis we focus on models where dependencies solely exist between two consecutive timeslices (Markov-1 assumption).

Definition 2.1 (Dynamic probabilistic graphical model, DPGM)**.** *A DPGM is syntactically defined as a tuple $(B_0, B_\rightarrow)$ with B_0, an initial PGM representing time $t = 0$ containing all state variables X_i^0 in $\vec{X}^0$, and $B_\rightarrow$, a consecutively repeated directed graph fragment for defining state dependencies between X_i^s and X_j^t, with $X_i^s \in \vec{X}^s, X_j^t \in \vec{X}^t, s \leq t$. For every random variable X_i^t a local CPD over all parents of X_i^t is specified. By repeating $B_\rightarrow$ for every time step $t > 0$, a DPGM $(B_0, B_\rightarrow)$ is unfolded into a directed graph representing a PGM B over random variables $\langle \vec{X}^0, \vec{X}^{1:t} \rangle$. The global semantics of a DPGM $(B_0, B_\rightarrow)$ is defined as B's global semantics.* ▲

Commonly, in a DPGM one distinguishes between permanently observed and unobservable variables (sensors and hidden states, respectively). For our work, we consider a fully observable Markov model containing only observable (but not constantly observed) random variables, i.e., hidden Markov models with varying observability spaces. Moreover, we will consider specific forms of DPGMs, namely, dynamic Bayesian networks. The relation of PGMs to Bayesian networks and of DPGMs to dynamic Bayesian networks is similar:

Proposition 2.1 (Dynamic Bayesian network, DBN)**.** *A DPGM $(B_0, B_\rightarrow)$ is called a DBN and is well-defined, if state dependencies defined in $B_\rightarrow$ are limited s.t. no cyclic dependencies are created during unfolding, i.e., B is a directed acyclic graph. This is the case if $B_\rightarrow$ is a directed acyclic graph fragment. An unfolded DBN represents a Bayesian network (BN), by which the global semantics of the DBN is given by the product of all locally defined CPDs of all timeslices.* ▲

A proof of Proposition 2.1 is directly equivalent to the proof of Theorem 1.1, as an unfolded DBN is a classic BN. As before, it is the simplicity of (D)BN semantics without global normalization factors that still enables local CPDs to define their own local semantics, i.e., to allow for local interpretations and specifications of (in)dependencies without considering global normalization factors. DPGMs, and therefore DBNs, are especially useful for reasoning about evolutions of processes, predicting future events, or performing analyses in retrospect, e.g., analyzing the current weather situation, predicting the weather for upcoming days, or reconstructing events that lead to a hail storm.

In the following, we revisit roles of indirect influences and dependencies in DBNs to broaden the representation abilities of (D)BNs while maintaining sound (D)BN semantics. To do so, we differentiate between different structural forms of DPGMs and DBNs.

Notation 2.2 (Inter- and intra-timeslice dependencies, and diagonal models)**.** *For state dependencies defined in $B_{\rightarrow}$ of the form $t-1 \leq s \leq t$ (see Definition 2.1), one speaks of a first order Markov property, which we focus on in this thesis. For any DPGM with $t-1 \leq s < t$, i.e., states at time t are only dependent of states at time $t-1$, an acyclicity constraint in the directed graph holds and a DPGM is a well-defined DBN. We call dependencies of the form $t-1 \leq s < t$* inter-timeslice *dependencies. We call DPGMs in which only inter-timeslices dependencies are defined* diagonal *models. For a limited set of dependencies of the form $t-1 \leq s \leq t$, called* intra-timeslice *dependencies, a DPGM is a well-defined DBN, as long as no directed cycles are created.*

More often than not, many DBNs are designed as diagonal models (as in Figure 2.2, gray), and introduced due to syntactic constraints on (D)BNs, but stand in conflict with an actual causality in their domain, as the following examples demonstrate. Such dependencies exist causally at $s = t$, but would create directed cycles in one timeslice. Dependencies on “sibling” states of one timeslice are then “spread over the past” and conflict with causality. This means that indirect causes among siblings are not anticipated correctly in a particular state, such that “chain reactions” are not covered appropriately. The following example introduces a running example used throughout this thesis.

Example 2.1 (Regulatory compliance)**.** *In a company a corrupt employee deliberately places fraudulent information, e.g., faked payment sums for bribe money, which spread throughout a company until every employee has (unknowingly) committed a compliance violation, i.e., has become corrupt, too. In order to trace back a potential source for a detected compliance violation, to reconstruct sequences of events and to prevent future compliance violations, we model a probabilistic regulatory-compliance domain using a DBN. If one employee spreads fraudulent information, we do not say that such an employee is “corrupt,” but in this context we assume that he is* tainted*. We speak of taintedness, because saying that an employee is corrupt implies that he is knowingly manipulating information; being tainted shall represent that he might distribute fraudulent information indeliberately. The taintedness state of every employee,* C*laire,* D*on and* E*arl, say, is represented by a state random variable in $\vec{X}^t$. The probability $P(X_i^t)$, encodes the belief in employee X_i being tainted $+x_i^t$ or being integrous $\neg x_i^t$ at time t. We model B_0 s.t. it models our prior belief in every employee being a source of fraudulent information, i.e., B_0 is a BN containing all $\vec{X}^0$ as random* *variables without any influences; say $P(+c^0) = 0.5, P(+d^0) = 0.6, P(+e^0) = 0.7$. Being tainted is assumed to be permanent, such that $B_{\rightarrow}$ describes all random variables X_i^t depending on X_i^{t-1} with conditional probability $P(+x_i^t | +x_i^{t-1}) = 1$.*

An employee might influence another employee in his writings or, rather, in his information state. A tainted employee might therefore (indeliberately) influence his colleague such that the colleague also falls prey to fraudulent information, i.e., becomes tainted, too. Influences happen through message exchanges, i.e., only if a message is passed from employee X to Y at t an influence is exerted. We represent message exchanges by random variables M_{XY}^t as part of our domain, with $+m_{XY}^t$ indicating that Y receives a message from X at time t.

In the example, one now can make observations, e.g., from unheralded compliance checkups, and trace a potential diffusion of false information throughout our company over time. Say, Claire influences Don, and if Claire is tainted there is a probability of Don becoming tainted, too. Further, if Don influences Earl there is a probability that Claire influences Earl *indirectly* through Don, i.e., Claire is an indirect cause of Earl becoming tainted. If one is inclined to model only this dependency of E on D on C, one can correctly represent the domain as in Figure 2.1. In this minimal example, indirect influences occur and are correctly anticipated. However, if more potential influences are supposed to be modeled one is at odds with causality as the following will show.

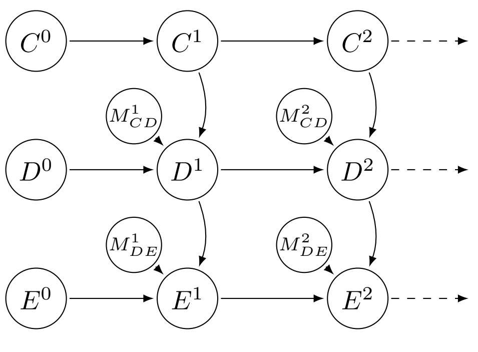

Figure 2.1. *A causally correctly represented world for the minimal case of Example 2.1 using a DBN. Messages M_{XY}^t are only possibly sent from Claire to Don and from Don to Earl. Modeling more possible message exchanges, i.e., influences, leads to Figure 2.2.*

Considering Example 2.1, we would like to model that *all employees can potentially influence each other*, which, however, would lead to a cyclic graph structure in $B_{\rightarrow}$ of a DPGM (as seen in Figure 2.2, black). In order to, at least, obtain a syntactically well-defined DBN, one must "bend" some dependencies to a consecutive/previous timestep in $B_{\rightarrow}$ (as done in Figure 2.2, gray) forming a diagonal model. This is unavoidable, but is seen as inaccurate from a world representation point of view, as indirect influences are now anticipated spuriously. Bending an influence to a consecutive timestep encodes an incubation time and, essentially, models a different process: Earl is now influenced by Claire *through* Don from a Claire of the penultimate timepoint. To approximate the intended meaning, a timeslice must be infinitesimally small, to somehow anticipate all indirect influences, and cannot be chosen appropriately for an intended use case, e.g., to represent a daily acquisition of information.

Remark 2.1 (Use of diagonal models). *Diagonal models can be used to simulate a cyclic "feedback" relationship, by letting a model converge to a stable state after long periods of time, which is used to simulate hidden Markov models with the use of DBNs, as, e.g., done by Murphy (2002) or Ghahramani (2001). However, when seeing each timeslice as a representation of a fraction of a real "wall-clock" time (a viewpoint taken, e.g., by Glesner and Koller (1995), Sanghai et al. (2005), Jaeger (2001), and us), a cyclic model would already expand to infinity during one timeslice, e.g., when going from day* 1 *(*$t = 1$*) to day* 2 *(*$t = 2$*).*

In the following we show that without a causal design, i.e., without modeling direct and indirect causes correctly, spurious results are obtained from such models. We propose a novel DPGM, called ADBN, able to represent locally seen cyclic dependencies to anticipate indirect influences without any demands on time granularity. Moreover, we show that it is remarkable similar to DBNs, but without an acyclicity constraint.

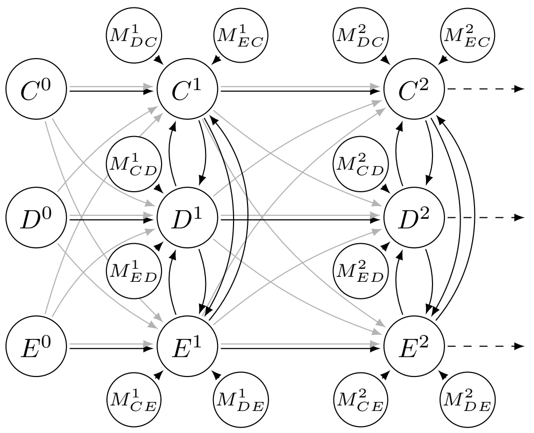

Figure 2.2. *Two options (black/gray) to represent a world using an (A)DBN for the running example if every employee (Claire, Don, Earl) can influence every other employee through messages* M^t_{XY}*. Syntactic DAG constraints of (D)BNs prevent cyclic dependencies, and diagonal state dependencies are enforced (indicated with gray arrows). The diagonal option is considered inaccurate from a world representation point of view. In the diagonal case,* M^t_{XY} *represents* $M^{t-1\,t}_{XY}$*, i.e.* M^t_{XY} *affects the dependency of state* Y^t *on* X^{t-1}*.*

Classically, a conditional independency in a Bayesian network is represented by the lack of arcs between some nodes. Another kind of independence in Bayesian networks, called context-specific independence (CSI), has been studied by Boutilier et al. (1996) & Ngo and Haddawy (1997). CSIs represent dependencies in a BN that are only present in specific contexts, and have mainly been used for more efficient inference, e.g., as studied by Poole and Zhang (2003) (cf. Section 2.5). We use the notation A_{XY} if a random variable acts as a so called *activator* random variable which activates a dependency of random variable Y on X in a given context. Note that activator random variables are *not* auxiliarily introduced, but are often already part of a domain. Therefore, we use the following definition to identify special random variables of a domain that have an activator nature.

Definition 2.2 (Activation / deactivation criteria)**.** *Let* $\mathrm{dom}(A_{XY}) = \{\mathit{true}, \mathit{false}\}$ *(extensions to non-Boolean domains are straightforward). The* deactivation *criterion for* $A_{XY} = \mathit{false}$ *is defined as*

$$\begin{aligned} &\forall x, x' \in \mathrm{dom}(X), \forall y \in \mathrm{dom}(Y), \forall \vec{z} \in \mathrm{dom}(\vec{Z}) : \\ &\qquad P(y|x, \neg a_{XY}, \vec{z}) = P(y|x', \neg a_{XY}, \vec{z}) = P(y|*, \neg a_{XY}, \vec{z}) \ , \end{aligned} \tag{2.1}$$

where $*$ *represents a wildcard and* $\vec{Z}$ *remaining direct dependencies of Y, i.e., direct parents of Y. Given* $\neg a_{XY}$*, we say that a direct causal dependence of Y on X, i.e., a direct causal influence of X on Y, is* inactive.

The activation *criterion describes a situation where Y becomes directly dependent on X, where the CPD entry for some* $y \in \mathrm{dom}(Y)$ *is not uniquely identified by just* ${}^{+}a_{XY}$ *and* $\vec{z}$*, hence*

$$\begin{aligned} &\exists x, x' \in \mathrm{dom}(X), \exists y \in \mathrm{dom}(Y), \exists \vec{z} \in \mathrm{dom}(\vec{Z}) : \\ &\qquad\qquad P(y|x, {}^{+}a_{XY}, \vec{z}) \neq P(y|x', {}^{+}a_{XY}, \vec{z}) \ . \end{aligned} \tag{2.2}$$

Given ${}^{+}a_{XY}$*, we say that the direct causal dependence of Y on X, i.e., the direct causal influence of X on Y, is* active.

If for a random variable A_{XY} *both activation and deactivation criteria are fulfilled by a local CPD definition of random variable* Y*,* A_{XY} *is called an* activator *random variable.* ▲

Note that Definition 2.2 is based on properties of locally defined CPDs of random variables in a (D)BN, i.e., some random variable of a (D)BN is identified to be an activator random variable by some specific numerical parameter settings in CPDs. Carefully note that the activation criteria and deactivation criteria only apply to these local CPDs, and do not apply to the global semantics of a (D)BN. Table 2.1 shows an arbitrarily specified CPD in which random variables show to have an activator nature and follow Definition 2.2. Note that activator criteria can be present in any form of CPDs, as remarked in the following.

Remark 2.2. *The (de)activation criterion can be summarized as: a probability is uniquely identified by active dependencies, and inactive dependencies become irrelevant. This property is easily confusable with a property of a "noisy-or combination" function (e.g., described by Henrion, 1988), where "false" dependencies (or as Heckerman and Breese (1996) say: in a "distinguished" state) are semantically supposed to be irrelevant. Under noisy-or assumptions, each conditional probability value in a CPD is obtained by a deterministic combination-function of individual probability fragments associated with "true" dependencies. Still, the general activation and deactivation criteria from Definition 2.2 are* not *linked to specific combination functions and can be present in arbitrarily specified CPDs (cf. Table 2.1). Nevertheless, the noisy-or combination function inherently defines activation and deactivation criteria in their semantics, where only "true dependencies act as potential causes" and a "false dependence does not cause any harm." We call such a property an "innocuousness-property," and explore their role in ADBNs deeply in Chapter 4, as ADBNs actually enable one to (semantically and syntactically) formalize such a property in CPDs. The difference between innocuousness and activator criteria is that for the general (de)activation criteria the presence (the activeness) of a dependence is* relevant *for every value. For innocuousness properties of CPDs, the presence of a "false" dependence is* irrelevant*, i.e., the activeness is only of interest for some values of the dependence.*

Furthermore, the following Example 2.2 demonstrates the identification of activator random variables on the running example of the taintedness domain.

Example 2.2 (Regulatory compliance continued)**.** *In Example 2.1 we modeled that Claire does not constantly exert an influence on Don, but* only if *Claire sends a letter to Don. In fact, message exchange variables* M^t_{XY} *from the domain of our Example 2.1 act as activator random variables according to Definition 2.2.*

We observe possible message exchanges from used envelopes (possibly found in the trash bin). On internal envelopes one usually finds multiple transfers from a coarse time interval in an imprecise or inaccurate order. For example, a transfer from Don to Earl and one from Claire to Don might include a transitive influence of Claire on Earl during the same time interval. *We show in the following that it is highly important to cover these indirect influences and to model indirect causes appropriately.*

Example 2.2 shows that activator random variables naturally exist in domains and do not necessarily need be introduced as auxiliary variables. If activator random variables are present in domains, an (effective) structure of an DPGM is not known in advance and even changes over time, i.e., changes at every timeslice when using the model. As effective structures are not known in advance, i.e., are not known when creating the model, only general structures are designable in advance covering all potential substructures. To correctly cover all implications of influences, i.e., to consider all direct and indirect causes, generally cyclic DPGMs are required (as in Figure 2.2, black).

Table 2.1. *Example for a CPD* $P(+x|V,W,Y,Z)$ *with arbitrary numbers* α*—*γ*. In fact, random variables* V *and* W *are identifiable as activator random variables according to Definition 2.2.* V *represents an* A_{YX} *and* W *represents an* A_{ZX}*.*

V	**W**	**Y**	**Z**	$\mathbf{P}(+x\|\ldots)$
$+v$	$+w$	$+y$	$+z$	α
$+v$	$\neg w$	$+y$	$+z$	β
$+v$	$\neg w$	$+y$	$\neg z$	β
$\neg v$	$+w$	$+y$	$+z$	ϵ
$\neg v$	$+w$	$\neg y$	$+z$	ϵ
$+v$	$+w$	$\neg y$	$+z$	η
$+v$	$+w$	$+y$	$\neg z$	γ
$+v$	$+w$	$\neg y$	$\neg z$	μ
$+v$	$\neg w$	$\neg y$	$+z$	ν
$+v$	$\neg w$	$\neg y$	$\neg z$	ν
$\neg v$	$\neg w$	$+y$	$+z$	φ
$\neg v$	$\neg w$	$+y$	$\neg z$	φ
$\neg v$	$\neg w$	$\neg y$	$+z$	φ
$\neg v$	$\neg w$	$\neg y$	$\neg z$	φ
$\neg v$	$+w$	$+y$	$\neg z$	ψ
$\neg v$	$+w$	$\neg y$	$\neg z$	ψ

2.2 Activator Dynamic Bayesian Networks

A DPGM in which some random variables are identifiable as activator random variables according to Definition 2.2 is called an Activator DBN, i.e., random variables in B_0 and $B_\rightarrow$ can syntactically be grouped into two (not necessarily disjoint) sets of state variables $\vec{X}^t$ and activator variables $\vec{A}^t$.

Notation 2.3 (Activator matrices). *Let $A^{s\,t}_{ij}$ be the activator random variable influencing X^t_j regarding a dependency on X^s_i. Let $A^{s\,t}$ describe the matrix of all activator random variables between time-slice s and t s.t.*

$$A^{s\,t} = \begin{pmatrix} A^{s\,t}_{11} & \cdots & A^{s\,t}_{1n} \\ \vdots & \ddots & \vdots \\ A^{s\,t}_{n1} & \cdots & A^{s\,t}_{nn} \end{pmatrix} .$$

Let $\vec{A}^{s\,t}_i$ denote the i^{th} column of $A^{s\,t}$, i.e., $\vec{A}^{s\,t}_i$ represents the vector of all activator random variables relevant for X^t_i regarding an influence by random variables of timeslice s. Let $\vec{\mathcal{A}}^{s\,t}$ denote the corresponding column vector of all entries of $A^{s\,t}$, i.e.,

$$\vec{\mathcal{A}}^{s\,t} = \left(A^{s\,t}_{11}, \ldots, A^{s\,t}_{1n}, \ldots, A^{s\,t}_{n1}, \ldots, A^{s\,t}_{nn}\right)^{\mathsf{T}} .$$

Let $\vec{\mathcal{A}}^{01:tt} = \langle \vec{\mathcal{A}}^{01}, \vec{\mathcal{A}}^{11}, \vec{\mathcal{A}}^{12}, \ldots, \vec{\mathcal{A}}^{tt} \rangle$ denote the vector of all activator random variables existing in and between timeslices 0 to t. For brevity, we write A^t for A^{tt} (excluding A^{tt}_{kk} s.t. X^{tt}_k cannot be dependent on itself), and correspondingly we write A^t_{ij}, $\vec{A}^t_i$ and $\vec{\mathcal{A}}^t$. Let $\vec{\mathcal{A}}^{1:t} = \langle \vec{\mathcal{A}}^1, \vec{\mathcal{A}}^2, \ldots, \vec{\mathcal{A}}^t \rangle$ then denote the vector of all intra-timeslice activator random variables $\vec{\mathcal{A}}^t$ for every timeslice t.

Activator dynamic Bayesian networks are DPGMs and share familiar syntax and semantics with dynamic Bayesian networks (compare Definition 2.1 and Proposition 2.1), but ADBNs are not bound to DAG constraints like DBNs:

Definition 2.3 (Activator dynamic Bayesian network, ADBN). *An ADBN is syntactically defined as a tuple $(B_0, B_\rightarrow)$ with B_0 defining an initial Bayesian network representing time $t = 0$ containing all states $X^0_i \in \vec{X}^0$ and a consecutively repeated activator Bayesian network fragment $B_\rightarrow$ consisting of dependencies between state variables X^s_i and X^t_j, $t - 1 \leq s \leq t$ (Markov-1) and dependencies between state variables X^t_i and activator random variables A^{st}_{ji}. For every random variable X^t_i, A^{st}_{ij} a local CPD over all parents, e.g., as a CPT is specified, where CPDs of state variables X^t_i follow Definition 2.2.*

By repeating $B_\rightarrow$ for every time step $t > 0$, an ADBN $(B_0, B_\rightarrow)$ is unfolded into a PGM defining an ADBN's semantics. ▲

Definition 2.3 defines ADBNs as a form of DPGM in which some random variables have an activator nature, i.e., some CPDs of state variables show certain properties that identify other random variables as activator random variables. Note that these activator random variables are *not* introduced externally, but are already part of a domain as shown in the following examples. The following theorem states in which cases an ADBN is well-defined. The idea is that two cases may exist: (1) an ADBN is a classical DBN, i.e., acyclic and well-defined by Proposition 2.1, or (2) an ADBN is not a classical DBN, i.e., it contains cyclic dependencies, for which a different well-definedness condition is introduced, which is a modified constraint of acyclicity.

Theorem 2.1 (ADBN well-definedness)**.** *An ADBN is well-defined if an ADBN is well-defined according to Proposition 2.1, i.e., is a well-defined DBN. An ADBN is well-defined for every instantiation $\vec{a}_*^{1:t}$ of $\vec{\mathcal{A}}^{1:t}$ if for all t, $\vec{a}_*^t$ satisfy the following conditions:*

$$\begin{aligned} \forall x, y, z \in \vec{X}^t : \mathfrak{A}(x,z)^t, \mathfrak{A}(z,y)^t \rightarrow \mathfrak{A}(x,y)^t \\ \neg\exists q : \mathfrak{A}(q,q)^t \ , \end{aligned} \tag{2.3}$$

with an acyclicity predicate $\mathfrak{A}(i,j)^t$ that is defined as

$$\mathfrak{A}(i,j)^t = \begin{cases} \textit{false} & \textit{if} \quad \neg a_{ij}^t \in \vec{a}_*^t \\ \textit{true} & \textit{if} \quad \textit{else} \end{cases} .$$

For every well-defined ADBN, semantics as $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{01:tt^\intercal})$ is sound and given equivalently to DBN semantics as the product of all locally defined CPDs.
▲

Theorem 2.1 means that ADBNs are not syntactically bound to DAG structures in $B_\rightarrow$ and still share familiar DBN semantics providing a local interpretation and specification of conditional probability distributions without the need of global normalization factors such as, e.g., required for conditional random fields. Proof of Theorem 2.1 is later given in Appendix B. Well-definedness is achieved through the novel acyclicity constraint, namely that not a syntactic graph structure is forced to be acyclic, but rather that an *instantiation* of an unrolled DBN is acyclic. This is useful for applications where actual structures are not known in advance and might change at every timeslice, depending on specific contexts. Such particular situations often arise if timeslices are not supposed to be infinitesimal small s.t. cyclic dependency structures in $B_\rightarrow$ arise naturally during a design phase s.t. indirect influences must be considered during usage of the model. In particular, the novel acyclicity constraint of ADBNs is highly beneficial for the running example, as demonstrated in Example 2.3 after the following definitions.

The novel acyclicity constraint means that it has to be enforced that only well-defined instantiations are used for inference. Enforcing the use of only well-defined instantiations can be ensured by adequate observations or modeling approaches (see later Chapter 5). We explicitly name such instantiations leading to well-defined ADBNs using the following definition.

Definition 2.4 (Regular and acyclic instantiations)**.** *If an instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ of $\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t}$ leads to a well-defined ADBN according to Theorem 2.1, we say that the instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ is* regular*. Let G_A^t represent a graph formed by active activators of a full instantiation $\vec{a}^t$, where every active activator $^+a_{ij}^t \in \vec{a}^t$ represents an edge from node i to j. If, for every $\vec{a}^i$ in $\vec{a}^{1:t}$, G_A^i is acyclic, we say that the instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ is* acyclic*.* ▲

Under Definition 2.4 every acyclic instantiation is regular. For the predicate $\mathfrak{A}$ defined in Theorem 2.1 every regular instantiation is also an acyclic instantiation. Later, in Chapter 4 we show that $\mathfrak{A}$ predicates exist for which regular instantiations do not necessarily need to be acyclic, i.e., not only contexts of activator random variables can assure regularity.

Corresponding to Notation 2.2 regarding inter- and intra-timeslice models, we distinguish ADBNs based on the density of their activator matrices:

Notation 2.4 (Dense, inter- and intra-timeslice ADBNs). *An* intra-timeslice *ADBN $(B_0, B_\rightarrow)$ is an ADBN with non-empty activator matrix $A^{t\,t}$. A* diagonal *or* inter-timeslice *ADBN $(B_0, B_\rightarrow)$ is an ADBN non-empty $A^{t-1\,t}$. An ADBN $(B_0, B_\rightarrow)$ is called a* dense *ADBN, if at least one activator matrix $A^{s\,t}$ is dense, i.e., there exists a random variable acting as an activator random variable for every modeled influence. In a dense intra-timeslice model, cyclic dependencies exist in $B_\rightarrow$, for which we also speak of* cyclic *ADBNs.*

Note again that activators in an ADBN are classical random variables and are part of a modeled domain, i.e., activators are not necessarily auxiliary variables. The following example shows that activator random variables are already present in the taintedness domain.

Example 2.3 (Inter- and intra-timeslice ADBNs). *Based on Example 2.2 the regulatory compliance domain can be modeled as an ADBN: Message exchange variables have activator nature and one obtains an ADBN with activators $\vec{A}^t = (M^t_{CD}, M^t_{DC}, M^t_{DE}, M^t_{ED}, M^t_{CE}, M^t_{EC})^\intercal$ and state random variables $\vec{X}^t = (C^t, D^t, E^t)^\intercal$.*

To model that every employee can potentially influence every other*, i.e., send him a document two options are available: (a) a diagonal (dense) inter-timeslice ADBN as shown in Figure 2.2 (gray), i.e., a well-defined DBN according to Proposition 2.1 or (b) a cyclic (dense) intra-timeslice ADBN as shown in Figure 2.2 (black), i.e., a* novelly *well-defined ADBN according to Theorem 2.1, if correctly instantiated.*

This example shows that activator random variables may already be part of a domain, and are not introduced externally. As discussed earlier, this example highlights both design options for the discussed example domain of an ADBN: one classic solution based on a diagonal ADBN and one using a newly allowed cyclic ADBN. As a diagonal (dense inter-timeslice) ADBN represents a classical DBN, we also write (A)DBN in the diagonal case. Note that the diagonal (A)DBN solution is the *only* consequent well-defined classical DBN solution[1]. Note that in this example, all $n^2 - n$ activator random variables are present leading to a dense intra- or inter-timeslice ADBN, but not always all activator random variables must be present. However, as discussed earlier, a diagonal option is seen as *inaccurate* from a world representation point of view and is subject to a significant problem with indirect causes, as pinpointed in the following example.

Example 2.4 (Example for diagonal (A)DBN restrictions). *Continuing Example 2.3, one observes a message transfer from Claire to Don ($^+m^1_{CD}$) and from Don to Earl ($^+m^1_{DE}$), say, and one can neglect all other transfers, i.e., $\neg m^1_{DC}$, $\neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC}$. To fully evaluate all implications of the observations, one has to anticipate an indirect influence from Claire to Earl through Don during timeslice* 1*. But, the diagonal (A)DBN in Figure 2.2 (gray) does not encode the domain correctly and leads to spurious results: Earl is only influencable by Claire* through *Don from a Claire of the penultimate timepoint, but the observed message exchanges only let t_1-Claire influence t_2-Don, and let t_1-Don influence t_2-Earl. Cf. later Examples 2.6 and 2.7.*

[1]A limited set of dependencies can be modeled as intra-timeslice dependencies, but it is an arbitrary decision why some dependencies are modeled as inter- and some are modeled as intra-timeslice dependencies. Cf. Section 3.2.

Example 2.4 shows that classical, diagonal DBNs are limited, as some observations may lead to spurious results. Due to an enforced bend of influences between timeslices in diagonal models, which is required to assure the acycliciy constraint of Proposition 2.1, one has to conclude that diagonal models (as a representative of classic DBNs) cannot anticipate indirect influences and are limited to reasoning using only direct dependencies.

Proposition 2.2 (Diagonal (A)DBN restrictions). *A diagonal inter-timeslice (A)DBN is restricted to certain instantiation sets. Indirect influences are spread over multiple timesteps and possible indirect influences inside one timestep cannot be considered. This enforces (a) an infinitesimal resolution of timeslices, where indirect effects do not need to be anticipated, or (b) restricts a DBN to instantiations where indirect influences strictly do not occur. We say that a diagonal (A)DBN is restricted to* indirect-free *instantiations. This forbids that any two activators A^t_{*i} and A^t_{i*} are probably active, i.e., the set of probably active activators must form a bipartite digraph with uniformly directed edges. This means that it must be enforced that only specific instantiations of a diagonal (A)DBN are instantiated. Further, only up to $n^2/4$ activators are allowed to be probably active per timestep, and* all *other activators must be* observed *to be* deactive*. If, in a diagonal DBN, observations can neither fulfill (a) nor (b), observation- and query-(de)serializations would be needed, and $n-2$ spurious "time"-slices would need to be inserted between $t-1$ and t. In our opinion, this degrades a BN to a reasoning tool.* ▲

Proposition 2.2 summarizes that DBNs are significantly limited in their expressiveness and are restricted to certain instantiations, namely, indirect-free instantiations.

However, Theorem 2.1 explicitly permits structures of $B_{\rightarrow}$ that are forbidden under classic DBN definitions: cyclic intra-timeslice models, as discussed in Example 2.3, are well-defined under some restrictions according to Theorem 2.1. In the following we show that restrictions of intra-timeslice models are softer than restrictions on classic, diagonal (A)DBN models:

Example 2.5 (Example for Cyclic (A)DBN benefits). *Continuing Ex. 2.4, with the same message transfer observations, but for a cyclic ADBN option (Figure 2.2, black): The observations only allow regular and acyclic instantiations which satisfy Theorem 2.1 and thus lead to a well-defined Bayesian network, even though $B_{\rightarrow}$ is based on a cyclic graph. Here, all implications of the observations, namely an indirect influence from Claire to Earl through Don during timeslice* 1 *is anticipated. As intended, t_1-Claire does influence t_1-Don of the same timeslice, and t_1-Don influences t_1-Earl in the same timeslice. To achieve this constellation in a general setting, $B_{\rightarrow}$ must proactively be designed with cycles, as actual dependencies are not known in advance and are changing over time.*

This example shows that in ADBNs indirect influences are anticipated under a time-granularity suited to a problem (e.g., days) instead of a time-granularity enforced by a reasoning framework. Essentially, ADBNs move an acyclicity constraint for well-definedness from a design phase of a Bayesian network to an actual instantiation of a Bayesian network. Example 2.5, further, shows how observations assure regularity. Still, situations may exist where not sufficient observations are made and one cannot preclude a non-regular instantiation; this situation, its implications, and possible mitigations are discussed in Section 2.5 and Chapter 5.

To summarize, we have shown that classic (A)DBNs are limited in their expressiveness and that instantiations of them are limited to indirect-free instantiation sets that do not contain indirect influences, if a time granularity is not infinitesimally small. With the identification of activator properties of random variables in ADBNs, more expressive graph structures for (A)DBNs are supported, share similar semantics, and larger sets of regular instantiations of ADBNs. Later, in Section 2.5, we numerically compare restrictions on the number of regular instantiations between diagonal DBNs and cyclic ADBNs to substantiate that restrictions of cyclic ADBNs are far softer than the ones opposed on diagonal DBNs. Nevertheless, situations may exist where insufficient observations are made to fulfill regularity, either that too many active activators are observed, or that too few deactive activators are observed. Both situations are discussed in Section 2.5 and, later, motivate Chapter 5.

In fact, every possible Markov-1 DBN can be represented as a dense inter-and-intra timeslice ADBN:

Proposition 2.3 (Completeness)**.** *An ADBN can model any joint probability and includes every possible Markov-1 DBN structure. Let $(B_0^*, B_\to^*)$ be a dense inter- and intra-timeslice ADBN with a dense inter-timeslice activator matrix $A^{t-1\,t}$ and a dense intra-timeslice activator matrix $A^{t\,t}$. Then, for every Markov-1 DBN $(B_0', B_\to')$ there exists a minimal set of instantiations $\vec{a}_\blacklozenge$ of $(B_0^*, B_\to^*)$ under which the same effective topological ordering is formed as defined by $(B_0', B_\to')$. Therefore, a dense inter-timeslice ADBN represents a superclass of all possible inter-timeslice DBN structures and a dense intra-timeslice ADBN represents a superclass of all possible intra-timeslice DBN structures.* ▲

We argue that often DBN models with diagonal state dependencies are used only due to syntactic constraints on (D)BNs (Proposition 2.1) and stand in conflict with an actual causality in their domain. Therefore, we focus in the following on a dense intra-timeslice ADBN, which includes and represents all possible cases of ADBNs that were not modellable in a classical DBN formalism, i.e., all intra-timeslice cycles. The following proposition derives the semantics according to Theorem 2.1 as the joint probability distribution over all random variables for a dense intra-timeslice ADBN.

Proposition 2.4 (Joint probability distribution of a dense intra-timeslice ADBN)**.** *If an ADBN is well-defined, the joint probability over all random variables is defined as the product of all locally defined CPDs. Therefore, a dense intra-timeslice ADBN's semantics is*

$$\begin{aligned}P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = P(X_1^0)\cdot\ldots\cdot P(X_n^0)\cdot\prod_{i=1}^{t} P(X_1^i|X_2^i,\ldots,X_n^i,A_{21}^i,\ldots,A_{n1}^i,X_1^{i-1})\\ \cdot\ldots\cdot P(X_n^i|X_1^i,\ldots,X_{n-1}^i,A_{1n}^i,\ldots,A_{(n-1)n}^i,X_n^{i-1})\cdot P(A_{12}^i)\cdot\ldots\cdot P(A_{n(n-1)}^i)\\ = \prod_{X_k^0\in\vec{X}^0} P(X_k^0)\cdot\prod_{i=1}^{t}\prod_{X_k^i\in\vec{X}^{\,i}} P(X_k^i|\vec{X}^{i^\intercal}\setminus X_k^i,\vec{A}_k^{i^\intercal},X_k^{i-1})\cdot\prod_{A_{cv}^i\in\vec{\mathcal{A}}^i} P(A_{cv}^i)\ ,\end{aligned} \tag{2.4}$$

As expected, the joint probability can be defined recursively:

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) =$$
$$P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal}) \cdot \prod_{X_k^t \in \vec{X}^{\,t}} P(X_k^t | \vec{X}^{t^\intercal} \backslash X_k^t, \vec{A}_k^{t^\intercal}, X_k^{t-1}) \cdot \prod_{A_{cv}^t \in \vec{\mathcal{A}}^t} P(A_{cv}^t) \ . \quad \blacktriangle \qquad (2.5)$$

Naively, every query to an (A)DBN can be answered by marginalization from the defined joint probability distribution over all random variables from all timeslices, which, however, is computationally intractable. Therefore, the following sections discuss special types of queries and derive exact and approximate solutions to associated problems in cyclic (A)DBNs.

2.3 Common Queries and Associated Answering Problems in ADBNs

A DBN is a temporal probabilistic knowledge base with respect to which queries can be posed. Common query types are known as *filtering*, *smoothing* and *most likely explanation* (cf. Murphy, 2002) and define query answering problems w.r.t. the semantics of the knowledge base. Queries are used to investigate on historical information in retrospect or are used to constantly monitor a specific variable over time, e.g., estimate a trajectory of a moving object for which position measurement values are noisy.

Naively, every query is answered by straight marginalization from the full JPD defined be the unrolled DBN, i.e., from the JPD defined by a Bayesian networks consisting of all timeslices at once. However, such a naive approach is only tractable for the very first timeslices and the curse of dimensionality prevents inference over long periods of time. Still, it is a highly important property of DBNs that commonly known problems, such as filtering- and smoothing-problems, remain practically solvable even over long periods of time via commonly known algorithms such as the forward-backward-algorithm. Without restriction of any kind this property must be preserved in any novel dynamic probabilistic model, such as the one introduced by us—ADBNs. Therefore, we prove in this section that inference in ADBNs remains in the same complexity class as corresponding problems in multiply connected DBNs do. In general, inference problems in multiply-connected (D)BNs are known to be NP-hard as shown by Cooper (1990). However, focusing on single input parameters of inference problems associated with DBNs, problems show to be fixed-parameter tractable. This means that algorithms exist, e.g., the forward-backward-algorithm, which provide exact solutions to common query answering problems in a tractable time- and memory-complexity over time, i.e., remain tractable even for large numbers of consecutive timeslices. In this section we show and prove that inference in ADBNs does not introduce any overhead for solving commonly known problems, that solutions, such as the forward-backward-algorithm remain directly applicable for ADBNs based on cyclic graphs, and that inference in ADBNs remains in a classical and familiar calculus as DBNs without any introduction of novel operators, i.e., one preserves classical marginalization from a JPD based on classical random variables with associated CPDs.

We denote observations of random variables in probabilistic knowledge bases and formulate queries to these probabilistic knowledge bases based on these observations using the following definition.

Definition 2.5 (Query answering language and observations)**.** *Let $\vec{Z}^t \subseteq \vec{X}^t$ be a set of observed and $\vec{\zeta}^t = \vec{X}^t \backslash \vec{Z}^t$ be the corresponding set of not-observed state variables. Let $B^t \subseteq A^t$ be a set of observed activators and $\vec{\mathcal{B}}^t \subseteq \vec{\mathcal{A}}^t$ be the corresponding column vector representation. Likewise, let $\vec{\beta}^t = \vec{\mathcal{A}}^t \backslash \vec{\mathcal{B}}^t$ be the column vector of all unobserved activators. Then, observations of state variables $\vec{z}^t$ are instantiation assignments $X_i^t = x_i \in \mathrm{dom}(X_i^t)$ and observations of activator random variables $\vec{b}^t$ are instantiation assignments $A_{ij}^t = a_{ij} \in \mathrm{dom}(A_{ij}^t)$.*

Let $\vec{x}^t \in \mathrm{dom}(\vec{X}^t)$ be a full instantiation of $\vec{X}^t$ and let $\vec{a}^t \in \mathrm{dom}(\vec{\mathcal{A}}^t)$ be a full instantiation of $\vec{\mathcal{A}}^t$. Further, let every instantiation assignment in $\vec{x}^t, \vec{z}^t, \vec{a}^t, \vec{b}^t$ uniquely define the value of its respective random variable in $\vec{X}^t$ or $\vec{\mathcal{A}}^t$.

Then, a query to a probabilistic knowledge base defined as $(B_0,\, B_{\rightarrow})$ is a request for a result of $P(\vec{x}^{k^\intercal}, \vec{a}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$, i.e., the probability of a full instantiation $\vec{x}^k, \vec{a}^k$ at an arbitrary timestep k, given (partial) evidence $\vec{z}^{0:t}, \vec{b}^{1:t}$ until an arbitrary timestep t. The answer to a query for a probability of an instantiation contradicting observations, i.e., an observed state variable $X_i^k \in \vec{Z}^k$ is contradictorily defined by a $z_j^k \in \vec{z}^k$ and $x_i^k \in \vec{x}^k$ or an observed activator random variable $A_{ij}^k \in \vec{\mathcal{B}}^t$ is contradictorily defined by a $b_{cv}^k \in \vec{b}^k$ and $a_{ij}^k \in \vec{a}^k$, is defined to be of probability zero, e.g., $P(+c, +d, +e | \neg d) = 0$. A query for an uninstantiated set of random variables is a query for a distribution $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$ and is answerable by queries for all possible instantiations of $\vec{X}^k, \vec{\mathcal{A}}^k$. ▲

Answers to queries defined by Definition 2.5 are obtained by probabilistic inference in the knowledge base. As discussed earlier, inference must be based solely on regular instantiations. If observations are supposed to enforce regularity, we talk about regular or acyclic observations.

Definition 2.6 (Regular and acyclic observations)**.** *An observation is regular/acyclic, if every instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ for which $P(\vec{x}^{0:t^\intercal}, \vec{a}^{1:t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) > 0$ holds is regular/acyclic.* ▲

Notation 2.5 (Summation over uninstantiated random variables)**.** *Let R be an uninstantiated, i.e., an unobserved and unqueried, random variable. Then let $\sum_R P(R)$ denote a summation over all possible instantiations r of R, $r \in \mathrm{dom}(R)$, e.g., for a binary R: $\sum_R P(R) = P(+r) + P(\neg r)$. Let $\vec{R}$ denote a column vector of uninstantiated random variables $R_i \in \vec{R}, 0 \leq i \leq n$. Then let $\sum_{\vec{R}} P(R_0, R_1, \ldots R_n)$ denote a nested summation over all uninstantiated random variables $\sum_{R_i \in \vec{R}}$, i.e., a summation over all possible instantiation combinations of all random variables in $\vec{R}$. For example, for binary $\vec{R} = \langle R_0, R_1 \rangle$: $\sum_{\vec{R}} P(R_0, R_1) = P(+r_0, +r_1) + P(+r_0, \neg r_1) + P(\neg r_0, +r_1) + P(\neg r_0, \neg r_1)$. Respectively, this notation applies to* $\arg\max$.

Definition 2.5 defines queries about full instantiations of all random variables in a timeslice. Queries for partial subsets, e.g., distributions of single random variables, are answerable using the following proposition.

Proposition 2.5 (Answering queries about partial subset of instantiations)**.** *All queries about partial subsets of instantiations $\vec{x}_q^k$, $\vec{a}_q^k$ of random variables $\vec{X}_q^k \in \vec{X}^k$ and $\vec{\mathcal{A}}_q^k \in \vec{\mathcal{A}}^k$ are answerable by marginalization from $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$ as*

$$P(\vec{x}_q^{k^\intercal}, \vec{a}_q^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) = \sum_{\vec{X}_u^k} \sum_{\vec{\mathcal{A}}_u^k} P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) \ ,$$

where $\vec{\mathcal{A}}_u^k$ is the vector of unqueried activator variables $\vec{\mathcal{A}}_u^k = \vec{\mathcal{A}}^k \backslash \vec{\mathcal{A}}_q^k$ and $\vec{X}_u^k$ is the vector of unqueried state variables $\vec{X}_q^k = \vec{X}^k \backslash \vec{X}_q^k$. Given the distribution $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal})$, marginalization is exponential in the dimension of unqueried variables from timestep k and in the largest domain $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ *of unqueried random variables $X_+ \in \vec{X}_u^k$, $A_{++} \in \vec{\mathcal{A}}_u^k$, i.e., in $\mathcal{O}(|\mathrm{dom}(X_+)|^{|\vec{X}_u^k|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}_u^k|})$.* ▲

2.3.1 Filtering Queries and Problems

Queries $P(\vec{x}^{k^\intercal}, \vec{a}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal})$ with $k = t$ to a probabilistic knowledge base are commonly known as filtering queries. Finding answers to queries of the type $P(\vec{x}^{t^\intercal}, \vec{a}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal})$ poses two different filtering problems.

Given historical information about a probabilistic process, filtering queries for multiple timeslices from a broad timerange considering long periods of evidences need to be answered. To answer these queries efficiently, the offline filtering problem needs to be solved.

Definition 2.7 (Offline filtering problem)**.** *Given a probabilistic knowledge base B_0, $B_\rightarrow$, the offline filtering problem is the task of determining the conditional probability of random variables at all timeslices $0 \leq j \leq t$ given all obtained evidence $\vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}$ so far. This is, to obtain*

$$P(\vec{X}^{j^\intercal}, \vec{\mathcal{A}}^{j^\intercal} | \vec{z}^{0:j^\intercal}, \vec{\mathscr{b}}^{1:j^\intercal}) \ , \forall j : 0 \leq j \leq t \ .$$

We denote a parametrized offline filtering problem as `OffFP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$, *where t represents the (unary coded) total number of passed timeslices so far.* ▲

The offline filtering problem determines the complete conditional joint probability distribution $P(\vec{X}^{j^\intercal}, \vec{\mathcal{A}}^{j^\intercal} | \vec{z}^{0:j^\intercal}, \vec{\mathscr{b}}^{1:j^\intercal})$ at every timestep j, called a filtering distribution, such that every filtering query for a timeslice j can be answered directly. Further, an answer to filtering query about partial instantiations can be derived by marginalization, without determining the complete filtering distribution again.

If knowledge about a distribution's evolution over time is required incrementally from $t-1$ to t for every newly obtained evidence, i.e., queries need to be answered temporally consecutively only for the current timeslice t, one has to solve the online filtering problem.

Definition 2.8 (Online filtering problem)**.** *Given a probabilistic knowledge base B_0, $B_\rightarrow$ and a stored solution to the online filtering problem at $t-1$, e.g., $P(\vec{X}^{t-1^\intercal}, \vec{\mathcal{A}}^{t-1^\intercal} | \vec{z}^{0:t-1^\intercal}, \vec{\mathscr{b}}^{1:t-1^\intercal})$, the online filtering problem is the task of determining the conditional probability distribution of random variables at time t given newly obtained evidence $\vec{z}^t, \vec{\mathscr{b}}^t$. This is, to obtain*

$$P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) \ ,$$

conditioned on the stored solution to the online filtering problem at $t-1$. We denote a parametrized online filtering problem as `OnlFP`$(B_0, B_\rightarrow, \vec{z}^{t-1:t}, \vec{\mathscr{b}}^{t-1:t}, t)$, *which includes a solution to* `OnlFP`$(B_0, B_\rightarrow, \vec{z}^{t-2:t-1}, \vec{\mathscr{b}}^{t-2:t-1}, t-1)$. ▲

The following example demonstrates how a filtering query can be used to gain knowledge about potential infringements in the running example.

Example 2.6 (Filtering)**.** *With Theorem 2.1 one can actually model cyclic dependencies as desired in Example 2.3 and build an ADBN for our example as shown in Figure 2.2 (black). We assume a noisy-or combination for every state* X^t *and an individual probability of influence of* 0.8.

Say, Don and Earl did pass an initial checkup, but Claire did not. At $t = 1$, *one observes a document transfer from Claire to Don, one is unsure about one from Don to Earl, i.e.,* M_{DE} *is uninstantiated, and one can neglect all other transfers. As Claire is known to be tainted, one expects her to influence Earl* through *Don, analyzable in a filtering query* $P(E^1|\vec{z}^{0:1^\intercal}, \vec{b}^{1^\intercal})$, *with* $\vec{z}^{0:1} = (+c^0, \neg d^0, \neg e^0)^\intercal$, *and* $\vec{b}^1 = (+m^1_{CD}, \neg m^1_{DC}, \neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC})^\intercal$. *As* $\vec{b}^1$ *is regular, one obtains* $P(E^1|\vec{z}^{0:1^\intercal}, \vec{b}^{1^\intercal}) = \langle 0.8 \cdot 0.8 \cdot 0.5, 1 - 0.8 \cdot 0.8 \cdot 0.5\rangle = \langle 0.32, 0.68\rangle$ *using a cyclic ADBN, i.e., a cyclic ADBN considers that Earl is possibly influenced by Claire through Don and there exists a probability of* 32% *that Earl is now tainted. A diagonal DBN cannot anticipate the indirect influence, because* t_1*-Earl is influenced by a* t_0*-Don that has not received a document from Claire. Therefore, a diagonal DBN assures that Earl is* not *tainted, i.e.,* $P(E^1|\vec{z}^{0:1^\intercal}, \vec{b}^{1^\intercal}) = \langle 0, 1\rangle$ *and there is absolutely no possibility that Earl is tainted.*

To obtain a correct answer in a diagonal DBN for this situation, one needs observations at a finer time scale where indirect influences are not evident during one timeslice. For example, one first has to observe m^1_{CD}, *considered in an evaluation of* $P(E^1|\vec{z}^{0:1^\intercal}, \vec{b}^{1^\intercal})$, *and then insert a "correcting" "time"-slice* $t = 1.1$, *where possibilities of* M^1_{DE} *are considered during the evaluation of* $P(E^{1.1}|\vec{z}^{0:1.1^\intercal}, \vec{b}^{1:1.1^\intercal})$. *To achieve the result of one ADBN evaluation, one needs* $n-1$ *"diagonal"-evaluations. Somehow, a reasoning framework would need to map queries to serialized observations in a diagonal DBN. Introducing obscure "correcting" time-slices in this manner degrades a BN to a reasoning tool.*

This example demonstrates that it is highly important to consider indirect causes, as otherwise highly unexpected results are obtained. Moreover, it demonstrates that by the use of cyclic ADBNs, indirect causes are correctly represented and anticipated, in contrast to a diagonal ADBN. The correct anticipation of indirect influences is achieved by novelly allowed cyclic dependencies in an ADBN, which, as stated by the following theorems, do not cause any overhead in solving associated filtering problems.

Theorem 2.2 (Exact solution to the offline filtering problem)**.** *Given an offline filtering problem* $\texttt{OffFP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$, *finding an exact solution is linear in* t. *Finding an exact solution is exponential in the maximal dimension of unobserved variables* $\vec{\zeta}^*$, $\vec{\beta}^*$ *in a timestep* $0 < * < t$, *and in the largest domain* $\mathrm{dom}(\zeta_+)$, $\mathrm{dom}(\beta_+)$ *of all random variables* $\zeta_+ \in \vec{\zeta}^{0:t}$, $\beta_+ \in \vec{\beta}^{1:t}$. *Finding an exact solution to the offline filtering problem is exponential in the dimension of number of random variables* $|\vec{X}^t|$, $|\vec{\mathcal{A}}^t|$ *and a respective maximal domain size* $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ *of all random variables* $X_+ \in \vec{X}^t$, $A_{++} \in \vec{\mathcal{A}}^t$. ▲

Theorem 2.2 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{OffFP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$ in time-complexity $\mathcal{O}(t \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\mathrm{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\mathrm{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with space-complexity $\mathcal{O}(t \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ for storing $P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal}|\vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$ for every timeslice t. The proof is combined with the proof for the following theorem.

Theorem 2.3 (Exact solution to the online filtering problem)**.** *Given an online filtering problem* $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{t-1:t}, \vec{b}^{t-1:t}, t)$*, finding an exact solution is constant in* t*. Finding an exact solution is exponential in the dimension of unobserved variables from timestep* $t-1$ *and in the largest domain* $\mathrm{dom}(\zeta_+)$, $\mathrm{dom}(\beta_+)$ *of random variables* $\zeta_+ \in \vec{\zeta}^{t-1}$, $\beta_+ \in \vec{\beta}^{t-1}$*. Finding an exact solution to the online filtering problem is exponential in the dimension of number of random variables* $|\vec{X}^t|$, $|\vec{\mathcal{A}}^t|$ *and a respective maximal domain size* $\mathrm{dom}(X_+)$, $\mathrm{dom}(A_{++})$ *of all random variables* $X_+ \in \vec{X}^t$, $A_{++} \in \vec{\mathcal{A}}^t$*.* ▲

Theorem 2.3 is proven by showing that an algorithm algorithm exists that finds an exact solution with time-complexity $\mathcal{O}(|\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\mathrm{dom}(\zeta_+^{t-1})|^{|\vec{\zeta}^{t-1}|} \cdot |\mathrm{dom}(\beta_+^{t-1})|^{|\vec{\beta}^{t-1}|})$ and space-complexity $\mathcal{O}(|\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$.

Theorems 2.2 and 2.3 discuss the computational complexity of filtering problems in general ADBNs, and most importantly state that filtering in ADBNs still remains constant over time, as one is used to for classical DBN filtering problems. Both theorems are proven below, where a major result is a commonly known recursive definition of a filtering equation for solving filtering problems.

Proof of Theorems 2.2 and 2.3 (Solutions to filtering problems). Filtering is generally defined from the joint probability as

$$P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) = \alpha \cdot \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) \ , \tag{2.6}$$

with a normalization factor α s.t. $\sum_{\vec{X}^t, \vec{\mathcal{A}}^t} P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) = 1$, as usual. In Eq. 2.6 all values of variables $\vec{X}^t, \vec{\mathcal{A}}^t$ are defined by the query and variables $\vec{X}^{0:t-1}, \vec{\mathcal{A}}^{1:t-1}$ are defined by either observations in the sets $\vec{z}^{0:t-1}, \vec{b}^{1:t-1}$ or through summation over unobserved variables in $\vec{\zeta}^{0:t-1}, \vec{\beta}^{1:t-1}$.

Using the recursive definition of the joint probability in Equation 2.5, one obtains

$$\begin{aligned}
&P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) \\
&= \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal}) \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t^\intercal} \backslash X_i^t, \vec{A}_i^{t^\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&= \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} \sum_{\vec{\zeta}^{0:t-2}} \sum_{\vec{\beta}^{1:t-2}} P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal}) \\
&\qquad \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t^\intercal} \backslash X_i^t, \vec{A}_i^{t^\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&= \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t^\intercal} \backslash X_i^t, \vec{A}_i^{t^\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&\qquad \cdot \sum_{\vec{\zeta}^{0:t-2}} \sum_{\vec{\beta}^{1:t-2}} P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal}) \\
&= \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t^\intercal} \backslash X_i^t, \vec{A}_i^{t^\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \\
&\qquad \cdot P(\vec{X}^{t-1^\intercal}, \vec{\mathcal{A}}^{t-1^\intercal} | \vec{z}^{0:t-1^\intercal}, \vec{b}^{1:t-1^\intercal}) \ .
\end{aligned}$$

One obtains the general filtering equation as

$$
\begin{aligned}
P(\vec{X}^{t\intercal}, \vec{\mathcal{A}}^{t\intercal} | \vec{\mathscr{z}}^{0:t\intercal}, \vec{\mathscr{b}}^{1:t\intercal}) = \alpha \sum_{\vec{\zeta}^{t-1}} \sum_{\vec{\beta}^{t-1}} & P(\vec{X}^{t-1\intercal}, \vec{\mathcal{A}}^{t-1\intercal} | \vec{\mathscr{z}}^{0:t-1\intercal}, \vec{\mathscr{b}}^{1:t-1\intercal}) \\
& \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t\intercal} \setminus X_i^t, \vec{A}_i^{t\intercal}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \ . \qquad (2.7)
\end{aligned}
$$

The general filtering equation reuses a previously stored filtering results from $t-1$. Evaluating Equation 2.7 for all instantiations of $\vec{X}^t, \vec{\mathcal{A}}^t$ is an algorithm that gives an exact solution to the online filtering problem $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{\mathscr{z}}^{t-1:t}, \vec{\mathscr{b}}^{t-1:t}, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{\mathscr{z}}^{t-1:t}, \vec{\mathscr{b}}^{t-1:t}$ the algorithm has constant time- and space-complexity $\mathcal{O}(1)$.

Iteratively evaluating Equation 2.7 for all instantiations of $\vec{X}^i, \vec{\mathcal{A}}^i$, $\forall i : 0 \leq i \leq t$, is an algorithm that gives an exact solution to the offline filtering problem $\texttt{OffFP}(B_0, B_\rightarrow, \vec{\mathscr{z}}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{\mathscr{z}}^{1:t}, \vec{\mathscr{b}}^{1:t}$ the algorithm has linear time- and space-complexity $\mathcal{O}(t)$.

To obtain a complete distribution, both algorithms require exponentially many evaluations in the dimension of all possible instantiations of random variables per timeslice. Every incremental evaluation is exponential in the number and domain size of unobserved variables from the previous timeslice. ■

This proof shows that filtering has constant complexity over time in dense intra-timeslice ADBNs, i.e., the most general form of cyclic ADBNs. Moreover, it shows that *no novel calculus* (in significant contrast to Milch et al., 2005; Bilmes, 2000; Geiger & Heckerman, 1996) is required to perform exact inference in dense intra-timeslice ADBNs, despite them being based on cyclic graphs. Furthermore, note that at any timestep an effective structure is *not* known and remains unknown, and, still, the general filtering equation does not require an postponed analysis of such a structure.

If one would be constrained to achieve similar results in an acyclic, diagonal (A)DBN, a serialized observation set with spurious intermediate timeslices had to be generated. Filtering in such a generated diagonal network would then be n-times slower. Additionally, a generated order needs to be stored for answering queries. We refrain from a detailed analysis how similar results could be achieved in a diagonal (A)DBN by artificially serializing observations; a diagonal (A)DBN simply represents the wrong model and cyclic ADBNs are immediately required to provide a local and causal parametrization of such models.

Prediction is used to propagate a possible evolution into the future. Essentially, prediction is a filtering query with an empty observation set. If observations are supposed to enforce regularity, plain prediction is not possible in our formalism, as a minimal set of observations is needed to assure regularity. Nevertheless, one could use a *most likely acyclic observation* for prediction. Finding a most likely acyclic observation is a special case of a most likely explanation problem. Later, we introduce in Chapter 5 a solution based on our formalism that fully supports prediction without requiring observations to enforce regularity.

2.3.2 Smoothing Queries and Problems

Queries $P(\vec{x}^{k^\intercal}, \vec{a}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\beta}^{1:t^\intercal})$ with $k < t$ to a probabilistic knowledge base are commonly known as smoothing queries and are used to obtain knowledge in retrospect. Finding answers to queries $P(\vec{x}^{k^\intercal}, \vec{a}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\beta}^{1:t^\intercal})$, $k < t$ defines two smoothing problems.

Similar to the offline filtering problem, the complete smoothing problem needs to be solved when answering multiple smoothing queries for different timeslices from broad timerange of evidences investigated in hindsight.

Definition 2.9 (Complete smoothing problem)**.** *Given a probabilistic knowledge base B_0, $B_\rightarrow$, the complete smoothing problem is the task of determining the conditional probability of random variables at all times $0 \leq j < t$, considering evidence $\vec{z}^{0:t}, \vec{\beta}^{1:t}$ until time t. This is, to obtain*

$$P(\vec{X}^{j^\intercal}, \vec{\mathcal{A}}^{j^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\beta}^{1:t^\intercal}) \ , \ \forall j : 0 \leq j < t \ .$$

We denote a parametrized complete smoothing problem as $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\beta}^{1:t}, t)$. ▲

Solutions to the complete smoothing problem are used to investigate on a distribution's evolution over time in retrospect, but by considering evidence until a later timepoint.

Similar to the online filtering problem, the fixed lag smoothing problem needs to be solved, if smoothing queries need to be answered temporally consecutively for every newly obtained evidence.

Definition 2.10 (Fixed lag smoothing problem)**.** *Given a probabilistic knowledge base B_0, $B_\rightarrow$ and a solution to the respective fixed-lag smoothing problem at $t-1$, the fixed lag smoothing problem at time t is the task of determining the conditional probability distribution of random variables at a time $k = t - \Delta$, considering evidence $\vec{z}^{0:t}, \vec{\beta}^{1:t}$ until time t. This is, to obtain*

$$P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\beta}^{1:t^\intercal}) \ .$$

We denote a parametrized fixed lag smoothing problem as $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{t-\Delta-1:t}, \vec{\beta}^{t-\Delta-1:t}, \Delta, t)$, *which includes a solution to* $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{t-\Delta-2:t-1}, \vec{\beta}^{t-\Delta-2:t-1}, \Delta, t-1)$. ▲

Answers to the fixed lag smoothing problem are used to track a distribution over time. A fixed lag smoothing tracking "lags" Δ timeslices behind real time, but a trajectory is smoothed out by a look ahead in time (cf. Russell & Norvig, 2010, p. 571).

The following example shows up merits of a smoothing query to the running example and accentuates the need for cyclic ADBNs in favor of diagonal DBNs.

Example 2.7 (Explaining away)**.** *Continuing Example 2.6 this example demonstrates that smoothing handles explaining away over multiple timesteps and respects indirect causes. Say, only Don underwent a successful compliance check at time $t = 0$, i.e., $\vec{z}^0 = (\neg d^0)$. For $t = 1$ one found the same document transfer as in Example 2.6, and for $t = 2$, a Sunday, one can neglect all, i.e., $\vec{\beta}^2 = \emptyset$. On that Sunday also irregularities in Earl's documents were found, i.e., $\vec{z}^2 = (+e^2)$.*

If one performs a smoothing query for Claire's initial belief state ($t = 0$) without considering evidence from $t = 2$, an answer is equivalent to the prior belief of $P(C^0|\vec{z}^{0:1^{\mathsf{T}}}, \vec{b}^{1^{\mathsf{T}}}) = \langle 0.5, 0.5\rangle$, as one has not gained any new information with the evidence from $t = 1$. However, with observations from $t = 2$, one needs to consider an indirect influence by Claire onto Earl and the belief in Claire being tainted rises to $P(C^0|\vec{z}^{0:2^{\mathsf{T}}}, \vec{b}^{1:2^{\mathsf{T}}}) \approx \langle 0.532, 0.468\rangle$, because the observation $+e^2$ tells one indirectly something about $+c^0$.

The slow increase from 0.5 to 0.532 is due to the high prior belief in Earl manipulating documents of $P(+e^0) = 0.7$ and it is more likely that Earl has been manipulating documents ever since. If, say, Earl can be relieved from initial incrimination, i.e., $\neg e^0$, the only explanation for this situation is an indirect cause of Claire being tainted, and that Claire has influenced Earl through Don at time $t = 2$, which is correctly handled as $P(+c^1, +m^1_{DE}|\vec{z}^{0:2^{\mathsf{T}}}, \vec{b}^{1:2^{\mathsf{T}}}) = 1$. One can further update an initial prior belief using a smoothing query and find that $P(+d^0) = P(+e^0) = 0$ but $P(+c^0) = 1$. This means that the only explanation for the observations made is that Claire has been corrupt *from the beginning ($+c^0$) and that Don has actually sent a message to Earl ($+m^1_{DE}$), i.e., the possibility of $\neg m^1_{DE}$ is explained away due to the observations made at times $t = 0 \ldots 2$.*

In a diagonal (A)DBN the last example is inexplicable, as indirect influences of t_1 (causally) are first anticipated a step later at t_2 (for $n = 3$). The reason for the inexplicability is confusing because it is not causal: at t_2, the time of incriminating evidence for Earl, one knows that Earl is only influenced by himself, i.e. only t_1-Earl can be the source of his taintedness. At t_1, Earl only receives a document from integrous t_0-Don (observation). This is where the problem lies: t_0-Claire should have influenced t_0-Don by now, but t_0-Claire influences t_1-Don with her message $+m^1_{CD}$. This means that Earl cannot become tainted and the observation $+e^2$ is inexplicable. Mathematically one obtains $P(+e^0|\vec{z}^{0:2^{\mathsf{T}}}, \vec{b}^{1:2^{\mathsf{T}}}) = 0$ in a diagonal DBN, as

$$\begin{aligned}
&P(+e^0|\vec{z}^{0:2^{\mathsf{T}}}, \vec{b}^{1:2^{\mathsf{T}}}) \\
&\quad = \alpha \cdot \sum_{C^0} P(C^0, \neg d^0, \neg e^0) \cdot \sum_{C^1} P(C^1|\ldots, C^0) \cdot \sum_{D^1} P(D^1|\ldots, C^0, \ldots) \\
&\qquad\qquad \cdot \sum_{M^1_{DE}} P(+m^1_{CD}, \neg m^1_{DC}, \neg m^1_{ED}, \neg m^1_{CE}, \neg m^1_{EC}, M^1_{DE}) \\
&\quad \cdot \Big(P(\underline{+e^1}|\underline{\neg m^1_{CE}}, M^1_{DE}, C^0, \underline{\neg d^0}, \underline{\neg e^0}) \cdot P(\neg e^2|\neg m^2_{CE}, \neg m^2_{DE}, C^1, D^1, +e^1) \\
&\quad + P(\neg e^1|\neg m^1_{CE}, M^1_{DE}, C^0, \neg d^0, \neg e^0) \cdot P(\underline{\neg e^2}|\underline{\neg m^2_{CE}}, \underline{\neg m^2_{DE}}, C^1, D^1, \underline{\neg e^1})\Big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \cdot P(\neg m^2_{**}) = 0
\end{aligned}$$

where both alternatives of E^1 are impossible under the given observations, and CPD entries $P(+e^2|\underline{\neg m^2_{*E}}, C^1, D^1, \underline{\neg e^1})$ and $P(+e^1|M^1_{DE}, \underline{\neg m^1_{CE}}, C^0, \underline{\neg d^0}, \underline{\neg e^0})$ are uniquely identified to be 0 by the underlined dependencies. By definition, one obtains $P(\neg e^2|\ldots, +e^2, \ldots) = 0$, and, thus, obtained results from a diagonal DBN stand in conflict with the probability axioms of Kolmogorov (compare Proposition 2.2 under which $\vec{b}^{1:2}$ is not indirect-free). As Example 2.7 shows, the observation of Example 2.7 is regular and an intra-timeslice ADBN fully respects indirect influences while remaining a first-class representation.

The following theorems state the complexity of solving smoothing problems and the related proof derives a commonly known *forward-backward algorithm* for finding exact solutions to smoothing problems.

Theorem 2.4 (Exact solution to the complete smoothing problem)**.** *Given a complete smoothing problem* $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$*, finding an exact solution is linear or quadratic in* t*. Finding an exact solution is exponential in the maximal dimension of unobserved variables* $\vec{\zeta}^*$*,* $\vec{\beta}^*$ *in a timestep* $0 < * \leq t$*, and in the largest domain* $\mathrm{dom}(\zeta_+)$*,* $\mathrm{dom}(\beta_+)$ *of all random variables* $\zeta_+ \in \vec{\zeta}^{0:t}$*,* $\beta_+ \in \vec{\beta}^{1:t}$*. Finding an exact solution is exponential in the dimension of number of random variables* $|\vec{X}^t|$*,* $|\vec{\mathcal{A}}^t|$ *and a respective maximal domain size* $\mathrm{dom}(X_+)$*,* $\mathrm{dom}(A_{++})$ *of all random variables* $X_+ \in \vec{X}^t$*,* $A_{++} \in \vec{\mathcal{A}}^t$*.* ▲

Theorem 2.4 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$ in time-complexity $\mathcal{O}(t^2 \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\mathrm{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\mathrm{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with $\mathcal{O}(t \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ space-complexity for storing all smoothing distributions. The proof is combined with the proof for the following theorem.

Theorem 2.5 (Exact solution to the fixed lag smoothing problem)**.** *Given a fixed lag smoothing problem* $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{t-\Delta-1:t}, \vec{\mathscr{b}}^{t-\Delta-1:t}, \Delta, t)$*, finding an exact solution is constant in* t*. Let* $k = t - \Delta$ *for brevity. Finding an exact solution is exponential in the maximal dimension of unobserved variables* $\vec{\zeta}^*$*,* $\vec{\beta}^*$ *in a timestep* $k - 1 < * \leq t$*, and in the largest domain* $\mathrm{dom}(\zeta_+)$*,* $\mathrm{dom}(\beta_+)$ *of all random variables* $\zeta_+ \in \vec{\zeta}^{k-1:t}$*,* $\beta_+ \in \vec{\beta}^{k-1:t}$*. Finding an exact solution is exponential in the dimension of number of random variables* $|\vec{X}^t|$*,* $|\vec{\mathcal{A}}^t|$ *and a respective maximal domain size* $\mathrm{dom}(X_+)$*,* $\mathrm{dom}(A_{++})$ *of all random variables* $X_+ \in \vec{X}^t$*,* $A_{++} \in \vec{\mathcal{A}}^t$*.* ▲

Theorem 2.5 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{k-1:t}, \vec{\mathscr{b}}^{k-1:t}, \Delta, t)$ with time-complexity $\mathcal{O}(\Delta \cdot |\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\mathrm{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\mathrm{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and $\mathcal{O}(|\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|})$ space-complexity for storing a smoothing distribution of timeslice k.

Theorems 2.4 and 2.5 discuss the computational fixed-parameter complexity of commonly known smoothing problems in dense intra-timeslice ADBNs. Most importantly Theorem 2.4 states that solving the complete smoothing problem has linear complexity over the number of observed timeslices, as one expects in classical DBNs, and Theorem 2.5 states that solving the fixed lag smoothing problem has constant complexity over the number of observed timeslices, as one expects in classical DBNs as well. Both theorems are proven below, where a major result is that the classically known forward-backward-algorithm still remains applicable to cyclic ADBNs.

Proof of Theorems 2.5 and 2.4 (Solutions to smoothing problems). The general smoothing equation is obtained by straight marginalization from the joint probability as

$$P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) = \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) \; .$$

Using Equation 2.5 one obtains

$$P(\vec{X}^{k\top},\vec{\mathcal{A}}^{k\top}|\vec{z}^{0:t\top},\vec{b}^{1:t\top}) = \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:t-1\top},\vec{\mathcal{A}}^{1:t-1\top}) \\ \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t|\vec{X}^{t\top} \setminus X_i^t, \vec{A}_i^{t\top}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \quad . \tag{2.8}$$

We define an intermediate joint probability as

$$P(\vec{X}^{k:t\top},\vec{A}^{k:t\top}) = \prod_{X_i^k \in \vec{X}^k} P(X_i^k|\vec{X}^{k\top} \setminus X_i^k, \vec{A}_i^{k\top}, X_i^{k-1}) \\ \cdot \prod_{A_{ij}^k \in \vec{\mathcal{A}}^k} P(A_{ij}^k) \cdot P(\vec{X}^{k+1:t\top},\vec{\mathcal{A}}^{k+1:t\top}) \ ,$$

which is a term $T^{k-1:t}$ depending on variables from time $k-1$ until t. One can therefore split the recursive definition of the joint probability and obtain

$$\begin{aligned} & P(\vec{X}^{k\top},\vec{\mathcal{A}}^{k\top}|\vec{z}^{0:t\top},\vec{b}^{1:t\top}) \\ &= \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:k\top},\vec{\mathcal{A}}^{1:k\top}) \cdot P(\vec{X}^{k+1:t\top},\vec{\mathcal{A}}^{k+1:t\top}) \\ &= \alpha \cdot \sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} P(\vec{X}^{0:k\top},\vec{\mathcal{A}}^{1:k\top}) \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t\top},\vec{\mathcal{A}}^{k+1:t\top}) \\ &= \alpha \cdot \left(\sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} P(\vec{X}^{0:k\top},\vec{\mathcal{A}}^{1:k\top}) \right) \cdot \left(\sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t\top},\vec{\mathcal{A}}^{k+1:t\top}) \right) \ . \end{aligned} \tag{2.9}$$

One finds a previous (stored) filtering problem in the first sum-product, known as a "forward message," and a new latter sum-product commonly known in smoothing equations as a "backward-message." Using an adequate recursive definition for the latter term, one obtains the so-called forward-backward algorithm. The commonly known "sensor model" is, due to in-time-slice dependencies, included in the forward, as well as backward message.

The latter sum-product term from $k+1$ to t corresponds to $P(\vec{z}^{k+1:t\top},\vec{b}^{k+1:t\top}|\vec{X}^{k\top},\vec{\mathcal{A}}^{k\top})$ from a probabilistically point of view, but essentially is a just a term $T^{k:t}$ over variables from time k to t. The splitting into two disjoint products is possible, because all variables from time k are query variables, i.e., are constant in a query. Without determining a *full* joint probability, a backward message is not derivable in multiply connected BNs. In fact, the probabilistic view also derives from Bayes' theorem and a first-order Markov assumption, but we rely on straight calculus here, as intra-timeslice dependencies pose various pitfalls here.

Therefore, one obtains

$$
\begin{aligned}
&P(\vec{z}^{k+1:t^\intercal}, \vec{b}^{k+1:t^\intercal} | \vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t^\intercal}, \vec{\mathcal{A}}^{k+1:t^\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1^\intercal} \backslash X_i^{k+1}, \vec{A}_i^{k+1^\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \\
&\qquad\qquad \cdot P(\vec{X}^{k+2:t^\intercal}, \vec{\mathcal{A}}^{k+2:t^\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1^\intercal} \backslash X_i^{k+1}, \vec{A}_i^{k+1^\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \\
&\qquad\qquad \cdot \sum_{\vec{\zeta}^{k+2:t}} \sum_{\vec{\beta}^{k+2:t}} P(\vec{X}^{k+2:t^\intercal}, \vec{\mathcal{A}}^{k+2:t^\intercal}) \ , \qquad (2.10)
\end{aligned}
$$

in which one finds the backward message as the latter term. Using a recursive definition, one obtains the backward message as

$$
\begin{aligned}
&P(\vec{z}^{k+1:t^\intercal}, \vec{b}^{k+1:t^\intercal} | \vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal}) \\
&\quad = \alpha \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1^\intercal} \backslash X_i^{k+1}, \vec{A}_i^{k+1^\intercal}, X_i^k) \\
&\qquad \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \cdot P(\vec{z}^{k+2:t^\intercal}, \vec{b}^{k+2:t^\intercal} | \vec{X}^{k+1^\intercal}, \vec{\mathcal{A}}^{k+1^\intercal}) \quad . \qquad (2.11)
\end{aligned}
$$

Finally, by combining Equation 2.9 and 2.11, one obtains the general smoothing equation as

$$
P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) = P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:k^\intercal}, \vec{b}^{1:k^\intercal}) \cdot P(\vec{z}^{k+1:t^\intercal}, \vec{b}^{k+1:t^\intercal} | \vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal}) \, . \qquad (2.12)
$$

With the recursive definition of the backward message, Eq. 2.11, this yields

$$
\begin{aligned}
&P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) = \alpha \cdot P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:k^\intercal}, \vec{b}^{1:k^\intercal}) \\
&\quad \cdot \sum_{\vec{\zeta}^{k+1}} \sum_{\vec{\beta}^{k+1}} \prod_{X_i^{k+1} \in \vec{X}^{k+1}} P(X_i^{k+1} | \vec{X}^{k+1^\intercal} \backslash X_i^{k+1}, \vec{A}_i^{k+1^\intercal}, X_i^k) \cdot \prod_{A_{ij}^{k+1} \in \vec{\mathcal{A}}^{k+1}} P(A_{ij}^{k+1}) \\
&\qquad\qquad \cdot P(\vec{z}^{k+2:t^\intercal}, \vec{b}^{k+2:t^\intercal} | \vec{X}^{k+1^\intercal}, \vec{\mathcal{A}}^{k+1^\intercal}) \ . \qquad (2.13)
\end{aligned}
$$

Evaluating the general smoothing Equation 2.12 for all instantiations of $\vec{X}^k, \vec{\mathcal{A}}^k$ decrementally for descending $k = t-1 \dots 0$ is an algorithm that gives an exact solution to the complete smoothing problem `ComplSP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$. During the decremental evaluation the backward message is stored: an intermediate result $P(\vec{z}^{k+1:t^\intercal}, \vec{b}^{k+1:t^\intercal} | \vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal})$ (the backward message) obtained during an evaluation of $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$ at time k is needed in an upcoming (but temporally backward) evaluation of $P(\vec{X}^{k-1^\intercal}, \vec{\mathcal{A}}^{k-1^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$ at time $k-1$. Thus, obtaining the last term of Equation 2.13 is constant in t for every evaluation. The first term $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:k^\intercal}, \vec{b}^{1:k^\intercal})$ of the general smoothing equation poses a filtering problem, for which a solution is found in $\mathcal{O}(1)$ in a storage from a solution to an offline filtering problem `OffFP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$, which is found in $\mathcal{O}(t)$ by the previously derived algorithm in Proof of Theorems 2.2 and 2.3. If memory is scarce, $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:k^\intercal}, \vec{\mathcal{b}}^{1:k^\intercal})$ is obtained in $\mathcal{O}(k)$ for every k by solving all $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{z}^{i-1:i}, \vec{\mathcal{b}}^{i-1:i}, i), 0 \leq i \leq k$. Thus, for a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{1:t}, \vec{\mathcal{b}}^{1:t}$ the algorithm has linear time-complexity $\mathcal{O}(t)$ and linear space-complexity $\mathcal{O}(2 \cdot t)$ or quadratic time-complexity $\mathcal{O}(t^2)$, but halved space-complexity $\mathcal{O}(t)$. For every evaluation at time k, the backward message "moves one down" from $k+1$ to k and the forward message "ripples up" from 0 to k (or is obtained from a storage). For special DPGMs, a forward message can also be "moved one down" by using an inverse matrix calculation. With this picture in mind, the forward-backward-algorithm is sometimes seen as a "dance" between messages.

Evaluating the general smoothing Equation 2.12 for all instantiations of $\vec{X}^k, \vec{\mathcal{A}}^k$ for $k = t - \Delta$ is an algorithm that gives an exact solution to the fixed lag smoothing problem $\texttt{FLagSP}(B_0, B_\rightarrow, \vec{z}^{t-\Delta-1:t}, \vec{\mathcal{b}}^{t-\Delta-1:t}, \Delta, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{t-\Delta-1:t}, \vec{\mathcal{b}}^{t-\Delta-1:t}$, the algorithm obtains the first term as a solution to an online filtering problem in $\mathcal{O}(1)$ and the latter term in $\mathcal{O}(\Delta)$ using the recursive definition.

To obtain a complete distribution, both algorithms require exponentially many evaluations in the dimension of all possible instantiations of random variables per timeslice. Every incremental evaluation is exponential in the number and domain size of unobserved variables from the previous and consecutive timeslice. ■

This proof shows that smoothing problems are solvable in a classical calculus without any introduction of external frameworks or novel mathematical operators. Moreover, it shows that the commonly known forward-backward-algorithm still remains applicable to dense intra-timeslice ADBNs without any modification, despite them being based on cyclic graphs.

2.3.3 Most Likely Explanation Problem

It is sometimes desirable to obtain knowledge in the form of discrete explanations, instead of in the form of probability distributions. A common query for this case is a query for a most likely explanation for a given observation, i.e., to find the sequence of instantiations that most likely took place given the observed evidence.

Definition 2.11 (Most likely explanation query and problem). *Given a probabilistic knowledge base B_0, $B_\rightarrow$, and obtained evidence $\vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}$, it is the task to find the instantiation $mle^t = \left(\vec{x}^{0:t}, \vec{a}^{1:t}\right)$ conforming with evidence $\vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}$ that maximizes the complete joint probability distribution to $P(mle^t)$. This is to determine,*

$$mle^t = \left(\operatorname*{arg\,max}_{\vec{\zeta}^{0:t}, \vec{\beta}^{1:t}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathcal{b}}^{1:t^\intercal}) \right) \cup \left(\vec{z}^{0:t^\intercal}, \vec{\mathcal{b}}^{1:t^\intercal} \right) \ ,$$

and

$$P(mle^t) = \max_{\vec{\zeta}^{0:t}, \vec{\beta}^{1:t}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathcal{b}}^{1:t^\intercal}) \ .$$

We denote a most likely explanation problem as $\texttt{MleP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathcal{b}}^{1:t}, t)$. ▲

As a continuation of Example 2.7, the following example answers a most likely explanation query.

Example 2.8. *From Example 2.7 it is already evident that only one possible explanation remains. Under observations $\vec{z}^{0:2}$, $\vec{\mathscr{b}}^{1:2}$ the instantiations of interest are $C^0, C^1, D^1, E^1, M^1_{DE}, C^2, D^2$. As discussed in Example 2.7, the only explanation is that Claire is tainted from the beginning and Claire has indirectly tainted Earl through Don during timeslice* 1. *Therefore, one obtains*

$$mle^2 = ({+c^0}, \neg d^0, \neg e^0, {+c^1}, {+d^1}, {+e^1}, {+c^2}, {+d^2}, {+e^2}, {+m^1_{CD}}, {+m^1_{DE}}, \neg m^1_{**}, \neg m^2_{**})$$
$$P(mle^2) = 1 \ .$$

As the example demonstrates, finding answers to most likely explanation queries are especially useful if one is interested in a concrete sequence of actions. While a concrete instantiation is easier to interpret than a complete conditional probability distribution of a timeslice, only considering *one* chain of events can easily lead to false assumptions or accusations. Therefore, when considering answers from most likely explanation queries, it is highly important to consider $P(mle^t)$ as well. As one obtains $P(mle^2) = 1$ in this example, it is certain to say that “corrupt Claire has directly and indirectly tainted all other employees.”

The following theorem discusses the complexity of obtaining such a most likely explanation.

Theorem 2.6 (Exact solution to the most likely explanation problem)**.** *An algorithm providing an exact solution to* `MleP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$ *has the same time-complexity as an algorithm solving an offline filtering problem* `OffFP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$*. An algorithm has $\mathcal{O}(|\operatorname{dom}(X_+)|^{|\vec{X}^t|} \cdot |\operatorname{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot (|\vec{X}^t| + |\vec{\mathcal{A}}^t|) \cdot t)$ space-complexity for (intermediately) storing a distribution over possible most likely explanations.* ▲

Both, the most likely explanation problem and the filtering problem, as well as their respective solutions are very similar. Likewise, also an on- and offline most likely explanation problem could be formulated. For brevity, and as we expect offline most likely explanation queries to be more frequently, we do not discuss an online most likely explanation problem. We prove Theorem 2.6 by deriving an exact solution to the general most likely explanation equation from the joint probability based on a maximization problem. The following proof is similar to Proof of Theorems 2.2 and 2.3 and, likewise, shows that neither a novel calculus nor any novel operators are required to solve most likely explanation query answering problems.

Proof of Theorem 2.6 (Solution to the most likely explanation problem). Let $mle^t_{\vec{x}^t,\vec{a}^t}$ be *some* most likely explanation ending on $\vec{x}^t, \vec{a}^t$. Then, to find $mle^t_{\vec{x}^t,\vec{a}^t}$ the following problems need to be solved

$$mle^t_{\vec{x}^t,\vec{a}^t} = \left(\operatorname*{arg\,max}_{\vec{\zeta}^{0:t-1}, \vec{\beta}^{1:t-1}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) \right) \cup \left(\vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal} \right) \ ,$$

and

$$P(mle^t_{\vec{x}^t,\vec{a}^t}) = \max_{\vec{\zeta}^{0:t-1}, \vec{\beta}^{1:t-1}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) \ ,$$

where all variables of timeslice t are fixed by the query, and all unobserved variables of timeslice 0 to $t-1$ are subject to maximization. Remaining variables from timeslices 0 to t are then fixed by evidence. Let MLE^t be the set of all most likely explanations $mle^t_{\vec{x}^t,\vec{a}^t}$ with all possible endstates $\vec{x}^t, \vec{a}^t$.

For this proof we derive an exact solution to the problem of finding $P(mle^t_{\vec{x}^t,\vec{a}^t})$ and, later, show how it provides an exact solution to the most likely explanation problem. Using the recursive definition of the joint probability distribution for a dense inter-timeslice ADBN from Proposition 2.4 one obtains

$$\begin{aligned} P(mle^t_{\vec{x}^t,\vec{a}^t}) = \max_{\vec{\zeta}^{0:t-1},\vec{\beta}^{1:t-1}} & P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal} | \vec{z}^{0:t-1^\intercal}, \vec{b}^{1:t-1^\intercal}) \\ & \cdot \prod_{X^t_k \in \vec{X}^t} P(X^t_k | \vec{X}^{t^\intercal} \backslash X^t_k, \vec{A}^{t^\intercal}_k, X^{t-1}_k) \cdot \prod_{A^t_{cv} \in \vec{\mathcal{A}}^t} P(A^t_{cv}) \end{aligned}$$

Splitting the maximization operator yields

$$\begin{aligned} P(mle^t_{\vec{x}^t,\vec{a}^t}) = \max_{\vec{\zeta}^{t-1},\vec{\beta}^{t-1}} \max_{\vec{\zeta}^{0:t-2},\vec{\beta}^{1:t-2}} & P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal} | \vec{z}^{0:t-1^\intercal}, \vec{b}^{1:t-1^\intercal}) \\ & \cdot \prod_{X^t_k \in \vec{X}^t} P(X^t_k | \vec{X}^{t^\intercal} \backslash X^t_k, \vec{A}^{t^\intercal}_k, X^{t-1}_k) \cdot \prod_{A^t_{cv} \in \vec{\mathcal{A}}^t} P(A^t_{cv}) \ , \end{aligned}$$

which can be combined with

$$P(mle^t_{\vec{x}^t,\vec{a}^t}) = \max_{\vec{\zeta}^{0:t-1},\vec{\beta}^{1:t-1}} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal}) \ ,$$

and one obtains the general most likely explanation equation as

$$\begin{aligned} P(mle^t_{\vec{x}^t,\vec{a}^t}) = \max_{\vec{\zeta}^{t-1},\vec{\beta}^{t-1}} & P(mle^{t-1}_{\vec{x}^{t-1},\vec{a}^{t-1}}) \\ & \cdot \prod_{X^t_k \in \vec{X}^t} P(X^t_k | \vec{X}^{t^\intercal} \backslash X^t_k, \vec{A}^{t^\intercal}_k, X^{t-1}_k) \cdot \prod_{A^t_{cv} \in \vec{\mathcal{A}}^t} P(A^t_{cv}) \ . \end{aligned} \qquad (2.14)$$

The general most likely explanation equation (Equation 2.14) is equivalent to the general filtering equation (Eq. 2.7) with exchanged $\sum$ and $\arg\max$ operators and has the same time-complexity. To keep track of the maximization argument $mle^t_{\vec{x}^t,\vec{a}^t}$, additional storage is required, which grows linearly in the number of random variables and time t. Storage requirements are exponential in the dimension of all possible instantiations of all random variables to store all $mle^t_{\vec{x}^t,\vec{a}^t}$. The final most likely explanation mle^t is obtained from MLE^t by

$$mle^t = \mathop{\arg\max}_{mle^t_{\vec{x}^t,\vec{a}^t} \in MLE^t} P(mle^t_{\vec{x}^t,\vec{a}^t}) \ . \qquad (2.15)$$

Evaluating Equation 2.14 for all instantiations of random variables $\vec{X}^t, \vec{\mathcal{A}}^t$ and consecutively evaluating Equation 2.15 is an algorithm that gives an exact solution to `MleP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{b}^{1:t}, t)$. For a fixed $B_0, B_\rightarrow$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{1:t}, \vec{b}^{1:t}$, the algorithm has linear time- and space-complexity $\mathcal{O}(t)$. ■

2.3.4 Experimental Evaluation of Exact Inference

Section 2.3 shows that no novel calculus must be invented to perform inference, i.e., to solve commonly known query answering problems, in dense intra-timeslice ADBNs, despite them being based on cyclic graphs, and, moreover, shows that no computational overhead is introduced to perform inference. We provide substantiating empirical evidence for the latter results by showing that solving multiple on- and offline filtering- and smoothing-problems remains tractable even over large periods of time[2]. To do so, we perform multiple experiments, where in every experiment, state variables in $\vec{X}^t, t > 0$ are assigned a randomly generated individual CPD following Definition 2.2. Further, in every experiment random priors are assigned to random variables $\vec{X}^0, \vec{\mathcal{A}}^t$. Every randomly generated probability is taken from the range $[0.1, 0.9]$ to avoid impossible observations. For every timestep $t > 0$ we generate random observations $\vec{b}^t$ conforming with Theorem 2.1. We refrain from observing state variables $\vec{X}^t$, i.e., $\vec{z}^t = \emptyset$, to achieve worst-case time complexity.

In each experiment, state variables in $\vec{X}^t, t > 0$ are assigned a randomly generated individual CPD following Definition 2.2. Further, in each experiment random priors are assigned to random variables $\vec{X}^0, \vec{\mathcal{A}}^t$. Every randomly generated probability is taken from the range $[0.1, 0.9]$ to avoid impossible observations. For every timestep $t > 0$ we generate random observations $\vec{b}^t$ conforming with Theorem 2.1. We refrain from observing state variables $\vec{X}^t$, i.e., $\vec{z}^t = \emptyset$, to achieve worst-case time complexity.

In every experiment, we consecutively solve the online filtering problem $\texttt{OnlFP}(B_0, B_\rightarrow, \vec{z}^{i-1:i}, \vec{b}^{i-1:i}, i)$ at every timestep i using the algorithm derived in Proof of Theorems 2.2 and 2.3, and denote the computation time per timeslice i in Figure 2.3. Further, we solve the complete smoothing problem $\texttt{ComplSP}(B_0, B_\rightarrow, \vec{z}^{0:i}, \vec{b}^{1:i}, i)$ at every timestep i, using the algorithm derived in Proof of Theorems 2.5 and 2.4 using stored filtering results (Figure 2.5) and recalculated filtering results (Figure 2.4) and denote the computation time per timeslice i.

As $\vec{z}^t = \emptyset$, one does not acquire any new knowledge about state variables using smoothing, i.e., $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:k^\intercal}, \vec{b}^{1:k^\intercal})$ is equal to $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{b}^{1:t^\intercal})$. A Kullback-Leibler-Divergence of both solutions was measured to be zero (in the range of double precision), which verifies our implementation in that sense.

Experiments were repeated 139 times for $n = 4$, i.e., an ADBN consisting of 16 random variables $\vec{X}^t, \vec{\mathcal{A}}^t$ per timeslice for a timerange of 40. All experimental results validate expected fixed-parameter tractability of filtering and smoothing problems in ADBNs. Nevertheless, experiments show that for models beyond $n = 5$, i.e., beyond 25 random variables, approximate inference techniques are needed, for which the following section provides an introduction and a demonstration.

[2]All experiments are reproducible, for which we supply an experimental framework implementing inference in dense ADBNs in C available at: http://adbn.motzek.org

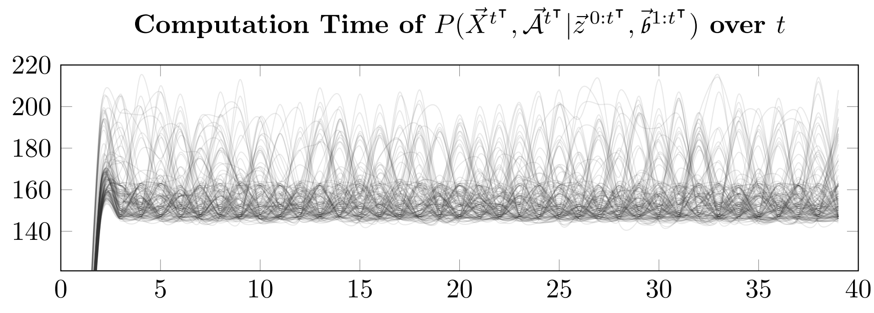

Figure 2.3. *Finding an exact solution to the online filtering problem* `OnlFP`$(B_0, B_\rightarrow, \vec{z}^{i-1:i}, \vec{b}^{i-1:i}, i)$ *at timestep i (abscissa) is constant (computation time in ms, ordinate, Pearson's product-moment correlation coefficient between t* > 1 *and computation time r* ≈ -0.015*) at every timestep i. (139 evaluations superimposed)*

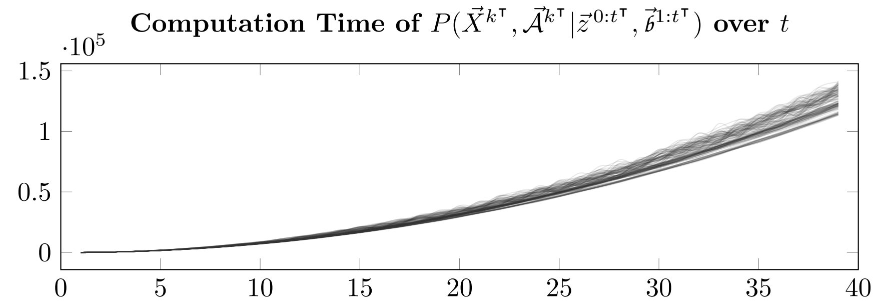

Figure 2.4. *Finding an exact solution to the complete smoothing problem* `ComplSP`$(B_0, B_\rightarrow, \vec{z}^{0:i}, \vec{b}^{1:i}, i)$ *at timestep i (abscissa), without stored filtering results, scales quadratically (computation time in ms, ordinate,* $R^2 = 0.9913$ *for quadratic regression) with increasing timesteps, but has constant memory requirements.*

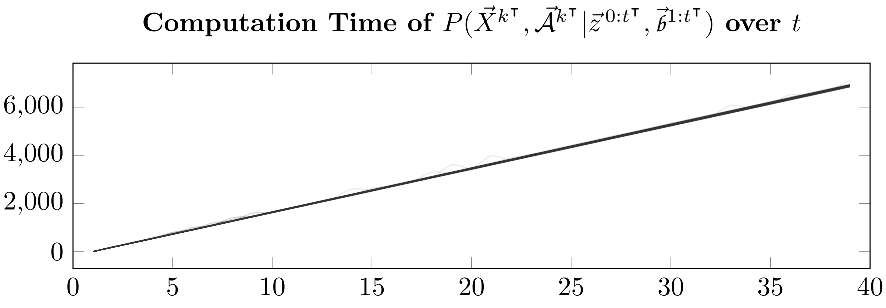

Figure 2.5. *Finding an exact solution to the complete smoothing problem* `ComplSP`$(B_0, B_\rightarrow, \vec{z}^{0:i}, \vec{b}^{1:i}, i)$ *at timestep i (abscissa), using stored filtering results scales linearly (computation time in ms, ordinate,* $r = 0.9999$*) with increasing timesteps, but has linear memory requirements over time.*

2.4 Approximate Inference Techniques in ADBNs

Finding exact solutions to common problems is only tractable in small toy domains, i.e., with very few random variables. Approximate inference techniques have shown to be a valuable alternative for Bayesian networks and can be divided into two categories: (i) Approximations to the derivation of exact calculations, such as loopy belief propagation or variational methods, and (ii) stochastic sampling methods on which we focus in this section. Introductions to sampling methods for general graphical probabilistic models from a stochastical perspective are provided by Arulampalam, Maskell, Gordon, and Clapp (2002), Murphy (2012, pp. 823–831) and Doucet and Johansen (2009).

Naive adaptations of approximate inference techniques for BNs towards DBNs, however, show to be suboptimal, as an approximation error accumulates over inference time, i.e., only for few consecutive time-slices accurate results are obtained. Fortunately, modifications to these approaches allow for approximate inference in DBNs even for long periods of inference time under a bounded and constant approximation error. Such a property, i.e., approximation algorithm providing a bounded and constant error over time must be preserved for any novel DPGM such as ADBNs. Therefore, we show and prove in this section that commonly known "particle filters" aka sequential important resampling (SIR) techniques remain applicable in ADBNs without any overhead, and provide approximations with constant error over time. On top of that, an interesting challenge remains: approximate inference techniques for DBNs are based on the topological ordering of $B_{\rightarrow}$, but a topological ordering of a (dense intra-timeslice) ADBN is firstly known in an instantiation. Therefore, an approximate inference technique must rapidly adapt to specific contexts at every sampling step.

Remark 2.3 (Approximate inference nomenclature)**.** *Sampling based approximate inference techniques for DBNs are referred to under various names (cf. Russell & Norvig, 2010, pp. 603ff). In general, every sampling based technique can be classified as a* Monte Carlo *simulation. For DBNs, two major sampling based approximations are known as* Sequential Importance Sampling *(SIS) and* Sequential Importance Resampling *(SIR), both are sometimes referred to as* Sequential Monte Carlo *(SMC) techniques (cf. Arulampalam et al., 2002). Furthermore, SIS and SIR are often referred to as* particle filters *(especially by Murphy, 2012, pp. 823–831 and Doucet, de Freitas, Murphy, & Russell, 2000). We adapt the naming of SIS and SIR.*

The key idea behind sampling approaches is to estimate a probability distribution, e.g., $P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal})$, by a large number of samples, instead of performing exact inference. This means that a modeled stationary process over time is simulated multiple times, where every simulation generates a sample S with a specific outcome.

Definition 2.12 (Sample)**.** *Let $S = (\vec{x}^t, \vec{a}^t, w)$ denote a sample of random variables $\vec{X}^t, \vec{\mathcal{A}}^t$ at time t, i.e. an arbitary instantiation $\vec{x}^t, \vec{a}^t$ of $\vec{X}^t, \vec{\mathcal{A}}^t$. Each sample is assigned a weight w. A sample S is sometimes called a particle.*

Let $\vec{S}^t$ denote the set of all samples obtained at time t. Let $n_S = |\vec{S}^t|$ be the number of samples per timeslice. Let w_S denote the assigned weight w of a sample S. Let $S^t_{\vec{x},\vec{a}}$ denote a sample with instantiation $\vec{x}^t, \vec{a}^t$ and let $\vec{S}^t_{\vec{x},\vec{a}}$ denote the set of all samples $S_{\vec{x}^t,\vec{a}^t}$ obtained at time t. We use $S^t_{\vec{x},\vec{a}}$ in favor of a notation for continuous state spaces using the delta Dirac mass distribution located at $\vec{x}^t, \vec{a}^t$ (cf. Murphy, 2012, pp. 823–831 and Doucet & Johansen, 2009). ▲

The key challenge behind approximate inference techniques is to efficiently and correctly generate these samples for a given (dynamic) probabilistic graphical model and a set of observations. In the following section we discuss two approaches, namely, *Sequential Importance Sampling* (SIS) and *Sequential Importance Resampling* (SIR).

2.4.1 SIS and SIR in ADBNs

In order to approximate solutions to filtering and smoothing problems, a distribution of obtained samples must correspond to the distribution they shall approximate. For classic Bayesian networks multiple techniques exist to generate stochastically correct samples, namely "prior sampling," "rejection sampling," "likelihood weighting," and "Gibbs sampling" (cf., Russell & Norvig, 2010, pp. 530–535). SIS and SIR represent an adaption of likelihood weighting to DBNs. In SIS and SIR samples are sequentially updated at each timestep and are weighted according to their importance, i.e., their conformity with evidence.

Informally, we sketch SIS for a generic DBN $(B_0, B_\to)$ with state variables $\vec{X}^t$ as follows: To update a sample S^{t-1}, i.e., an arbitrary instantiation of random variables at $t-1$, to an updated sample S^t, one follows the stochastic model of the DBN: Following the topological ordering of $B_\to$ every random variable X_i^t is (a) observed or (b) unobserved. In case (a) the instantiation of X_i^t in the updated sample S^t is fixed to the observed value $x_i^t \in \vec{z}^t$ and weighted according to conformity of the current evolution $S^{t-1} \to S^t$ with the observed evidence $P(x_i^t|S^{t-1}, S^t)$. In case (b) the instantiation of X_i^t is sampled, i.e., is randomly instantiated, corresponding to its probability distribution conditioned on the current evolution, i.e., sampled according to $P(X_i^t|S^{t-1}, S^t)$. S^{t-1} is a full instantiation of all random variables at $t-1$, and S^t is generated sequentially according to a topological ordering. Therefore, S^t is initially empty and grows with every sampled X_i^t until it contains a sample of every random variable of timeslice t, i.e., until the topological ordering has been traversed. Note that some random variables in S^{t-1}, S^t may not exert a direct influence on X_i^t and are, therefore, non-descendants of X_i^t. Further, all parents of X_i^t are given by S^{t-1}, S^t which renders X_i^t conditionally independent of all non-descendants. This is a highly beneficial circumstance, as, then, $P(X_i^t|S^{t-1}, S^t)$ is directly given by a locally defined CPD entry and is obtained without any computational cost.

Adapting SIS to a dense intra-timeslice ADBN with variables $\vec{X}^{0:t}$ and $A^{1:t}$, gives an update procedure as Algorithm 1 that is used in sequential importance (re)sampling techniques as shown in Algorithm 2. This update procedure requires an effective topological ordering in a timestep t based on the observed activator random variables in $\vec{b}^t$.

Notation 2.6 (Sampling operator $\propto$). *$X \propto P(X)$ represents a sampling operation. The sampling operation instantiates a random variable X randomly to one of its possible values $x \in \mathrm{dom}(X)$ according to a given random distribution P of X for all $x \in \mathrm{dom}(X)$.*

The analysis of an effective topology is necessary to sample from the correct distribution and to correctly weight a sample. Classically, one constant topological ordering is given by $B_0, B_\to$ for each timeslice t. However, in cyclic ADBNs multiple structures, i.e., topological orderings, are represented by $B_\to$, to which a cyclic ADBN intrinsically adapts in each timeslice. Therefore, it must

Algorithm 1 Evolving a sample

```
procedure EVOLVE(from $S^{t-1}$ to $S^t$, given $\vec{z}^{\,t}$, $\vec{\mathscr{b}}^t$)
    for all $A^t_{ij} \in \vec{\mathcal{A}}^t$ do                                  ▷ skip at t=0
        if $a^t_{ij} \in \vec{\mathscr{b}}^t$ then                                      ▷ is observed
            $w \leftarrow w \cdot P(a^t_{ij})$                              ▷ initalized by Alg. 2
            set $A^t_{ij} = a_{ij}$ in $S^t$
        else
            set $A^t_{ij} \propto P(A^t_{ij})$ in $S^t$
    for all $X^t_i \in \vec{X}^t$ following topological ordering induced by $\vec{\mathscr{b}}^t$ do
        if $x^t_i$ is observed in $\vec{z}^{\,t}$ then
            $w \leftarrow w \cdot P(x^t_i | \vec{x}^{\,t}, \vec{a}^{\,t}_i, x^{t-1}_i)$
            set $X^t_i = x_i$ in $S^t$
        else
            set $X^t_i \propto P(X^t_i | \vec{x}^{\,t}, \vec{a}^{\,t}_i, x^{t-1}_i)$ in $S^t$
```

be analyzed at every timeslice. This circumstance does not come as a surprise and is later discussed in Section 2.5. In consequence, Algorithm 1 uses local CPDs conditioned on instantiations of $\vec{X}^t$ that are not yet updated, i.e., are undefined yet in S^t. Still, the procedure remains sound, as all uninstantiated or undefined instantiations of dependencies are deactive due to the induced topological ordering of $\vec{\mathscr{b}}^t$.

Algorithm 2 Sequential importance (re)sampling (SIS/SIR)

```
begin
    init $\vec{S}$ with $n_S$ $S = (\emptyset, \emptyset, 1)$ samples
    set $t = 0$
    loop
        given $\vec{z}^{\,t}$, $\vec{\mathscr{b}}^t$                                          ▷ t=0: $\vec{\mathscr{b}}^t = \emptyset$
        for all samples $S \in \vec{S}$ do
            $S \leftarrow$EVOLVE$(S, \vec{z}^{\,t}, \vec{\mathscr{b}}^t)$
        for all samples $S \in \vec{S}$ do
            $w_S \leftarrow \frac{w_S}{\sum_{S' \in \vec{S}} w_{S'}}$
        $\vec{S} \propto P(\vec{S})$                                        ▷ Only for SIR.
        next $t$
    return $\vec{S}$
end
```

Sequential importance sampling provides approximate solutions to filtering and smoothing problems, as the following theorems state.

Theorem 2.7 (Exact sampling based solutions to filtering problems)**.** *For $n_S \to \infty$ and infinite numerical precision, samples $\vec{S}^t$ generated by Algorithm 2 (SIS) represent the filtering distribution $P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{z}^{\,0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal})$ with*

$$P(\vec{x}^{\,t^\intercal}, \vec{a}^{\,t^\intercal} | \vec{z}^{\,0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) = \frac{\sum_{S \in \vec{S}_{\vec{x}^t, \vec{a}^t}} w_S}{\sum_{S \in \vec{S}} w_S} \ . \tag{2.16}$$

Thus, Algorithm 2 solves the offline filtering problem (Definition 2.7) in linear time-complexity over time, scales linearly with the number of samples n_S and linearly with the number of random variables $|\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|$, i.e., solves $\texttt{OffFP}(B_0, B_{\rightarrow}, \vec{z}^{\,0:t}, \vec{b}^{1:t}, t)$ *in $\mathcal{O}(t \cdot n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ space- and time-complexity. Initializing Algorithm 2 with samples $\vec{S}$ from a previous solution $\vec{S}^{t-1}$ from $t-1$ is an exact solution to the online filtering-problem (Definition 2.8) with constant time-complexity over time $\mathcal{O}(1)$, i.e., solves* $\texttt{OnlFP}(B_0, B_{\rightarrow}, \vec{z}^{\,t-1:t}, \vec{b}^{t-1:t}, t)$ *in $\mathcal{O}(n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ time- and space-complexity.* ▲

Proof of Theorem 2.7 requires a detailed notation of samples as follows.

Notation 2.7 (Nostalgic sample). *Let $\mathfrak{S} = \left(\vec{x}^{\,0:t-1}, \vec{a}^{\,1:t-1}, \vec{x}^t, \vec{a}^t, w\right)$ denote a sample of all random variables from time 0 to time t. This means that sample $\mathfrak{S}$ carries its complete history $\vec{x}^{0:t-1}, \vec{a}^{1:t-1}$ that ends in an end-sample $S = (\vec{x}^{\,t}, \vec{a}^{\,t}, w)$. Respectively, let $\mathfrak{S}^t_{\vec{x}^{0:t},\vec{a}^{1:t}}$ denote a sample with evolution sequence $\vec{x}^{\,0:t}, \vec{a}^{\,1:t}$ and let $\vec{\mathfrak{S}}^t_{\vec{x}^{\,0:t},\vec{a}^{\,1:t}}$ denote the set of all samples $\mathfrak{S}^t_{\vec{x}^{\,0:t},\vec{a}^{\,1:t}}$ obtained at time t.*

Proof of Theorem 2.7 (Correctness of Algorithm 2, SIS). We prove that for $n_S \to \infty$, Theorem 2.7 provides an exact answer to a filtering query $P(\vec{x}^t, \vec{a}^t | \vec{z}^{\,0:t^{\intercal}}, \vec{b}^{1:t^{\intercal}})$ for the SIS case of Algorithm 2 in a dense intra-timeslice ADBN. Using Notation 2.7, Eq. 2.16 is written as

$$P(\vec{x}^{t^{\intercal}}, \vec{a}^{t^{\intercal}} | \vec{z}^{\,0:t^{\intercal}}, \vec{b}^{\,1:t^{\intercal}}) \approx \frac{\sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} \sum_{\mathfrak{S} \in \vec{\mathfrak{S}}^t_{\vec{x}^{0:t},\vec{a}^{1:t}}} w_{\mathfrak{S}}}{\sum_{S \in \vec{S}^t} w_S} \ ,$$

i.e., to sum over all samples $S_{\vec{x}^t,\vec{a}^t}$, one now sums over all possible histories leading to such samples. As some instantiations are fixed by evidence in $\vec{z}^{\,0:t}, \vec{b}^{1:t}$, the summation over all unobserved variables leads to all possible evolution sequences. Following Algorithm 1 it is evident that the weight $w_{\mathfrak{S}}$ of a nostalgic sample $\mathfrak{S}$ is uniquely determined by its evolution instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$, i.e., all samples in $\vec{\mathfrak{S}}^t_{\vec{x}^{0:t},\vec{a}^{1:t}}$ carry the weight

$$w_{\vec{x}^{0:t},\vec{a}^{1:t}} = \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{b}^k} P(A^k_{ij}) \cdot \prod_{X^k_i \in \vec{z}^{\,k}} P(X^k_i | \vec{X}^{k^{\intercal}} \backslash X^k_i, A^{k^{\intercal}}_i, X^{k-1}_i) \ . \tag{2.17}$$

Note that a weight w_S of a sample $S_{\vec{S}^t_{\vec{x},\vec{a}}}$ is not uniquely determined by its instantiation $\vec{x}^t, \vec{a}^t$. Let $N^t_{\vec{x}^{0:t},\vec{a}^{1:t}} = |\vec{\mathfrak{S}}^t_{\vec{x}^{0:t},\vec{a}^{1:t}}|$ be the number of nostalgic samples at time t with evolution sequence $\vec{x}^{0:t}, \vec{a}^{1:t}$. Then

$$P(\vec{x}^{t^{\intercal}}, \vec{a}^{t^{\intercal}} | \vec{z}^{\,0:t^{\intercal}}, \vec{b}^{1:t^{\intercal}}) \approx \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} N^t_{\vec{x}^{0:t},\vec{a}^{1:t}} \cdot w_{\vec{x}^{0:t},\vec{a}^{1:t}} \ , \tag{2.18}$$

with a normalization factor $\alpha = \sum_{S \in \vec{S}^t} w_S$. For $n_S \to \infty$, the number of a sample S, $N^t_{\mathfrak{S}}$, equals their existence probability $P(\mathfrak{S})$. Algorithm 1 generates a nostalgic sample $\mathfrak{S} = \left(\vec{x}^{0:t}, \vec{a}^{1:t}, w\right)$ with probability

$$P(\mathfrak{S}) = \prod_{k=0}^{t} \prod_{A^k_{ij} \in \vec{\beta}^k} P(A^k_{ij}) \cdot \prod_{X^k_i \in \vec{\zeta}^{\,k}} P(X^k_i | \vec{X}^{k^{\intercal}} \backslash X^k_i, A^{k^{\intercal}}_i, X^{k-1}_i) \ . \tag{2.19}$$

For brevity, let Eq. 2.17 and Eq. 2.19 ignore the special case of $t = 0$. By combining Equations 2.17–2.19, one obtains

$$P(\vec{x}^{t^\intercal}, \vec{a}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) = \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} \prod_{k=0}^{t} \prod_{A_{ij}^k \in \vec{\beta}^k} P(A_{ij}^k)$$

$$\cdot \prod_{X_i^k \in \vec{\zeta}^{\,k}} P(X_i^k | \vec{X}^{k^\intercal} \setminus X_i^k, A_i^{k^\intercal}, X_i^{k-1}) \cdot \prod_{k=0}^{t} \prod_{A_{ij}^k \in \vec{\mathscr{b}}^k} P(A_{ij}^k) \cdot \prod_{X_i^k \in \vec{z}^{\,k}} P(X_i^k | \vec{X}^{k^\intercal} \setminus X_i^k, A_i^{k^\intercal}, X_i^{k-1}) \,,$$

as $\vec{Z}^{0:t} \cap \vec{\zeta}^{\,0:t} = \vec{X}^{0:t}$ and $\vec{\mathcal{B}}^{1:t} \cap \vec{\beta}^{1:t} = \vec{\mathcal{A}}^{1:t}$ this is

$$P(\vec{x}^{t^\intercal}, \vec{a}^{t^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) = \alpha \sum_{\vec{\zeta}^{0:t-1}} \sum_{\vec{\beta}^{1:t-1}} \prod_{k=0}^{t} \prod_{A_{ij}^k \in \vec{\mathcal{A}}^k} P(A_{ij}^k)$$

$$\cdot \prod_{X_i^k \in \vec{X}^{\,k}} P(X_i^k | \vec{X}^{k^\intercal} \setminus X_i^k, A_i^{k^\intercal}, X_i^{k-1})$$

which corresponds to the exact solution for the filtering problem in Eq. 2.6 with the complete unrolled joint probability from Eq. 2.4. ■

Algorithm 2 and Theorem 2.7 can be modified to provide solutions to smoothing problems, as the following proposition states.

Proposition 2.6 (Exact sampling based solutions to the smoothing problem)**.** *By keeping track of nostalgic samples $\mathfrak{S}$ in Algorithm 2, instead of simple end-samples S, generated samples $\vec{\mathfrak{S}}$ represent any smoothing distribution $P(\vec{X}^{k^\intercal}, \vec{\mathcal{A}}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal})$, $k < t$ with*

$$P(\vec{x}^{k^\intercal}, \vec{a}^{k^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\mathscr{b}}^{1:t^\intercal}) \approx \frac{\sum_{\vec{\zeta}^{0:k-1}} \sum_{\vec{\beta}^{1:k-1}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} \sum_{\mathfrak{S} \in \vec{\mathfrak{S}}^t_{\vec{x}^{0:t}, \vec{a}^{1:t}}} w_{\mathfrak{S}}}{\sum_{S \in \vec{S}^t} w_S} \,,$$

i.e., the proportional weight of samples conforming with evidence and $\vec{x}^k, \vec{a}^k$ as part of their evolution history, for infinite samples n_S and infinitesimal numerical precision. Thus, Algorithm 2 with nostalgic samples $\mathfrak{S}$ solves `ComplSP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$ *in time-complexity as stated in Theorem 2.7, but with additional storage requirements, increasing linearly over time, for nostalgic samples histories.* ▲

Proof of Proposition 2.6 is similar to Proof of Theorem 2.7. Further, it is easy to show that the history of the nostalgic sample $\hat{\mathfrak{S}}^t$ with the highest weight w in $\vec{S}^t$ represents a solution to the *most likely explanation problem.*

For finite amounts of samples and finite numerical precision, SIS and SIR deliver approximate solutions for filtering problems. However, for finite amounts of samples, SIS techniques suffer from a "degeneracy" problem (cf., Murphy, 2012, pp. 823–831,Doucet & Johansen, 2009) in DBNs, as well as in ADBNs: Gradually, a significant amount of samples degrade and carry a very low weight, i.e., during sequential updating sampled instantiations do not fit to evidence at later timepoints. Therefore, a large amount of samples become practically irrelevant for the approximation and a few left over samples carry a high weight and are highly important. This means that an approximation error does not remain bounded over time and gradually increases. To overcome this problem a resampling procedure is introduced in SIR, which resamples n_S samples from a current sample distribution (Alg. 2, Line 10) according to their importance-weights w_S—the reason why some people speak of the "survival of the fittest" in SIR techniques.

As a resampling technique we use stratified resampling as presented by Hol, Schön, and Gustafsson (2006), which showed to be more accurate and faster than multinomial resampling (cf. Hol et al., 2006 and Massey, 2008).

Theorem 2.8 (Approximate sampling based solutions to the online filtering problem)**.** *For finite number of samples n_S, finite numerical precision and non-zero probabilities in every defined CPD, SIR, as in Algorithm 2, provides an approximate solution to the online filtering problem (cf. Theorem 2.7) with bounded error over time. SIR scales linearly in computation time and memory requirements with the number of samples n_S and linearly with the number of random variables per timestep. Thus SIR solves $\texttt{OnlFP}(B_0, B_\to, \vec{z}^{\,t-1:t}, \vec{\mathscr{b}}^{\,t-1:t}, t)$ in time-complexity $\mathcal{O}(n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$. Further, has only $\mathcal{O}(n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ for storing samples from which every filtering distribution can be obtained. Likewise, it provides an approximate solution to the offline filtering problem by solving t online filtering problems, i.e., solves $\texttt{OffFP}(B_0, B_\to, \vec{z}^{\,0:t}, \vec{\mathscr{b}}^{\,1:t}, t)$ in $\mathcal{O}(t \cdot n_S \cdot |\vec{X}^t| \cdot |\vec{\mathcal{A}}^t|)$ space- and time-complexity.* ▲

Theorem 2.8 is proven empirically in the following section. Approximate solutions to smoothing problems are possible with Proposition 2.6, but require impractical large amounts of samples (Särkkä, 2013), and are therefore not further considered in this thesis. More advanced approaches to approximate solutions to smoothing problems are given by Doucet and Johansen (2009) or, as an overview, by Särkkä (2013, pp. 167ff.), but are beyond the scope of this thesis. In principle, more advanced "particle smoothers" are based on the forward-backward-algorithm as derived in Proof of Theorems 2.5 and 2.4, where both forward- and backward-messages are obtained by a sampling procedure.

Answering queries via a sampling based approximation has significant advantages in computational complexity and required storage complexity.

Proposition 2.7 (Filtering query answering from samples)**.** *Every filtering query $P(\vec{x}_q^{\,t^\intercal}, \vec{a}_q^{\,t^\intercal} | \vec{z}^{\,0:t^\intercal}, \vec{\mathscr{b}}^{\,1:t^\intercal})$, where $\vec{x}_q^{\,t}, \vec{a}_q^{\,t}$ are partial instantiations of random variables $\vec{X}_q^t \in \vec{X}^t, \vec{\mathcal{A}}_q^t \in \vec{\mathcal{A}}^t$, are directly answerable from the set of samples $\vec{S}^t$ obtained at time t in $\mathcal{O}(n_S)$, by summation over all samples $\vec{S}_{\vec{x}^t, \vec{a}^t}$ that contain the partial instantiation $\vec{x}_q^{\,t}, \vec{a}_q^{\,t}$. This reduces the space-complexity of the filtering problem to storing n_S samples for all timeslices, i.e., t timeslices for the offline filtering problem, and 1 for the online filtering problem. Further, this reduces time-complexity of answering a query for any partial instantiation to $\mathcal{O}(n_S)$.* ▲

This means that a probability distribution has neither to be stored nor to be made explicit to marginalize out answers to queries. Further, plain linear complexity is obtained, which permits the use of ADBNs in largely scaled problems.

2.4.2 Experimental Evaluation of Approximate Inference

In the following, we empirically prove Theorem 2.8. We evaluate SIR in terms of approximation accuracy and performance as stated in Theorem 2.8 compared to exact inference. To judge the accuracy of an approximated result obtained by SIR, the Hellinger distance between approximated and exactly calculated distribution is used.

Definition 2.13 (Hellinger distance). *Let P_R, P_S, $\operatorname{rank} P_R = \operatorname{rank} P_S = N$ denote two probability distributions over the same N events $p \in P_R$, respectively $p \in P_S$. The Hellinger distance $H(P_R, P_S)$ between both distributions is defined as (cf. Gibbs & Su, 2002)*

$$H(P_R, P_S) = \frac{1}{\sqrt{2}} \sqrt{\sum_{p \in P_R} \left(\sqrt{P_S(p)} - \sqrt{P_R(p)} \right)^2}$$

The Hellinger distance is symmetric, i.e., $H(P_R, P_S) = H(P_S, P_R)$ and is bounded between 0 *and* 1. ▲

A Hellinger distance of 0 exists between two distributions that assign the same probability to common events (no error). A distance of 1 occurs if every event that is deemed impossible in P_S, is supposed to be possible in P_R. Naturally, one expects small differences between approximated and exact results, i.e., low, near 0 Hellinger distances. We refrain from using the more common Kullback-Leibler-Divergence (KLD), as negligible errors can lead to a KLD of ∞. A single event p with $P_R(p) = \varepsilon$, very small $\varepsilon > 0$, but with $P_S(p) = 0$ leads to $KLD(P_R, P_S) = \infty$.

For evaluation we consider the same models as in the previous evaluation of exact inference discussed in Section 3.2. As an evaluation against exact inference requires exact solutions of online filtering problems, we keep $|\vec{X}^t| = 4$ for a detailed evaluation over 250 timesteps (Figure 2.6, repeated 25 times) and demonstrate it on $|\vec{X}^t| = 5$ for only 25 timesteps (Figure 2.8, repeated 2 times). For $|\vec{X}^t| = 4$ one state variable X_i^t is observed per timestep and for $|\vec{X}^t| = 5$ two are observed. Observations of state variables lead to the aforementioned degeneracy problem of SIS, which is clearly evident in Figure 2.6. Further, Figure 2.6 shows over 250 evaluated timesteps of constant error for SIR, which delivers a highly satisfying approximation accuracy for 10 000 and more samples (Hellinger distance below 0.1). Judging from Figure 2.6 it seems that the Hellinger distance of an SIR approximation linearly lowers with an increasing number of samples. The constant, low Hellinger distance over long periods of inference time evident from Figure 2.6 shows that SIR indeed delivers an approximate solution to the online and offline filtering problem for a finite amount of samples, scaling linearly with the number of random variables (Figure 2.7), as stated by Theorem 2.8, which we consider empirically proven. ■

While an algorithm finds an exact solution to the online filtering problem for $|\vec{X}^t| = 4$ with one observation almost instantly, it takes more than 12 minutes for $|\vec{X}^t| = 5$ with one observation and around 6 minutes for two observations *per timestep*, as shown in Figure 2.8 (black). In contrast, an approximate solution with 10 000 000 samples is found by SIR in 16 seconds per timestep in the same experiment. Exact inference on $|\vec{X}^t| = 6$, i.e., 36 random variables in $B_\rightarrow$ is impossible, as the probability distribution alone requires 250GB of memory and computation time is expected to be more than 6 days per timestep. For $|\vec{X}^t| = 6$

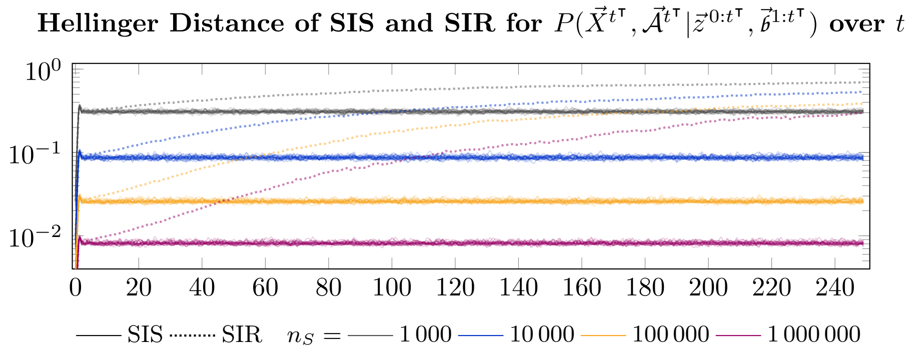

Figure 2.6. *Stratified Sequential Importance Resampling (SIR) using n_S samples solves* OnlFP$(B_0, B_\rightarrow, \vec{z}^{i-1:i}, \vec{\mathfrak{b}}^{i-1:i}, i)$ *at timestep i (abscissa) with constant and bounded error (Hellinger distance compared with exact inference, ordinate, $r \approx 0.09$), whereas Sequential Importance Sampling (SIS) has exponential growth of error over time (semilogarithmic plot). SIR plotted overlain, mean plot for SIS.*

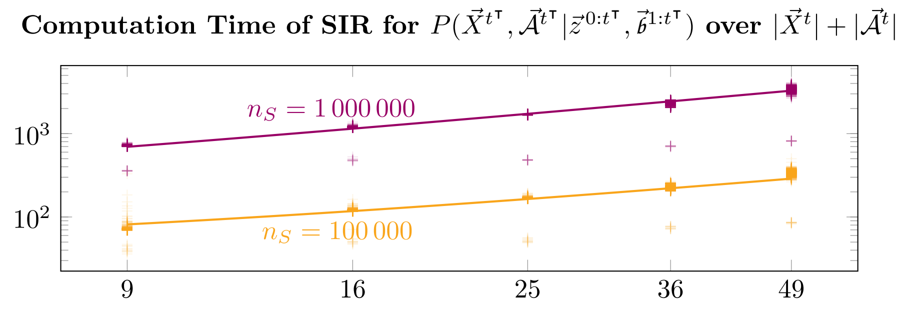

Figure 2.7. *Stratified Sequential Importance Resampling (SIR) using n_S samples solves* OnlFP$(B_0, B_\rightarrow, \vec{z}^{t-1:t}, \vec{\mathfrak{b}}^{t-1:t}, t)$ *(computation time in ms, ordinate) linearly ($r \approx 0.956$) in the number of random variables (abscissa) $N = |\vec{X}^t| + |\vec{\mathcal{A}}|$ (*double logarithmic plot*).*

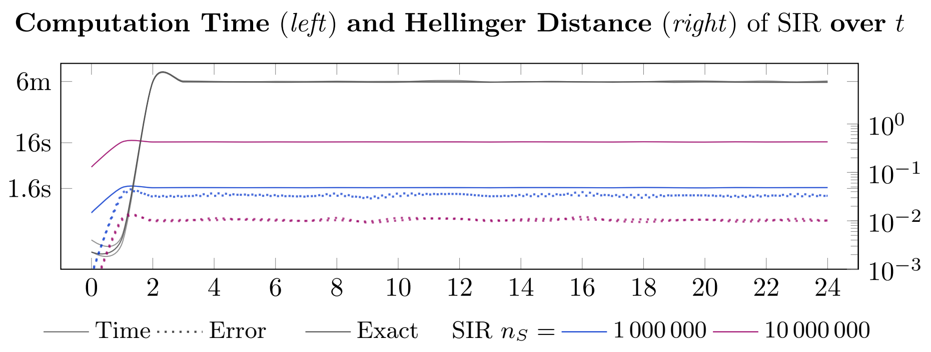

Figure 2.8. *SIR solves* OnlFP$(B_0, B_\rightarrow, \vec{z}^{i-1:i}, \vec{\mathfrak{b}}^{i-1:i}, i)$ *at timestep i (abscissa) magnitudes (semilogarithmic plot) faster (solid, left ordinate, computation time in ms) than an exact algorithm (black) and still achieves satisfying accuracy (dotted, right ordinate, Hellinger distance).*

SIR obtained an approximate solutions to the online filtering problem in 12 seconds using 5 000 000 samples, and in 2 minutes when using 50 000 000 samples, and only requires storage for all samples. In fact, Figure 2.7 shows that the SIR algorithms scales linearly with the number of random variables (evaluated on 10 different models over 50 timesteps, each) in an ADBN and shows that approximate inference in large ADBNs is even feasible when considering the absolute running time of the SIR algorithm.

2.5 Discussion and Related Work

Using an ADBN has the benefit of anticipating indirect causes per timestep in an over-the-time evolving process and ADBNs permit cyclic dependencies from local points of view. Some random variables need to be identified as activator random variables, on which minimal observation sets are enforced (Theorem 2.1: any acyclic constellation of probably active activators is allowed) or require different modeling approaches (see later Chapter 5). Note that, we come from a point of view where activators naturally exist in the modeled domain, and, in fact, Proposition 2.2 shows that classic (diagonal) DBNs are significantly restricted as well. We conclude that cyclic, as well as, diagonal ADBNs are only usable for certain instantiations of all random variables. To emphasize that the restrictions on cyclic ADBNs are far smaller than the identified restrictions on diagonal ADBNs, we explicitly compare the numbers of allowed instantiations in cyclic (Theorem 2.1) and diagonal ADBNs (Proposition 2.2) per timestep in the following propositions.

Proposition 2.8 (Number of indirection-free instantiations in diagonal ADBNs)**.** *Proposition 2.2 enforces that instantiations* $(\vec{x}^{0:t}, \vec{a}^{1:t})$ *of* $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$ *of a dense diagonal (A)DBN cannot contain two possibly active activator instantiations* a^t_{*i}*,* a^t_{i*} *to obtain an intended joint probability* $P(\vec{x}^{0:t^\intercal}, \vec{a}^{1:t^\intercal})$ *result. Thus, the set of active activators in an instantiation* $\vec{a}^t$ *must form a uniformly directed bipartite graph where isolated nodes belong to a fixed group. For* $n = |\vec{X}^t|$ *state variables, the number of these bipartite digraphs is given in Sequence* A001831 *by Sloane (2015). For every allowed activator constellation* $\vec{a}^t$*,* 2^n $\vec{X}^t$*-instantiations are possible. The total number* $\mathcal{N}^{/}{}_n$ *of regular instantiations of a joint probability in a dense inter-timeslice (A)DBN with n state variables is therefore*

$$\mathcal{N}^{/}{}_n = 2^n \cdot \sum_{k=0}^{n} \binom{n}{k} \cdot \left(2^k - 1\right)^{n-k} \qquad \blacktriangle$$

Proposition 2.9 (Number of regular instantiations in cyclic ADBNs)**.** *Theorem 2.1 enforces that instantiations* $(\vec{x}^{0:t}, \vec{a}^{1:t})$ *of* $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$ *do not contain a cycle w.r.t. active activators to obtain a well-defined joint probability* $P(\vec{x}^{0:t^\intercal}, \vec{a}^{1:t^\intercal})$ *result. Thus, the set of active activators in an instantiation* $\vec{a}^t$ *must form a directed acyclic graph (DAG), where every active activator* $a^t_{ij} \in \vec{a}^t$ *represents an edge from a node i to j in the DAG. The number of DAGs for* $n = |\vec{X}^t|$ *nodes is given in Sequence* A003024 *by Sloane (2015). For every allowed activator constellation* $\vec{a}^t$*,* 2^n $\vec{X}^t$*-instantiations are possible. The total number* $\mathcal{N}^{\circlearrowleft}{}_n$ *of regular instantiations of a joint probability in a dense intra-timeslice DBN with n state variables is therefore*

$$\mathcal{N}^{\circlearrowleft}{}_n = 2^n \cdot A003024_n \qquad \blacktriangle$$

Figure 2.9 shows both curves of Proposition 2.8 and Proposition 2.9. Even in a logarithmic plot, a cyclic ADBN has an exponential advantage in favor of a classic diagonal (A)DBN when considering the number of allowed instantiations per timestep.

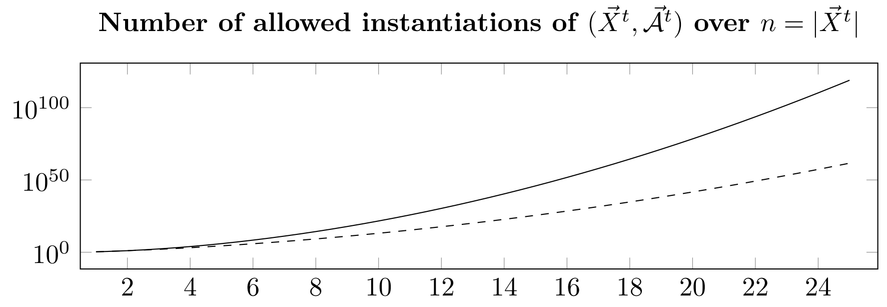

Figure 2.9. *Cyclic ADBNs ($\mathcal{N}^{\circlearrowleft}{}_n$, solid, Proposition 2.9) clearly outperform classic diagonal DBNs ($\mathcal{N}^{/}{}_n$, dashed, Proposition 2.8) in the number of allowed instantiations of $(\vec{X}^t, \vec{\mathcal{A}}^t)$ per timestep. (semi-logarithmic plot)*

The need to anticipate indirect influences originates from coarse observation timesteps, where indirect influences must be anticipated to explain observations. While we motivate coarse timesteps from an unavailability of observations at a finer time-scale, the choice of coarser timesteps is also motivated by computational feasibility. Not being bound to the finest available observation granularity relaxes the rate of needed time-updates and motivates Asynchronous DBNs by Pfeffer and Tai (2005) using Nodelman, Shelton, and Koller's (2002) Continuous-Time-BNs (CTBNs). Asynchronous DBNs provide a distributed, decentralized update of nodes in a DBN, instead of enforcing a synchronized update of all nodes in a DBN at the smallest given update frequency. Still, Asynchronous DBNs encounter the same problem given in Proposition 2.2 of anticipating indirect influences during one timestep, if observations are not known at a near continuous time granularity. CTBNs are motivated from a perpendicular perspective taken in ADBNs: observations are fully known at an infinitesimal small time granularity without hidden variables. A combining view on ADBNs and CTBNs is later discussed in Section 3.3.

Our work considers dynamic Bayesian networks as a stationary process over time, where random variables and dependencies between them represent a causal influence, and a time represents an actual flow of a wall-clock time. Further, we consider random variables with discrete domains and consider all random variables to be potentially observable. The same view on dynamic Bayesian networks is taken by Sanghai et al. (2005) in their work on relational DBNs and by Jaeger (2001) in his work on recursive relation probabilistic models. Both works follow the intention of Pearl and Russell (2003) to model causal relationships instead of reasoning processes and extend DBNs to relational domains. Both, Jaeger (2001) and Sanghai et al. (2005), consider timesteps at a near infinitesimal small time granularity s.t. only single events occur per timestep, by which both circumnavigate the problem of anticipating indirect influences under an acyclicity constraint of relations. Nevertheless, it remains an interesting future work to integrate an ADBN's adapting structure per timestep into relational domains.

DBNs with changing structure over time are also the main focus of non-stationary DBNs by Robinson and Hartemink (2008) and of time-varying DBNs by Song, Kolar, and Xing (2009). However, Robinson and Hartemink (2008) and Song et al. (2009) take a fundamentally different perspective on changing structures where changing structures represent an evolutionary change of a structure over time during a long evolutionary process and structures are slowly modified by a set of possible actions. ADBNs can rapidly change over time, namely at *every* timestep depending on a specific context. A need for a sometimes rapidly switched structure has also been presented by Yoshida, Imoto, and Higuchi (2005) by using Markov switching in linear models in an application towards gene networks, but veers away—unlike ADBNs—from a world-representing first-class declaration character of (D)BNs as emphasized by Pearl and Russell (2003). In fact, ADBNs succeed non-stationary DBNs, and every non-stationary DBN can be modeled as an ADBN, as it will be discussed, later, in Chapter 5.

Dynamic Bayesian networks in which actual dependencies depend on a specific context of random variables are a main focus of Dynamic Bayesian Multinets (DBMs) by Bilmes (2000) and represent a form of dynamic Bayesian networks similar to our work. However, in a DBM a value of a single, externally introduced, variable (Q) steers a structure of a network by introducing a new syntax using so-called dependency functions. Encoding every possible structure in one variable introduces a significant overhead in specifications and veers away from a world-representing declaration. In ADBNs multiple activator variables are part of a domain and, thus, do not introduce a modeling overhead. Further, activator random variables adhere all properties of classic random variables neither introducing a new syntax nor a new semantics. On top of that, in an ADBN a higher degree of expressiveness is achieved, as activators can be left unobserved, representing that part of a specific structure is left unknown. Nevertheless, any DBM is transformable into an ADBN, but not vice versa, as DBMs explicitly do not support proactively designed cyclic dependencies and run into problems involving indirect influences as discussed in Proposition 2.2.

A further consideration of roles and implications of context-specific independencies in probabilistic graphical models (PGMs) is done by Milch et al. (2005) focusing on an increased expressiveness of PGMs by the framework of (infinite) contingent Bayesian networks (CBNs). In CBNs edges are labeled with instantiations of some random variables, if the edge, i.e., dependency, is subject to a context-specific change. This edge-labeling follows a similar motivation as activator random variables do, and, similarly, an ADBN-activator-random-variable-instantiation $A_{XY} = {}^{+}a_{xy}$ can be seen as such an edge label in CBNs. Most notably, Milch et al. identify that certain domains cannot be modeled as one acyclic PGM and cyclic dependencies are required as we have as well motivated by the taintedness domain. However, Milch et al. (2005), similarly to Geiger and Heckerman (1996), introduce a novel calculus for their context-specific PGMs, which stands in a significant contrast to ADBNs. In ADBNs no novel calculus is required, no external "outerloop" is required, and all random variables, CPDs, and the JPD as the product of all locally defined CPDs are business as usual. Moreover, CBNs make acyclicity constraints explicit by exclusive edge-labels, i.e., it is impossible to instantiate a (unrolled) cyclic CBN, whereas, in the ADBN formalism, a cyclic ADBN may be instantiated/unrolled, as regularity constraints are never enforced or required to become explicit. As Chapter 4 will show, this circumstance is highly beneficial, as not only acyclicity is a regularity constraint. Moreover, we have shown that classical algorithms such as the forward-backward algorithm remain applicable for solving filtering or smoothing problems in ADBNs without any significant modification. While CBNs may be able to represent ADBNs to some extent, it remains unclear whether the novelly introduced CBN-calculus still allows for these algorithms to persist and remain applicable.

Our proof that an ADBN template $B_{\rightarrow}$ can be based on a cyclic graph under certain conditions is based on an equivalence of an unrolled dynamic Bayesian network whose instantiations are equivalent to instantiations of an acyclic dynamic Bayesian network. This equivalence allows ADBNs to be designed with cyclic structures proactively, as cycles are first resolvable in an instantiation and, in fact, ADBNs represent all structures with which observations are explainable. Sets of equivalent classes of graphs are also considered by Acid and de Campos (2003) for more efficient structure learning in Bayesian networks. Learning ADBNs represents a novel research area of Bayesian network learning, as a structure is not known in advance and can only be designed, i.e., learned, proactively and is therefore discussed in the following Chapter 3.

A view on DBNs as, e.g., taken by us, is significantly more expressive than viewing DBNs as an unrolled hidden Markov model, where specific variables must be constantly observed and a view over "time" is used to resolve cycles, instead of representing a wall-clock time. Notwithstanding, if cyclic models are intended to be designed, one could switch to chain graphs as described by Drton (2009) or general Markov networks. However, by switching to (partially) undirected models one loses the direct and intuitive interpretation of locally specified conditional probability distributions by introducing a burden of computing non-local normalization quotients. On top of that, an undirected model inherently models a different domain as undirected edges imply symmetric influences, whereas in an ADBN one can specify asymmetric influence strengths. Further, a cyclic Markov network models a steady state influence domain, where influences let random variables converge to a stable state. The latter is an application of a diagonal DBN, where a cyclic directed graphical model is unfolded into a diagonal DBN and simulated over a long timeperiod to obtain said stable state. Note that, the "misuse" of time to simulate this cyclic behavior is exactly the problem that motivated this thesis and leads to the problem of being unable to anticipate indirect influences in one timestep of a DBN, if time is supposed to represent an actual wall-clock time. Nevertheless, as Proposition 2.3 shows, an ADBN can model any DBN and can therefore also be used to simulate feedback loops, i.e., cyclic dependencies that do not resolve to be acyclic.

Our work is significantly based on activator random variables, which exploit context-specific independencies in Bayesian networks. The presence of context-specific independencies and their advantages for easier specifications and representations of local parameters has notably been described by Boutilier et al. (1996). Poole and Zhang (2003) extend work by Boutilier et al. (1996) in order to exploit context-specific independencies for more efficient exact reasoning in (dynamic) Bayesian networks by considering effective independencies in instantiated structures. Notwithstanding, their work was fundamental for our work, but the authors solely focused on performance optimization, whereas we focus on increasing expressiveness of DBNs for anticipating indirect influences correctly by rapid adaptations to (even unknown) contexts.

Recent advances in probabilistic logic languages by Fierens et al. (2015) and de Raedt et al. (2007) can encode and simulate (dynamic) Bayesian networks using probabilistic rules. Often probabilistic logic languages are not subject to acyclicity constraints as a reasoning process is not based on Bayesian inference, but are reduced to other commonly known computational problems. This means that similar answers to inference queries in ADBNs are obtainable using, e.g., works by de Raedt et al. (2007) by adequate specifications of probabilistic rules, but then procedural rules are designed instead of utilizing a DBN as a first-class representation of a causal process.

We have discussed the complexity of problems over time and have shown that in an ADBN one obtains the same, and even simpler (see Example 2.6) complexities as in classic DBNs for solving commonly known query answering problems. However, like in any other DBN, the complexity in terms of number of random variables for solving query problems demands approximate inference techniques. We demonstrate that approximate inference techniques solve inference problems in ADBNs under reasonable complexities and derive an approximation technique for dense intra-timeslice ADBNs. As dense intra-timeslice ADBNs represent a superclass (Proposition 2.3) of all intra-timeslice ADBNs, the technique is universally applicable to other ADBN models.

Further, we show that familiar approaches to approximate inference in DBNs remain applicable in cyclic ADBNs, even over long periods of inference time. However, unlike exact inference, the demonstrated approximate inference technique requires an effective topological ordering to become explicit, which is obtained for acyclic observations by a topological topological sort of $\vec{\mathfrak{b}}^{1:t}$. Performing a topological sorting does not represent a bottleneck and the requirement does not come as a surprise, as sequential importance sampling is an adaptation of likelihood weighting, which does require a topological ordering similar to prior and rejection sampling (cf., Russell & Norvig, 2010, pp. 530–535). However, not only contexts of activators can assure regularity, as Motzek and Möller (2015a) (cf. Chapter 4) have shown. It is therefore desirable to find an approximate inference technique for ADBNs that does not require a topological ordering to become explicit beforehand. Considering that Gibbs sampling is an approximate inference approach for Bayesian network that is not based on a topological ordering and is only based local conditional independencies, an adaption of Gibbs sampling towards approximate inference in dynamic (activator) Bayesian networks is a promising research field for future work.

Notwithstanding, it is possible that a set of observations is not regular in a particular situation, e.g., when time granularity is chosen too coarse or too few observations are acquired. In such a particular situation, a cyclic ADBN may return spurious results, which is verifiable by Theorem 2.1, i.e., satisfaction of regularity conditions is used as a *postponed* sufficiency check to obtain well-defined results. Note that in any situation where an observation does not assure regularity in a cyclic ADBN, a diagonal ADBN will deliver spurious results as well, as every cyclic observation immediately includes the necessity to anticipate indirect influences. Furthermore, in contrast to work by Milch et al. (2005), these conditions neither become explicit, nor are enforced, nor need be checked once one trusts that observations will be regular, as solutions to commonly known inference problems are *not* based on any of these conditions and are defined directly on the global semantics of ADBNs. Moreover, Chapter 5 will introduce a solution to insufficient structural context information. Nevertheless, in a situation where a context is actually observed to be cyclic, i.e., too many active activators are observed, an action to assure well-definedness is needed. Such an action can be to investigate on novel regularity predicates, as done in Chapter 4, which shows that even cyclic observations may be regular. If no novel regularity predicates are applicable, an invasive action to the observation is required. Such an invasive action implies a willful modification of the real world, which stands in a significant conflict with our views on Bayesian network as a first-class representation of the real world. We, therefore, refrain from deriving such an invasive action and advise to rethink the underlying model and problem in such situations. Still, we briefly sketch a possible approach to modifying observations to assure regularity in Chapter 7.

2.6 Conclusion

We have shown that indirect causes in dynamic Bayesian networks cause conflicts in representing causality. These conflicts arise from using a modeled dimension for assuring syntactic requirements. By identifying activator criteria of random variables in (dynamic) Bayesian networks, we introduce ADBNs and are able to move acyclicity constraints from a design phase to a later operation phase. Without the need of external reasoning frameworks, degrading a Bayesian network to a reasoning process, we obtained a solid mathematical basis similar to Bayesian networks with a causally correct consideration of indirect causes in dynamic Bayesian networks. ADBNs provide the ability to intrinsically let a DBN adapt to observed contexts, where a DBN's structure is not known in advance and changes over time while maintaining a Bayesian network as a world-representing first-class declaration.

In this chapter, we show that ADBNs exist under an exploitation of context-specific independencies of activator random variables. Still, by considering further properties of local CPDs, we later show in Chapter 4 that not only activator random variables are able to assure regularity.

Local semantics of specified CPDs enable an expert to validate individual CPD parameters, instead of requiring a validation of inferred results. This is highly beneficial for applications where large datasets of ground truth are unavailable and experts' assessments must be incorporated directly. We utilize ADBNs and exploit their local semantics later in Chapter 6.

While an ADBN allows for a local and causal representation of dependencies by an expert, such an expert might not always exist, or must be assisted in specifying locally interpretable CPDs. In such a particular situation, CPD parameters must be learned from possibly incomplete datasets, which poses an interesting new problem in ADBNs addressed in the following chapter.

Chapter 3

Learning Dynamic Bayesian Networks where Structures Cannot be Known in Advance

While ADBNs are suited as a first-class representation of the world by the use of local and global semantics, not always there is a domain expert available who has the precise expertise to define a complete (A)DBN for a particular domain. In that particular situation the expert must be supported, or even be replaced, by a learning approach for (A)DBNs, which learns an ADBN from long, but incomplete, sequences of partial observations. However, learning ADBNs poses two imminent novel challenges: *an ADBN's structure can be rapidly changing at each single timestep* and *a structure cannot be known in advance.*

To more clearly motivate the following abstract statements and problems, consider the following example from the taintedness domain.

Example 3.1 (ADBN learning challenges)**.** *Given a long, but incomplete, sequence of observations of compliance infringements in a company, it is of utmost interest to learn a cyclic ADBN to precisely predict future compliance violations and to reconstruct events that lead to previous infringements. However, on one day not all message transfers were obtainable, and are therefore missing in the data available for learning. One must now consider all potential message transfers during that day, based on remaining information about the surrounding days, e.g., known compliance adherences or violations. If one would know a precise structure and all parameters of a required ADBN, one could reason about all potential message transfers by solving a classical offline smoothing problem. However, these structures and parameters shall be learned. Moreover, the biggest challenge is that a required structure in question depends on these message transfers and rapidly changes at every timestep, i.e., first once one knows the message transfers, one can decide on a structure, which is required in the first place to reason about these transfers.*

Commonly, one is concerned with two problems when learning DBNs: (i) incomplete data, i.e., some instantiations of random variables are missing in data, and (ii) an unknown structure, i.e., the presence of random variables and their respective domain is known, but their causal relationships are unclear. Moreover, one distinguishes between two forms of learning: parameter learning, i.e., learning parameters of local CPDs, and structural learning, i.e., learning dependency structures between random variables. Naturally, structural learning and parameter learning must be intertwined.

Classically, problems (i) and (ii) are approached in three (incrementally repeated) steps by (1) obtaining a structure, (2) restoring hidden/missing variables in incomplete data based on a structure and parameter candidates, and (3) optimizing, or rather, learning new parameters. If a structure is known, (1) is trivial, and an incremental repetition of (2) and (3) learns parameters from incomplete data, which is known as the EM-algorithm. If a structure is unknown, (2) cannot be performed, and different approaches exist for obtaining structures, such as a fixed bootstrapped heuristic search, or, an incrementally repeated heuristic search based on newly optimized parameters. Friedman, Murphy, and Russell (1998) call the former a parametric EM approach, where a structure of a (D)BN is once fixed during (1) and an algorithm iterates between (2) and (3) to perform parameter learning. If a parametric EM approach does not deliver satisfying results, the complete procedure is repeated based on a different choice in (1). Friedman (1998) introduces the structural EM algorithm as a more efficient learning approach towards BNs under an unknown structure, which has been applied to DBNs by Friedman et al. (1998). The structural EM algorithm iterates between all three steps, i.e., incrementally optimizes and learns a structure as well as CPD parameters at every repetition.

In classical structural EM algorithms a structure is assumed to be constant over time, i.e., it is assumed that there exists one structure which is able to explain a process at time $t = 1$, as well as at, say, time $t = 500$. However, the previous chapter shows that an ADBN rapidly adapts to a specific context at every timestep, which must be incorporated by a learning approach. That structures do not necessarily remain constant over time has, e.g., been discussed in the form of non-stationary DBNs, where structures slowly evolve over time. Robinson and Hartemink (2008) show that even for evolving structures, learning remains possible and is highly relevant for interpreting data. However, a structure only changes *slowly* over time, and not at every timestep. Therefore, Robinson and Hartemink's (2008) approach is not directly applicable for ADBN learning. However, a much deeper challenge is imminent for finding a learning approach towards ADBNs: Classical learning approaches assume that for the act of restoring values of missing variables a structure remains constant. This means that a calculation of a distribution of potential value-candidates is based on *one* (previously optimized) structure. This is where the pertinacious challenge lies: In ADBNs an effective structure is not knowable in advance, i.e., a required structure changes and is only identifiable in retrospect given a specific context, i.e., a set of instantiations in data or observations. This means that there exists no single structure on which value candidates can be inferred, and classic learning algorithms are not applicable for ADBN learning. If one fixes a structure prior to inference, one immediately fixes instantiations of activators, which, if missing in data, must be inferred based on all possible encoded structures of a timeslice.

The contribution of this chapter can be summarized as follows: By proposing a learning approach for ADBNs for problems where effective structures are not known in advance, are rapidly changing over time, and are not directly evident from data while learning, we effectively fuse structure learning with parameter learning into one atomic task to learn ADBNs from incomplete data under unknown structures in this chapter. While we motivate cyclic dependencies to support a causal and local parametrization of ADBNs, it might be possible that "diagonal" (A)DBNs can be learned, whose parameters do not bear a local semantics, but which deliver satisfying inference results anticipating indirect influences. To investigate the learnability of diagonal (A)DBNs we show, in this chapter, that not even the most general form of diagonal DBNs can be trained to anticipate indirect causes, neither by classical learning approaches nor by our novelly proposed learning procedure designated for ADBNs, which is able to learn highly accurate cyclic ADBNs.

This chapter is structured as follows: In Section 3.1, we derive and propose a learning approach for ADBNs from data containing hidden variables. Based on the derived approach, we discuss and experimentally evaluate different scenarios of hidden variables and modeling approaches in Section 3.2. We discuss our results and related works in Section 3.3 and conclude with Section 3.4.

3.1 Learning Structures where Structures are not Knowable

Learning any probabilistic model is based on large amounts of data in a dataset. A dataset $\vec{d}^{0:t}$ is a time series of instantiations of some random variables in $\vec{X}^{0:t}$ and $\vec{\mathcal{A}}^{1:t}$, i.e., every datum $\vec{d}^{t}$ is not necessarily a full instantiation of a timeslice t. Without loss of generality, we consider that learning is performed only on a single (long enough) time series $\vec{d}^{0:t}$ of data. This means, a DBN represents a process evolving over time, where time represents an irreversible wall-clock time at a specific granularity, instead of an arbitrary construct of operation-"time"-slots. This means, an initial BN fragment B_0 is only instantiated once, making it impossible to learn prior probabilities of B_0.

Often, learning-approaches and, especially, EM-approaches are introduced informally, by reducing learning to a simpler case of complete datasets, i.e., all values of variables are available for learning. Learning parameters from complete datasets effectively reduces to counting how often an event associated with a parameter is seen, i.e., learning from complete datasets is linear in the number of parameters and the size of the dataset. Based on learning from complete datasets, learning from incomplete data is often introduced via an idea that virtual data counts are created by inferring all possible value candidates from one incomplete datum. One has to hopefully expect that such an approach is applicable to cyclic ADBNs and that no external frameworks analyzing structures separately are required. As a cyclic ADBN represents multiple structures and ADBNs are subject to a novel acyclicity constraint, applicability of classical learning approaches is not granted, but is desired. We therefore, carefully derive a classic EM approach from a pure probabilistic point of view. To do so, one has to carefully distinguish between to-be-learned parameters and a numerical instantiation of them.

Notation 3.1 (Parameters). *Let $\vec{\Theta}$ represent the vector of all to-be-learned parameters as variables. Let $\vec{\vartheta}$ represent a specific numerical instantiation of all parameters $\vec{\Theta}$. Let $P_{\vec{\vartheta}}(\cdot)$ represent a probability derived based upon a set of a parameter instantiation $\vec{\vartheta}$, i.e., $P_{\vec{\vartheta}}(\cdot)$ is a* number. *Let $P_{\vec{\Theta}}(\cdot)$ represent a probability based on a set of parameter variables in $\vec{\Theta}$, i.e., $P_{\vec{\Theta}}(\cdot)$ is an* equation *with variables. Given a set of instantiated parameters $\vec{\vartheta}$, there exists a specific probability $P_{\vec{\vartheta}}(\vec{d})$ of observing a dataset $\vec{d}$, i.e., the likelihood of the data.*

A dataset $\vec{d}^{0:t}$ induces a probability distribution over all possible instantiations $\vec{x}^{0:t}$, $\vec{a}^{1:t}$ conforming with $\vec{d}^{0:t}$ under parameters $\vec{\vartheta}$ that could have been observed, also called the dataset distribution:

$$P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t}) = \alpha \cdot P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})$$

As before, for inference, a dataset $\vec{d}^{0:t}$ is seen as observed random variables $(\vec{z}, \vec{\beta})^{0:t}$.

Given a dataset distribution $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$, the probability of observing (note, not to have observed) $\vec{d}^{0:t}$ under parameters $\vec{\Theta}$ is

$$P_{\vec{\Theta}}(\vec{d}^{0:t}) = \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t}) \cdot P_{\vec{\Theta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) \ , \tag{3.1}$$

further called the likelihood of the dataset, and where instantiations in $\vec{d}$ uniquely identify the instantiation of respective random variables in $\vec{D}$. Note that both sums do not iterate over all instantiations; instantiations that are not iterated over have probability zero given $\vec{d}^{0:t}$, as they do not conform to the dataset.

Following a maximum likelihood approach, it is the goal to find a parameter instantiation $\vec{\vartheta}^*$ of $\vec{\Theta}$ that maximizes the probability of observing $\vec{d}^{0:t}$, i.e.,

$$\vec{\vartheta}^* = \arg\max_{\vec{\Theta}} P_{\vec{\Theta}}(\vec{d}) = \arg\max_{\vec{\Theta}} \log\left(P_{\vec{\Theta}}(\vec{d})\right) \ , \tag{3.2}$$

where an expectation maximization (EM) approach iterates between calculating a distribution $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$ (E-step) and Eq. 3.2 (M-step). In fact, there is a trivial algorithm to obtain $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$, but which requires an intractable amount of memory, which is why in the following an extended smoothing distribution is used.

Definition 3.1 (Extended smoothing problem)**.** *Given a probabilistic knowledge base B_0, $B_\rightarrow$, the extended smoothing problem is the problem of determining all extended smoothing distributions k, $k < t$ over all random variables in timeslices k and $k-1$ while considering evidence $\vec{z}^{0:t^\intercal}, \vec{\beta}^{1:t^\intercal}$ until time t. This is, to obtain*

$$P(\vec{X}^{j^\intercal}, \vec{\mathcal{A}}^{j^\intercal}, \vec{X}^{j-1^\intercal}, \vec{\mathcal{A}}^{j-1^\intercal} | \vec{z}^{0:t^\intercal}, \vec{\beta}^{1:t^\intercal}) \ , \ \forall j : 1 \leq j < t \ .$$

We denote a parametrized extended smoothing problem as $\texttt{ExtdSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\beta}^{1:t}, t)$. ▲

Finding exact solutions to extended smoothing problems is discussed in Theorem C.1 in the appendix, and, in summary, is linear in t and only requires storage for a distribution over all random variables of two timeslices.

For brevity, further derivations are given in Appendix C as the proof for the following theorem on the derived learning approach. Notwithstanding, when learning ADBNs one must consider activator criteria, for which we use a partition of activator instantiations, defined as follows.

Definition 3.2 (Activator instantiation partitioning)**.** *Let $X^i_\lambda \in \vec{X}^i$ be a state variable and let $P_\Theta(x^i_\lambda | \vec{x}^{i^\intercal} \backslash x^i_\lambda, \vec{a}^{i^\intercal}_\lambda, x^{i-1}_\lambda)$ be a to be learned parameter. We partition $\vec{a}^i_\lambda$ into two vectors $\vec{a}^i_\lambda = \langle {+\vec{a}^i_\lambda}, \neg\vec{a}^i_\lambda \rangle$ containing active and inactivate* *activator random variables. Then, let $\vec{x}^{i^\intercal} \backslash x^i_\lambda = \langle \vec{x}^i_{\blacksquare\lambda}, \vec{x}^i_{\square\lambda} \rangle$ be a partition of relevant and irrelevant dependencies of X^i_λ under an instantiation $\vec{a}^i_\lambda$ corresponding to the activator criteria from Definition 2.2, such that*

$$\begin{aligned} \forall k \ {+a}^i_{k\lambda} \in {+\vec{a}}^i_\lambda : x^i_k \in \vec{x}^i_{\blacksquare\lambda} \\ \forall k \ \neg a^i_{k\lambda} \in \neg \vec{a}^i_\lambda : x^i_k \in \vec{x}^i_{\square\lambda} \ . \end{aligned}$$

Respectively let $\vec{X}^i_{\blacksquare\lambda}, \vec{X}^i_{\square\lambda}$ represent the corresponding random variables of instantiations $\vec{x}^i_{\blacksquare\lambda}, \vec{x}^i_{\square\lambda}$. ▲

Using Definition 3.1 and Definition 3.2 one obtains an EM procedure.

Theorem 3.1 (EM procedure)**.** *Repeatedly evaluating*

$$P^*_\vartheta(x^i_\lambda | \vec{x}^{i^\intercal} \backslash x^i_\lambda, \vec{a}^{i^\intercal}_\lambda, x^{i-1}_\lambda) = \frac{\gamma'(X_\lambda = x_\lambda)}{\gamma'(X_\lambda = {+x}_\lambda) + \gamma'(X_\lambda = \neg x_\lambda)} \tag{3.3}$$

$$\textit{with} \qquad \gamma'(X_\lambda = x_\lambda) = \sum_{i=1}^{t} \sum_{\vec{\zeta}^{i-1} \backslash X^{i-1}_\lambda} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^{i} \backslash \vec{A}^i_\lambda} \sum_{\vec{X}^i_{\square\lambda}} P_{\vec{\vartheta}}(\vec{X}^{i-1^\intercal} \backslash X^{i-1}_\lambda, x^{i-1}_\lambda, \vec{\mathcal{A}}^{i-1^\intercal}, \vec{X}^{i^\intercal}_{\square\lambda}, \vec{x}^{i^\intercal}_{\blacksquare\lambda}, \vec{a}^{i^\intercal}_\lambda, \vec{\mathcal{A}}^{i^\intercal} \backslash \vec{A}^i_\lambda | \vec{d}^{0:t}) \ ,$$

and

$$P^*_\vartheta(A^i_{\mu\nu}) = \frac{\gamma'(A_{\mu\nu} = a_{\mu\nu})}{\gamma'(A_{\mu\nu} = {+a}_{\mu\nu}) + \gamma'(A_{\mu\nu} = \neg a_{\mu\nu})} \tag{3.4}$$

$$\textit{with} \qquad \gamma'(A_{\mu\nu} = a_{\mu\nu}) = \sum_{i=1}^{t} \sum_{\vec{\zeta}^{i-1:i}} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^{i} \backslash A^i_{\mu\nu}} P_{\vec{\vartheta}}(\vec{X}^{i-1^\intercal}, \vec{\mathcal{A}}^{i-1^\intercal}, \vec{X}^{i^\intercal}, \vec{\mathcal{A}}^{i^\intercal} \backslash A^i_{\mu\nu}, a^{i^\intercal}_{\mu\nu} | \vec{d}^{0:t}) \ .$$

for all parameters in $\vec{\Theta}$, is an algorithm that learns model parameters from incomplete data, where even structural information, i.e., activators, are hidden. Every iteration increases the likelihood of being able to observe $\vec{d}^{0:t}$ under an optimized parameter set $\vec{\vartheta}^*$ and converges to a local optimum. At every iteration, one evaluates $\gamma'(\cdot)$, and respectively solves `ExtdSP`$(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$ *to obtain $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$, (*E*xpectation-step), and consecutively evaluates $P^*_\vartheta(\cdot)$, (*M*aximization-step).* ▲

That the proposed algorithm maximizes the likelihood in each iteration is proven in Proof of Theorem 3.1 by a continued, detailed derivation on the procedure, which can be summarized as follows: By summing out a specific parameter, Eq. 3.2, i.e., the maximization of the likelihood is analytically solved in a closed form by partial derivation and finding zeros. Under careful consideration of identical parameters due to activator constraints, one obtains all equations stated by Theorem 3.1 after several transformations. The convergence of the proposed EM procedure is shown empirically in the following section.

To summarize, a learning approach for dense intra-timeslice ADBNs, where a structure cannot be known in advance, shows similar structure to EM algorithms for DBNs. In fact, one obtains a commonly known form of "expected counts" (see also Russell & Norvig, 2010, pp. 820-824) as a closed form for optimized parameters $\vec{\vartheta}^*$, as hoped for. Note that, at no time an effective structure is made explicit, and throughout the learning procedure a structure of each timeslice remains unknown. Therefore, we say that learning ADBNs fuses structure- and parameter learning into one atomic phase.

A dense intra-timeslice ADBN represents a superclass of all possible Markov-1 intra-timeslice DBN structures and is allowed to contain cyclic dependencies, as long as, a final instantiation of such network is acyclic. This means that multiple structures and, therefore, multiple joint probability distributions are represented by one ADBN. This is a crucial point, as, the E-step is performed based on multiple joint probability distributions. This is a required property, as first an expected distribution obtained *past* an E-step allows a structural analysis. Classic approaches apply heuristics *prior* to an E-step to improve or change structures (or rather, a set of potential structure candidates), but an E-step is performed on every single separately. This means that, if one uses only one structure to create "virtual data points" from incomplete data, one would immediately fix the to-be-inferred instantiations of activators as well, i.e., create too few and incorrect virtual data points.

In the previous chapters we have introduced the taintedness domain, where cyclic ADBNs arise naturally, and diagonal (A)DBNs cannot be parametrized causally from local perspectives. In the following section we learn taintedness domains from datasets and evaluate the proposed learning approach on cyclic ADBNs and answer the question whether diagonal (A)DBNs could actually learn suitable representations of the taintedness domain.

3.2 Convergence and Hidden Variables

This section explores and empirically evaluates different situations of hidden variables while learning ADBNs. For empirical evaluation, we learn models from multiple datasets gathered from simulated taintedness domains over long time sequences with $N = |\vec{X}^t|$ employees, i.e., long (incomplete) periods of observations of compliance infringements in a virtual company. We generate these datasets by a simple sampling procedure under randomly generated CPDs following Definition 2.2 in the taintedness domain; in detail by an SIR approximation (cf. Section 2.4) using one sample. To avoid impossible situations we constrain local CPDs to contain only probabilities in the range $[0.01, 0.99]$ and restrain all priors to be 0.5.

To prove the convergence of learned parameters $\vec{\vartheta}^*$ (as stated by Theorem 3.1) towards the original model parameters, we consider the absolute error between learned and original CPDs in different situations of missing data for cyclic and diagonal (A)DBNs. To validate learned models, we generate new sequences of observations and compare the accuracy of inference results for filtering and smoothing problems. To do so we calculate the Hellinger distance, as previously used in Section 2.4, between results obtained by the learned model and results obtained from the original model. Again, one must expect small differences of inference results, i.e., a low, near 0 Hellinger distance.

In summary, more than 200 experiments confirm that learned CPDs, using the learning procedure from Theorem 3.1, converge to a local optimum that closely represents the original CPDs, even from incomplete datasets with a very low observability spaces (see Figure 3.2). In combination with the analytical Proof of Theorem 3.1, we consider Theorem 3.1 as proven. ■

In the following paragraphs, we apply and discuss results of the learning procedure for different observability spaces and structural models.

Learning from complete data If $\vec{d}$ contains full instantiations $\vec{x}^{0:t}, \vec{a}^{1:t}$, then a probability distribution $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$ allocates all probability mass at $P_{\vec{\vartheta}}(\vec{d}^{0:t} | \vec{d}^{0:t}) = 1.0$, i.e., all other possible to-be-learned instantiations have probability zero of having been seen. Informally this means, given complete data for learning, learning parameters reduces to pure counts of observations.

As expected, learning from complete data achieves highly accurate results in representing original local CPDs (Figure 3.2: —◆—), and one obtains nearly identical inference results (Figure 3.1: —◆— Hellinger distance near 0) compared to results obtained from the original model. Note that, by adhering a correct prior $P(+a_{ij}^t) = 0.5$ in a dataset, not all possible activator settings are formed. This means, not all individual subset-structures are learned sequentially, but that a complete dense ADBN containing all possible substructures is learned as a bulk from data, without the need to analyze effective structures. As learning from complete data does not require a calculation of $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal} | \vec{d}^{0:t})$, it is linear in the size of the dataset and the number of parameters (further discussed in the following section.)

Can diagonal ADBNs learn indirect influences? In summary, no. The necessity of ADBNs is motivated by a causal design approach and the need to anticipate indirect influences. It is demonstrated that if a diagonal (A)DBN is parametrized from a causal design perspective, spurious results are returned and indirect influences are not anticipated. However, this raises an interesting question: *Can diagonal (A)DBNs learn from indirect influences and learn to anticipate, or at least, simulate them?* In order to answer this question, we repeat the previous experiment with complete datasets for four employees and different structures: **(a)** a cyclic, dense intra-timeslice ADBN (as previously), **(b)** a dense, diagonal inter-timeslice ADBN, **(c)** a dense, diagonal inter-timeslice DBN, and **(d)** a tree-structured DBN. The difference between (b) and (c) is that (b) enforces activator constraints as done in Eq. 3.3, but (c) does not and is learned by a classic EM algorithm. Learning a cyclic ADBN from complete datasets consisting of 10 000 datapoints delivers inference results with an extremely low Hellinger distance to the original model of around 0.03. To absolutely assure that sufficient datapoints are available for learning, we increase the dataset size to 50 000 datapoints in this experiment.

Case (d), a tree-structured DBN as called by Ghahramani (1997), represents a previously undiscussed modeling approach where as much intra-timeslice dependencies are modeled as possible and remaining dependencies are bent to a previous timeslice. It was not discussed previously, as it is an arbitrary decision on which dependencies are bent and which not. Generally, one could analyze activator distributions to obtain a most likely structure, which is correctly represented by intra-timeslice dependencies, and remaining dependencies are bent to a previous timeslice, forming a hybrid model consisting of intra- and inter-timeslice dependencies. However, in our domain, all structures are equally likely, which is why one has to arbitrarily chose some, e.g., chose all downward-pointing (compare Figure 2.2, but for four employees) dependencies to be intra-timeslice. Nevertheless, from a reasoning perspective, a desired influence between every random variable is present in each timeslice. However, these influences are then modeled from a completely acausal perspective as dependencies then represent cause→effect as well as effect→cause relationships at the same time. Therefore, such acausal hybrid models stand in conflict with the view taken on Bayesian networks in this thesis. Moreover, due to this acausality and inconsequence, the parametrization of hybrid models becomes practically impossible for a human, and parameters, i.e., CPDs, do not provide any local, nor intuitive understanding. Such hybrid acausal models then require a spurious inverse-reasoning mechanism over the complete model for specification and interpretation. Hence, tree-structured DBNs could be learnable in theory, but learned model parameters do not provide any meaning and learned models are only useful in black-box approaches.

As one expects, experiments showed that learned CPDs heavily differ from the original models. Still, diagonal ADBNs might be able to learn a *different* set of parameters which "emulates" an anticipation of indirect influences. When validating learned non-cyclic models against observations requiring an anticipation of indirect influences, experiments shown in Figure 3.1 show that diagonal and tree-structured (A)DBNs deliver highly inaccurate results (Figure 3.1: Hellinger distance of —+—, ⋯⋯, —+— is above 0.15), even if learned from a reasoning perspective. In fact, filtering and smoothing distributions obtained from learned non-cyclic ADBNs are almost as different from the original distribution as a distribution obtained from completely random CPDs is (Figure 3.1: —▲—). Note that all experiments are performed using the same amount of data, and that learning a tree-structured DBN is as computationally expensive as learning a cyclic ADBN. A tree-structured ADBN performs slightly more accurately than a diagonal ADBN, as, at least, half of all dependencies are coincidentally modeled causally correctly, and some indirect influences implied by observations are then handled correctly. However, a tree-structured model remains inconsequent, as it is ambiguous which dependencies are modeled in which way.

Interestingly, cases (b) and (c), i.e., a diagonal DBN with and without activator constraints, show similar results. Analyses of learned CPDs in both cases revealed that models are learned where dependencies on other random variables are eliminated, and a process solely based on a state-variable's history is learned, i.e., all CPDs encoded $P(X_i^t|\vec{X}^{t-1^\intercal}, \vec{A}_i^{t-1\,t^\intercal}) = P(X_i^t|X_i^{t-1})$. This emphasizes that diagonal (A)DBNs simply do not understand indirect influences, and cannot be learned from data containing indirect influences.

Furthermore, experiments on tree-structured DBNs (case d) were repeated without enforcing activator constraints. In such DBNs a desired influence between every random variable exists during one timeslice, and significantly more parameters are learned, which could provide an increased accuracy compared to other modeling approaches. However, experiments shown in Figure 3.1 show that unconstrained tree-structured DBNs (⋯⋯) perform even worse than their constraint counterpart, which may be explained by an overfitting to the training dataset.

The following learning cases do not consider diagonal models anymore and focus how cyclic ADBNs behave in extreme cases of incomplete data.

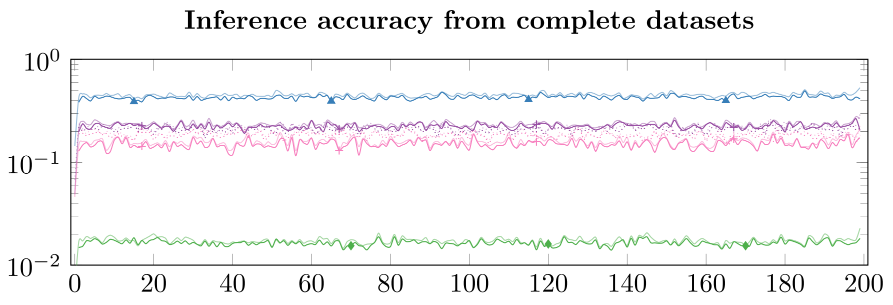

Figure 3.1. *Inference accuracy of learned models from complete datasets. Hellinger distance (ordinate) of filtering and smoothing (lighter color) results displayed per inference-timepoint (abscissa). Classic diagonal (A)DBNs (constrained* —+—*, unconstrained* ⋯+⋯*) and tree-structured (A)DBNs (constrained* —+—*, unconstrained* ⋯+⋯*) achieve unsatisfying inference accuracy. A learned cyclic model (*—◆—*) nearly comes to the same conclusions as the original model. For reference, results of randomly generated CPDs are given (*—▲—*). Average of 50 experiments displayed.* Semi-logarithmic plot.

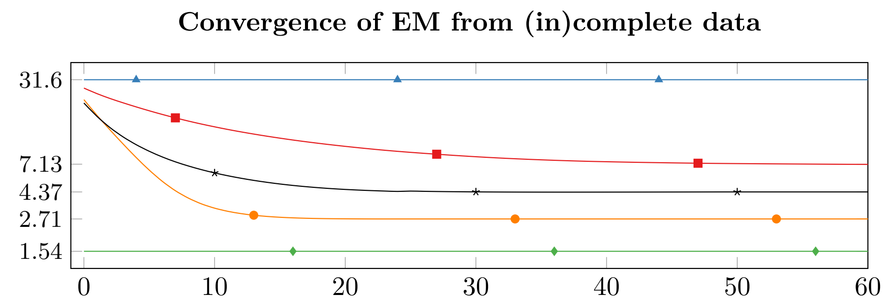

Figure 3.2. *Convergence of learned CPDs towards the original CPDs (mean of absolute error, ordinate) over number of EM iterations (abscissa) from complete data (*—◆—*, 50 experiments), incomplete state variables X^t (*—●—*, 75 exp.), incomplete activator random variables A^t_{ij} (*—★—*, 70 exp.) and incomplete X^t, A^t_{ij} (*—■—*, 10 exp.). For reference, errors of randomly generated CPDs are given (*—▲—*, 50 exp.). Learned from* 10 000 *data points in randomly generated ADBNs with $N = 3$.* Semi-logarithmic plot.

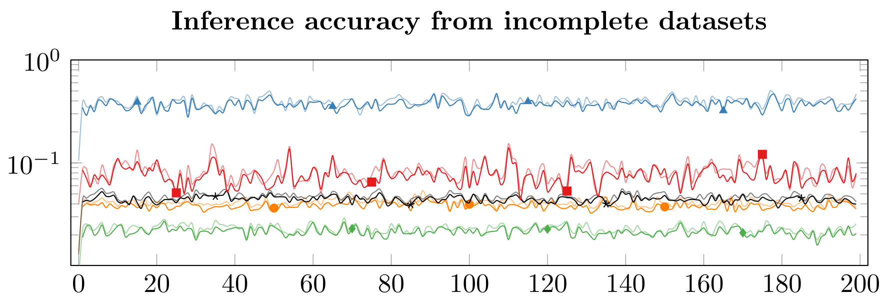

Figure 3.3. *Inference accuracy of learned models from incomplete datasets. Hellinger distance (ordinate) of filtering and smoothing (lighter color) results display per inference-timepoint (abscissa).* Same experiments and colors as in Figure 3.2. *For reference, results for randomly generated CPDs (*—▲—*) are given. Learned models nearly come to the same conclusions as the original model, even when structural information ($\vec{A}^t$) is missing.* Semi-logarithmic plot.

Hidden state variables are a common problem in classic hidden Markov models, where certain state variables are constantly unobservable and whose expected values must be restored during learning. In ADBNs no constant structure exists, and we extend hidden variables in ADBNs by introducing the problem of varying observability spaces, where roles of observable and unobservable variables rapidly change at every timestep. Therefore, we randomly exclude instantiations of variables $\vec{X}^t$ from each dataset at every timestep with a probability of 0.5. This means that inside one dataset $\vec{d}^{0:t}$ in every data $\vec{d}^t$ different instantiations of $\vec{x}^t$ are missing, and are sometimes missing completely. Figure 3.2 shows that after few EM iterations, learned local CPDs converge to a local optimum closely resembling the original CPDs (Figure 3.2: —●—). Further, inference results from a learned model are close (Figure 3.3: —●— below 0.1) to results from the original model.

Hidden activators in learning datasets $\vec{d}$ represent a novel problem of most interest. If activator instantiations are missing in a dataset, actual structural information for every timeslice is missing and must be restored based only on remaining information. Note that, at least as much information must remain to assure regularity of every learned ADBN (cf. Section 3.3). Therefore, we observe a set of activators to be deactive in our experiments, i.e., the only known structural information are *in*dependencies of a timestep and remaining structures are actively learned. Indeed, this fuses structural and parameter learning into one atomic phase.

Classic structural EM algorithms, e.g., proposed by Beal and Ghahramani (2003) or Friedman et al. (1998), score graphical models to avoid overly *complex* networks and to avoid small local optima formed by an EM approach. In order to learn ADBNs without sufficient structural information, a similar approach is incorporated into ADBN learning. Our experiments show that without full activator instantiations, an EM approach converges towards too *simple* networks, where no dependencies are present at all, i.e., a prior of near 0 for all $P({+}a_{ij}^t)$ is learned. To overcome a tendency towards too simple models, one is able to fix a prior of activators to a suitable estimation, e.g., the original value. Experimental results of Figure 3.2 show that with a fixed prior for activator variables, learned CPDs quickly converge to a local optimum (Figure 3.2: —⋆—) and deliver inference results in the same accuracy as when dealing with hidden state variables (Figure 3.3: compare —●— and —⋆—, both below 0.1).

Hidden activators and state variables are the lowest form of observability spaces. Structural information is partially hidden, and effects of remaining influences between state variables are hidden as well. In this case, a structure is not known in advance and structural context information is missing as well, but information, from which structural context information can be restored, is (partially) missing as well. Nevertheless, multiple experiments show that the proposed EM algorithm with fixed activator priors converges to a local optima with acceptable error (Figure 3.2: —■—) and which achieve satisfactory inference results compared to the original model (Figure 3.3: —■—). Note that for these experiments the length of each dataset was not increased, i.e., significantly fewer individual instantiations in a datapoint are available for learning.

3.3 Discussion and Related Work

Our approach for learning ADBNs is based on an EM approach for hidden variables in BNs as, e.g., presented by Ghahramani (1997) and Friedman (1997). In fact, Theorem 3.1 resembles a familiar "virtual" data count approach. Problems of hidden variables in addition to hidden structures have also been well studied by Friedman (1998) and have been applied to DBNs by Friedman et al. (1998), but all are subject to the assumption that a structure is knowable before random variables are instantiated. Thus, structural EM algorithms are greatly applicable to learning diagonal ADBNs, which, however, return spurious results, but structural EM algorithms are not applicable to learning cyclic ADBNs, as there exists no single structure on which an E-step can be performed.

A need for evolving structures over time has also been investigated by Robinson and Hartemink (2008) in the form of non-stationary DBNs and by Song et al. (2009) in the form of time-varying DBNs, which both find great applications in the field of biomedicine. Robinson as well as Le Song present learning approaches for such networks from complete datasets, i.e., no hidden variables are considered, and focus on *slowly* evolving processes, where a structure remains almost constant between two timeslices. Using an ADBN has the benefit of being able to proactively model cyclic dependencies and to rapidly change a structure depending on a context at every timestep and does not require an explicit scoring and optimization of learned structures over time, even from incomplete datasets. Models in which a context steers an effective structure are a main point of (dynamic) Bayes multinets (DBMs) by Geiger and Heckerman (1996) and by Bilmes (2000) as discussed in earlier chapters, and Geiger and Heckerman (1996) presents learning approaches towards DBMs as well. Still, in a DBM a structure is still bound to classic acyclicity constraints, and therefore would lead to the same spurious results as one obtaines in diagonal ADBNs. Still, as (diagonal) ADBNs are able to represent DBMs, as discussed in Section 2.5, the presented learning approach for ADBNs is applicable to DBMs as well. The need to anticipate indirect influences arises from naturally timesliced data, evolving over time, known only in coarse grained timeslices containing hidden variables. These are contrary domains suited for Nodelman et al.'s (2002) continuous time Bayesian networks (CTBNs) and related learning approaches by Nodelman, Shelton, and Koller (2003). Still, ADBNs and the presented learning approach can be seen as an addition to CTBNs, where observations affect variables in an uncertain coarse-grained temporal interval, while anticipating all potential implications of hidden variables during these intervals. Moreover, a cyclic ADBN contains all possible sequences formable by activator and state variables, and not solely the most likely sequences. Therefore, when learning cyclic ADBNs from data containing frequent, repeated sequences, such sequences must manifest themselves in CPDs and Markov-n ADBNs are a direct application to mining frequent sequences from incomplete, temporal datastreams. As shown by Saleh and Masseglia (2008), such frequent sequences might only exist in certain context, which is seen as a hidden variable in learning (cylic) ADBNs.

The computational complexity of the proposed learning approach for incomplete data is exponential in the number of random variables and in the number of missing instantiations in data. Please note that the same complexity, and the same amount of learning data, is required for learning a classic DBN, e.g., a tree-structured DBN and, still, DBNs show to be useful in practice. However, as we have shown, classic DBNs such as tree-structured DBNs are unable to anticipate indirect influences in extreme cases and return spurious results. Our learning approach shows a similar schema to classic learning approaches and is based on a smoothing distribution for which approximate solutions can be found via sampling sampling based approaches. If "virtual" data points are created by a sampling procedure, learning (A)DBNs reduces the complexity of counting samples. Note that, for the scope of this chapter we used a dense intra-timeslice ADBN, i.e., the most general intra-timeslice ADBN where all activator random variables are present. The dense intra-timeslice is chosen s.t. the proposed learning approach is universally applicable, as it encodes all possible intra-timeslice DBN structures. Notwithstanding, in practice, not always all dependencies are subject to a context-specific change, which will lead to fewer random variables naturally. Nevertheless, learning an ADBN with 25 random variables from complete datasets with 100 000 datums only takes 1.7s on average.

In this work, we consider situations where at least sufficient structural *in*dependence information remains in data. By adequate specification of local activator CPDs, it is possible to constrain ADBNs to assign zero-probabilities to non-regular instantiations by assigning a belief over possible activator constellations, i.e., structures. This means that by an adequate modeling approach, deeply discussed in Chapter 5, learning from data without any structural information, i.e., completely missing activator instantiations, is possible. This is closely related to the fact that a prior distribution over activator random variables must be fixed in our approach and is directly related to finding adequate heuristics finding and optimizing structures in structural EM algorithms. In fact, penalizing overly complex structures, already represents a prior belief about potential structures. As Chapter 5 discusses, a distribution over potential structure candidates is directly embedded into the ADBN formalism and represents a plain additional random variable, i.e., does not require an external framework for externally creating DBN candidates. The latter ideas on intrinsically representing a distribution over all possible structural candidates are achieved by specific parameter settings of CPDs in Chapter 5, for which this chapter on learning is unaffected. In one formalism therefore, structural- and parameter learning is collapsed to one atomic problem and integrates reasoning under multiple structure candidates. The latter idea is a novel view on work by Friedman and Koller (2003), but without external frameworks and solved in one world-representing first-class declaration of a Bayesian network.

3.4 Conclusion

In this chapter we have derived a learning approach for ADBNs, where a structure is only knowable in a specific context, i.e., an instantiation of random variables. In fact, we are able to show that classical "virtual data counts" approaches remain applicable, and, once again, that cyclic ADBNs do not introduce any modeling or computational overhead. We demonstrate that even if parts of structure relevant information are missing and, thus, no structure is knowable, ADBNs are still able to learn model parameters which reciprocally decide an effective structure. We have shown that not even the most general form of DBNs is able to learn the taintedness-domain of the running example, and that cyclic ADBNs are causally required. As ADBNs represent superclasses of Markov-1 DBNs, parameter and structural learning of ADBNs are fused into one atomic phase of constraint parameter learning, while being able to handle hidden variables and hidden structural information.

With a novel modeling approach, introduced and discussed later in Chapter 5, learning ADBNs fusing parameter- and structural- learning into one atomic phase will be directly applicable to learning (dynamic) Bayesian networks under hidden variables and unknown structures in one atomic phase without requiring heuristics.

So far, regularity of ADBNs is assured by observations of activators. The following chapter reconsiders regularity of ADBNs and shows that not only *activators* are relevant for regularity, but contexts formed by all random variables can assure regularity. This will show that cyclic activator observations can still be regular.

Chapter 4

Exploiting Innocuousness in Bayesian Networks

Boolean combination functions in Bayesian networks (BNs) are often credited a property stating that if a dependence is observed to be inactive (i.e., a precondition observed to be *false*) it shall not "cause any harm" and its arc becomes vacuous, i.e., could have been left out. We call such a property an "*innocuousness*" property of conditional probability distributions (CPDs) and it is of significant interest to formalize this property and to exploit it in ADBNs. Arcs becoming vacuous in specific contexts extends regularity conditions of observations and instantiations postulated by Theorem 2.1. To give an example in the taintedness domain: *messages from compliant employees, cannot taint anyone, i.e., dependencies on compliant employees are irrelevant, and are not subject to regularity constraints.* Unfortunately, one can neither specify such an innocuousness property in CPDs, nor formalize it up to now. Vacuous dependencies have shown to be of valuable interest for efficient reasoning in Bayesian networks, and an innocuousness property is widely associated with, e.g., noisy-or combination functions. Further, being able to explicitly specify an innocuousness property in CPDs provides more precise representations of the world, as demanded and emphasized by Pearl and Russell (2003).

Formalizing an innocuousness property can almost be achieved with previously discussed context-specific independencies (CSIs) introduced by Boutilier et al. (1996), but consider the following example: Say, random variable X is directly dependent on Y and C and one specifies a CPD $P(X|Y, C)$. CSIs represent that X becomes locally independent of Y in a specific context $C = c \in \mathrm{dom}(C)$, but X stays dependent on Y in another context $C = c' \in \mathrm{dom}(X)$. Boutilier et al. (1996) formalize: $P(X|Y, c) = P(X|c)$ holds, if $\forall y, y' \in \mathrm{dom}(Y), \forall x \in \mathrm{dom}(X) : P(x|y, c) = P(x|y', c)$, but there exists a $c' \in \mathrm{dom}(C)$ s.t. $\exists y, y' \in \mathrm{dom}(Y), \exists x \in \mathrm{dom}(X) : P(x|y, c') \neq P(x|y', c')$.

However, if one intends to formalize an innocuousness property stating that a context $C = c \in \mathrm{dom}(C)$ "removes" *itself* (and not only *another* variable), i.e., one would like to specify "$P(X|Y, c) = P(X|Y)$" in a CPD, one runs into the problem that a formal definition is neither available nor easily possible: the allegedly irrelevant random variable C in question is in fact the one that ought to be relevant for specifying the independence. Informally, this means that *other* variables can become independent in a context, but not the context *itself.*

The previous chapters introduced ADBNs as a novel form of DPGMs that are subject to different acyclicity constraint, by exploiting properties of activator random variables, which themselves represent CSIs. While the form ADBNs increases the global expressiveness of DPGMs, this chapter shows that, in fact, activators increase the expressiveness of local CPDs by providing a formal definition of an innocuousness property. Further, we exploit innocuousness properties in this chapter to relax acyclicity constraints of ADBNs, namely that not only instantiations of activators assure regularity.

The contribution of this chapter can be summarized as follows: By formalizing a yet unexpressed innocuousness property in CPDs, expressiveness of local semantics of CPDs is increased, and acyclicity restrictions on (A)DBNs are relaxed. Based on graph enumeration techniques we quantitatively explore new relaxations of syntactic restrictions of graphical models for Bayesian networks.

This chapter is structured as follows: In Section 4.1 we introduce the innocuousness property, an abstract concept of multiple innocuousness contexts, and formally define innocuousness using activator random variables in ADBNs. Subsequently, in Section 4.2 we exploit the innocuousness property for relaxing restrictions posed on (A)DBNs. We discuss our results and related work in Section 4.3 and show that the utility of (A)DBNs is significantly enhanced by exploiting innocuousness properties. The chapter concludes in Section 4.4 giving an outlook to future chapters of this thesis.

4.1 Innocuousness

We introduce innocuousness informally as "an inactive node does not cause any harm," but one is unable to give a formal definition for such a property in CPDs of classic (D)BNs. Often "accountability," i.e., $P({+x}|\neg *) = 0$ as defined by Cozman (2004), is confused with the innocuousness property, but semantically $P({+x}|\neg *) = 0$ can also represent that exactly one *false*-dependence is responsible for $P({+x}|\neg *)$ being 0 (cf. Section 4.3).

As an extension to context-specific independencies (CSIs) from Boutilier et al. (1996), we define a concept of *innocuousness contexts*, with fewer restrictions of CSIs. Boutilier et al. (1996) provide a formal definition for CSIs, where a variable X becomes independent of a variable Y in a context $\vec{C} = \vec{c} \in \text{dom}(\vec{C})$, where $X, Y \notin \vec{C}$. Therefore, CSIs allows specifying properties such as $P(X|Y, \vec{c}) = P(X|\vec{c})$ in local CPDs. But, $X, Y \notin \vec{C}$ prevents one from specifying that a context $\vec{C} = \vec{c}$ removes one of *its own* random variables $C \in \vec{C}$, e.g., "$P(X|Y, c) = P(X|Y)$." Using activators in ADBNs we extend Boutilier's work to innocuousness contexts. We formally define that in a context $\vec{C} = \vec{c}$, a context variable $C \in \vec{C}$ itself becomes irrelevant, which we call self-reflexive independence. Say, in a context $C = c \in \text{dom}(C)$, X shall become independent of C, given $C = c$, e.g., $P(X|c, A_{CX}, \vec{Z}) = P(X|A_{CX}, \vec{Z})$. Using an activator-enriched CPD we define this to hold for binary activator random variables if

$$\begin{aligned}
&\forall x \in \text{dom}(X), \forall \vec{z} \in \text{dom}(\vec{Z}) : \\
&\qquad P(x|c, {+a_{CX}}, \vec{z}) = P(x|c, \neg a_{CX}, \vec{z}) = P(x|*, \neg a_{CX}, \vec{z}) \ ,
\end{aligned} \tag{4.1}$$

where $\vec{Z}$ represents remaining further dependencies of X. Extensions to non-boolean activator random variables are straightforward.

This means, given $C = c$, A_{CX} becomes irrelevant for X, i.e., X becomes independent of A_{CX}. As A_{CX} can be instantiated in any form now from X's point of view, one is able to frankly assume $\neg a_{CX}$. Then, according to the deactivation criterion of an activator, X becomes independent of C given $\neg a_{CX}$, or rather X becomes independent of C given $\neg c$, which is exactly what was intended, i.e., an innocuousness property of C for X.

Now, assume to specify a CPD $P(X|C, A_{CX}, Q, \vec{Z})$ where the innocuousness property of a variable is only in place in a further context. For example, there exists a variable Q that activates the innocuousness property of C only given $Q = q \in \mathrm{dom}(Q)$ for X. In this case, Eq. 4.1 only holds for specific $\vec{z} \in \mathrm{dom}(\vec{Z})$. This means, one innocuousness context is defined by instantiations of multiple random variables, denoted as follows.

Notation 4.1 (Innocuousness contexts). *Activator random variables are marked with a dot, e.g., $\dot{A}_{YX}$, if they are subject to become irrelevant in specific contexts. We denote a context in which Y is innocuous for X as a* innocuousness context*, denoted as a left superscript on A_{YX}. If a context is met and Y is innocuous for X, we say that A_{YX} stands in the innocuousness context. For the first example, this is*

$$P(X|C, {}^{C=c}\dot{A}_{CX}, \vec{Z}) \ .$$

Further, this notation covers the previously discussed toggle variable Q as well: Only in the context $Q = q$ and *$C = c$, X becomes independent of C, as A_{CX} becomes freely instantiable. For this situation one writes*

$$P(X|C, {}^{Q=q,C=c}\dot{A}_{CX}, Q, \vec{Z}) \ ,$$

where $\vec{Z}$ represents further dependencies of X, but without X, Q, C and A_{CX}.

Moreover, one random variable might stand in multiple different innocuousness contexts, which we denote as innocuousness context vectors.

Notation 4.2 (Innocuousness context vectors). *Variables become innocuous in multiple contexts. Multiple innocuousness contexts $\varphi_{A_{YX}}$ of one activator A_{YX} are encapsulated in a vector $\vec{\varphi}_{A_{YX}}$ and are delimited by ; . An innocuousness context vector $\vec{\varphi}_{A_{YX}}$ can also be seen as a Boolean formula, where all contexts are disjunctions and a context is a conjunction of instantiations.*

This notation allows marking contexts in which an activator becomes irrelevant and could have been chosen to be deactive, and thus modifies the topology. Definition 4.1 describes the explicit specification of innocuousness in CPDs.

Definition 4.1 (Activator innocuousness). *Let $\Phi_{A_{YX}}$ be the vector of random variables used in a context $\varphi_{A_{YX}}$ associated with A_{YX}. Every innocuousness context $\varphi_{A_{YX}} \in \vec{\varphi}_{A_{YX}}$ is then defined to hold*

$$\begin{aligned} &\forall x \in \mathrm{dom}(X), \forall \vec{z} \in \mathrm{dom}(\vec{Z}) : P(x|\varphi_{A_{YX}}, {+a}_{YX}, \vec{z}) = P(x|\varphi_{A_{YX}}, \neg a_{YX}, \vec{z}) \\ &= P(x|\{\varphi_{A_{YX}} \backslash y \in \mathrm{dom}(Y)\}, y, \neg a_{YX}, \vec{z}) = P(x|\{\varphi_{A_{YX}} \backslash y\}, *, \neg a_{YX}, \vec{z}) \ , \end{aligned} \tag{4.2}$$

with remaining arbitrary dependencies of X on other random variables $\vec{Z}$ and $\vec{z}$ as an arbitrary instantiation of those, excluding A_{YX} and $\Phi_{A_{YX}}$. ▲

Frankly, with Definition 4.1 one can formulate the same CSIs as Boutilier et al. (1996), but, further, one can specify previously mentioned self-reflexive independencies. One is thus able to explicitly express $P(X|\{\varphi_{A_{YX}}\backslash y\}, y, A_{YX}, \vec{Z}) = P(X|\{\varphi_{A_{YX}}\backslash y\}, A_{YX}, \vec{Z})$ as demonstrated in the following example.

Example 4.1 (Activator innocuousness). *Continuing the taintedness domain from Example 2.6, we assume a noisy-or combination function for each CPD of a state X^t. With a noisy-or combination, every activator random variable (a message transfer) $\dot{M}^t_{XY}$ stands in the innocuousness context $\varphi_{M^t_{XY}} = \neg x^t$. One can now explicitly represent that Claire is not influenced by an untainted Earl, i.e., $P(C^t|C^{t-1}, D^t, \neg e^t, A^t_{DC}, A^t_{EC}) = P(C^t|C^{t-1}, D^t, A^t_{DC}, A^t_{EC})$, by fixing*

$$\begin{aligned}
&\forall\ C^t, C^{t-1}, D^t, A^t_{DC}, A^t_{EC} : \\
&\qquad P(C^t|C^{t-1}, D^t, \neg e^t, A^t_{DC}, {+a}^t_{EC}) = P(C^t|C^{t-1}, D^t, \neg e^t, A^t_{DC}, \neg a^t_{EC}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad \overset{(by\ Def.\ 2.2)}{=} P(C^t|C^{t-1}, D^t, {+e}^t, A^t_{DC}, \neg a^t_{EC})
\end{aligned}$$

in the respective CPD specification of C^t (likewise for untainted Don).

In this example, an arc in $B_{\rightarrow}$ representing a dependency of some X on some Y becomes vacuous in a context of the variable Y itself, which was previously impossible to formalize and impossible to define in a CPD without activators. Note that a dependance of Y on X is still highly relevant in the context of the variable Y. This is beneficial for more efficient reasoning and a higher causal accuracy of independence declarations in all DBNs with activator random variables.

Further, so far we did not consider any properties of CPDs for possible acyclicity constraints in ADBNs, and only focus on instantiations of activators. In the next section, we consider innocuousness properties of CPDs and relax restrictions posed on graphical models.

4.2 Exploiting Innocuousness

By considering properties of CPDs of state variables $\vec{X}^t$, we relax restrictions of Theorem 2.1 by introducing innocuousness contexts as a further acyclicity constraint. Note that in an ADBN, these checks and constraints are only sufficient conditions for achieving sound results and are not required for necessary calculations, if, e.g., observations can be trusted to fulfill these restrictions. Informally, as arcs become vacuous and could be left out once a variable stands in its innocuousness context, cycles are possibly eliminated as well. We, therefore, reconsider the well-definedness of an ADBN in the following theorem.

Theorem 4.1 (ADBN well-definedness revised). *An ADBN $(B_0, B_{\rightarrow})$ is well-defined, if an ADBN is well-defined according to Theorem 2.1. An ADBN $(B_0, B_{\rightarrow})$ is well-defined for every instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ of random variables $\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t}$, if for all t, $\vec{x}^{0:t}, \vec{a}^{1:t}$ satisfy the following conditions:*

$$\begin{aligned}
\forall x, y, z \in \vec{X}^t : \mathfrak{A}(x, z)^t, \mathfrak{A}(z, y)^t \rightarrow \mathfrak{A}(x, y)^t \\
\neg\exists q : \mathfrak{A}(q, q)^t\ ,
\end{aligned} \tag{4.3}$$

with a regularity predicate $\mathfrak{A}(i,j)^t$ that is defined as

$$\mathfrak{A}(i,j)^t = \begin{cases} \textit{false} & \textit{if} \quad \neg a^t_{ij} \vee \vec{\varphi}_{A^t_{ij}} \in \vec{x}^{0:t}, \vec{a}^{1:t} \\ \textit{true} & \textit{otherwise} \end{cases} ,$$

with the innocuousness context vector $\vec{\varphi}_{A^t_{ij}}$ seen as a disjunction of multiple contexts $\varphi_{A^t_{ij}}$ for activator A^t_{ij}, as defined in Definition 4.1. For every well-defined ADBN, semantics as $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{01:tt^\intercal})$ is sound and given equivalently to DBN semantics given as the product of all locally defined CPDs. ▲

Theorem 4.1 refines the well-definedness of an ADBN by refining the set of regular instantiations. According to Theorem 4.1, not only acyclic instantiations are regular, but also instantiations where activators stand in their innocuousness context are regular. Note that an ADBN's definition, syntax and semantics is unaffected, and Proposition 2.4 on the JPD on a dense intra-timeslice ADBN remains applicable. This means that previously derived algorithms for solving common query answering problems, and algorithms for learning ADBNs remain sound and valid.

We prove Theorem 4.1 by showing that for any regular instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ of $\vec{\mathcal{A}}^{1:t}$, $\vec{X}^{0:t}$, under Theorem 4.1, Proof of Theorem 2.1 holds.

Proof of Theorem 4.1 (ADBN well-definedness revision). Proof of Theorem 2.1 shows that for any regular instantiation under Theorem 2.1 a topological ordering exists and that the JPD stated by Proposition 2.4 is equivalent to a JPD of a Bayesian network using the same CPD parameters with an equivalent topological ordering. Per timeslice, activator random variables A^t only follow a lexicographical ordering, and Proof of Theorem 2.1 is analogous for every instantiation that induces the same topological ordering over random variables $\vec{X}^{0:t}$. Therefore, we prove Theorem 4.1 by showing that any regular instantiation under Theorem 4.1 is topologically equivalent to regular instantiation under Theorem 2.1 for which Proof of Theorem 2.1 holds.

Definition 4.2 (Topology equivalence). *Given an ADBN $(B_0, B_\rightarrow)$, an instantiation $(\vec{x}^{0:t}, \vec{a}^{1:t})_1$ is* topologically equivalent *to an instantiation $(\vec{x}^{0:t}, \vec{a}^{1:t})_2$, if for both the same topological ordering $\prec$ exists in $(B_0, B_\rightarrow)$.* ▲

Generally, in an acyclic (A)DBN, for every arbitrary instantiation $(\vec{x}^{0:t}, \vec{a}^{1:t})_*$ the same topological ordering $\prec_*$ exists. In a cyclic ADBN, a topological order is defined (at "runtime") by a minimal set of deactive activator random variables in an instantiation $\vec{a}^{1:t}$ of $\vec{\mathcal{A}}^{1:t}$ (c.f. Proposition 2.3). In that case, some state variables $\vec{X}^t$ become independent of state variables $\vec{X}^t_E$ which previously created cycles and prohibited a topological ordering $\prec$. Under Definition 4.1, an active activator A^t_+ might stand in a context $\varphi_{A^t_+}$, which renders A^t_+ innocuous or irrelevant. It is straightforward from Definition 4.1 that A^t_+ can then be seen as deactive from a topological perspective, which we call *topologically deactive.* Two instantiations $(\vec{x}^{0:t}, \vec{a}^{1:t})_1$, $(\vec{x}^{0:t}, \vec{a}^{1:t})_2$ then share the same topological ordering $\prec$, i.e., are topologically equivalent, if the set of *topologically-deactive* activators in $(\vec{a}^{1:t})_1$ is a superset of deactive activators in $(\vec{a}^{1:t})_2$.

Any regular instantiation $(\vec{x}_1^{0:t}, \vec{a}_1^{1:t})$ under Theorem 4.1 that is not regular under Theorem 2.1 contains instantiations that let some activators stand in their innocuousness context and are topologically deactive. Therefore, $(\vec{x}_1^{0:t}, \vec{a}_1^{1:t})$ is *topologically* equivalent to a regular instantiation $(\vec{x}_1^{0:t}, \vec{a}_2^{1:t})$ under Theorem 2.1 for which Proof of Theorem 2.1 holds. Thus, Proof of Theorem 2.1 proves Theorem 4.1. ■

Note that, while a structure of two topologically equivalent instantiations follows the same topological ordering, the joint probability for each instantiation need not be the same, as one needs to consider priors of $\vec{\mathcal{A}}^{1:t}$.

Theorem 4.1 shows and it is proven that ADBNs can be based on cyclic graphs and remain well-defined even for some observations that contain *cyclic* activator evidences. Previously, the latter was forbidden, but is now well-defined, as the following example shows.

Example 4.2 (Restriction relaxation example)**.** *We continue Example 4.1 without the made observations. Say, one observes $\neg e^1$, $^+m^1_{CD}$, $^+m^1_{ED}$, $^+m^1_{DE}$, and one can neglect all other transfers. One observes that Claire influenced Earl indirectly through Don, but Earl did not become tainted. This observation should lower our belief in Claire, as well as, Don being tainted. However, observations $^+m^1_{ED}$, $^+m^1_{DE}$ obviously lead to a cycle, which is prohibited by Theorem 2.1. Fortunately, the observation $\neg e^1$ meets the innocuousness context $\varphi_{M^1_{ED}}$ (cf. Example 4.1, noisy-or), i.e., E^1 is innocuous for D^1 given $\neg e^1$. Therefore, this observation fixes all instantiations of $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$ to regular instantiations under Theorem 4.1. Consequently, one obtains $P(C^1|\vec{z}^{0:1^\intercal}, \vec{b}^{1^\intercal}) \approx \langle 0.34, 0.66\rangle$ and $P(D^1|\vec{z}^{0:1^\intercal}, \vec{b}^{1^\intercal}) \approx \langle 0.39, 0.61\rangle$, which both correspond to the intuitive interpretation of the observation. To recall, the initial beliefs about Claire and Don were $P(C^0) = \langle 0.5, 0.5\rangle$ and $P(D^0) = \langle 0.6, 0.4\rangle$, respectively. The observation that Earl has received documents, but has not become tainted, implies that Claire and Don are likely to be untainted. Therefore, $P(^+c^1|\dots) < P(^+c^0)$ and $P(^+d^1|\dots) < P(^+d^0)$.*

Note that still this observation cannot be handled by a diagonal alternative, as one needs to anticipate the indirect influence: $\neg e^1$ tells us indirectly *something about C^1, namely that $\neg c^1$ is now more likely than without the observations.*

This example shows that the cyclic model can handle a larger set of observation constellations in contrast to a diagonal alternative. The next section generalizes these advantages for a general model in terms of the number of regular instantiations per timestep in cyclic, diagonal and innocuous ADBNs.

4.3 Discussion, Comparison and Related Work

In this section, we investigate how cyclic ADBNs compare to classical diagonal (A)DBNs. As discussed before, only regular instantiations of $(\vec{X}^t, \vec{\mathcal{A}}^t)$ in a timestep t lead to well-defined cyclic ADBNs, and diagonal (A)DBNs run into problems if instantiations are contain implications of indirect influences. Further, we explore how the exploitation of innocuousness properties can relax these restrictions. In fact, we show that with this exploitation one can handle significantly more instantiations, and that cyclic ADBNs heavily outperform their diagonal counterparts w.r.t. expressivity.

In cyclic ADBNs, instantiations of $(\vec{X}^t, \vec{\mathcal{A}}^t)$ during a timestep t are enforced to be regular under Theorem 2.1 and are relaxed under Theorem 4.1. For diagonal ADBNs, instantiations are restricted s.t. no indirect influences can occur (Proposition 2.2). Notwithstanding, innocuousness properties also relax this restriction. For a comparison, we consider the running Example 2.3 consisting of N employees, i.e., state variables $\vec{X}^t$, and likewise $N(N-1)$ message exchange variables in every network fragment $B_\rightarrow$.

Without considering innocuousness properties, i.e., one does not exploit contexts of an instantiation $\vec{x}^t$, the number of regular $\vec{\mathcal{A}}^t$ instantiations in a cyclic ADBN corresponds the number of DAGs (Sloane, 2015, Seq. A003024), as discussed in Proposition 2.9. In a classic diagonal ADBN no indirect effects are anticipated, and thus, no "interlocking" (possibly active) activator instantiations of $\vec{\mathcal{A}}^t$ are allowed, as discussed in Proposition 2.8. This corresponds to the number of uniformly directed bipartite graphs where isolated nodes belong to a fixed group (Sloane, 2015, Seq. A001831). For every of these instantiations, 2^N instantiations of all $\vec{X}^t$ exist (for a binary domain of all state variables).

To emphasize the effect of exploiting innocuousness context, we consider that q% out of all N state variables $\vec{X}^t$ in an ADBN $B_\rightarrow$ are innocuous states $\vec{X}_Q$, meaning that every state X_i^t "is not harmed" by any of these $X_Q^t \in \vec{X}_Q^t$ if $\neg x_Q^t$. This implies that every activator $\dot{A}_{ij}^t$ stands in the context $\varphi_{A_{ij}^t} = \neg x_i^t$, if $X_i^t \in \vec{X}_Q^t$. Thus, $\mathrm{rank}(\vec{X}_Q^t) = Q = \lfloor N \cdot q \rfloor$, for which flooring operations lead to wavy lines in Figure 4.1.

Innocuousness properties significantly relax restrictions opposed on cyclic ADBNs in the number of regular instantiations, which are enumerable as follows.

Proposition 4.1 (Number of regular instantiations in cyclic ADBNs)**.** *Given a cyclic, dense intra-timeslice (A)DBN $(B_0, B_\rightarrow)$, $N = |\vec{X}^t|$ and Q innocuous states, the number of regular instantiations $\mathcal{N}^{\mathcal{O}}{}_{N,Q}$ under Theorem 4.1 is*

$$\mathcal{N}^{\mathcal{O}}{}_{N,Q} = 2^{N-Q} \cdot \sum_{k=0}^{Q} 2^{k(N-1+N-k)} \cdot A003024_{N-k} \cdot \binom{Q}{k} .$$

$\mathcal{N}^{\mathcal{O}}{}_{N,Q}$ origins from the consideration that one has between $k=0$ to $k=Q$ "deactive" innocuous nodes. All non-innocuous state variables are freely instantiable, i.e., 2^{N-Q} instantiations. Activators regarding a dependance between $N-k$ nodes are bound to instantiations that form a DAG. For $N-k$ nodes, $A003024_{N-k}$ many DAGs exist. In each DAG combination, k deactive nodes exist, whose each $N-1$ activators (read: "outbound-activator") are freely instantiable, i.e., $2^{k(N-1)}$ instantiations. In each DAG combination, $N-k$ active nodes exist, whose activators regarding a dependance towards the k deactive nodes are freely instantiable, i.e., $2^{(N-k)k}$ further instantiations. For each combination, one has $\binom{Q}{k}$ options to choose which (labeled) innocuous states are deactive. ▲

Notwithstanding, the restriction that only indirect-free instantiations of $\vec{\mathcal{A}}^t$ are allowed in diagonal ADBNs (Proposition 2.2) is also relaxed by considering innocuousness properties of $\vec{X}^t$. The number of allowed instantiations in diagonal ADBNs, considering innocuousness properties is, therefore, enumerated as follows.

Proposition 4.2 (Number of indirect-free instantiations in diagonal ADBNs). *Given a diagonal, dense inter-timeslice (A)DBN $(B_0, B_\rightarrow)$, $N = |\vec{X}^t|$ and Q innocuous states, the number of indirect-free instantiations $\mathcal{N}^{/}{}_{N,Q}$ of random variables $(\vec{X}^t, \vec{\mathcal{A}}^t)$ in a timestep t is*

$$\mathcal{N}^{/}{}_{N,Q} = 2^{N-Q} \cdot \sum_{k=0}^{Q} \sum_{n=0}^{N-k} 2^{k(N+n-1)} \cdot A001831'_{N-k,n} \cdot \binom{Q}{k} .$$

$A001831'_{N,n}$ represents the number of uniformly directed bipartite graphs with groups of size n, m, $N = n + m$. This means that each node of the group with m nodes (black) is connected to at least one node of the group with n nodes (white). A black node is not connected to a black node. A white node is not connected to a white node and can be unconnected. All nodes are labeled.

$$A001831'_{N,n} = \binom{N}{n} \cdot (2^n - 1)^{N-n} .$$

$\mathcal{N}^{O}{}_{N,Q}$ origins from the consideration that one has between $k = 0$ to $k = Q$ "deactive" innocuous nodes. All non-innocuous state variables are freely instantiable, i.e., 2^{N-Q} instantiations. With k "deactive" innocuous nodes, there exist $N - k$ active nodes whose activators must be instantiated such that no two activators possibly interlock. The number of activator instantiations between these $N - k$ active nodes then corresponds to the number of uniformly directed bipartite graphs $A001831'_{N,n}$, arranged in all possible group sizes from $n = 1$ to $N - k$. In every arrangement, all $k(k-1)$ activators between deactive nodes are freely instantiable, all $2 \cdot k \cdot n$ activators between deactive nodes and nodes of the first group (and vice versa) are freely instantiable, and $k \cdot m$ activators regarding a dependenace of the deactive nodes on *nodes of the second group are freely instantiable, i.e., $2^{k(k-1)+2kn+m} = 2^{k(k-1)+2kn+k(N-k-n)} = 2^{k(N+n-1)}$ instantiations.* ▲

Figure 4.1 shows a comparison of $\mathcal{N}^{O}{}_{N,Q}$ and $\mathcal{N}^{/}{}_{N,Q}$ for $0 < N \leq 25$ and different Q. Note that even in a logarithmic plot, a cyclic ADBN has an exponential advantage in favor of a classic diagonal (A)DBN.

Independencies in Bayesian networks and graphical models in general have been extensively studied for efficient inference, notably by Zhang and Poole (1996) exploiting causal independencies. They extend their work on exploiting independencies for more efficient inference with Boutilier et al.'s (1996) contextual independencies in Poole and Zhang (2003). Still, a contextual independence where a context itself becomes independent was not considered in these works, and this hampers ways of more efficient reasoning and representations of causalities. To be precise, so far independencies in (dynamic) Bayesian network have almost always only been considered for their easier parametrization abilities, as only CPDs instead of complete JPDs must be specified, and for exploitations towards more efficient inference, but independencies in Bayesian networks were not considered for their increase of expressiveness, as done by us in this and previous chapters. However, the expressiveness and semantics of local CPDs have been well studied, often evolving around combination functions that procedurally create CPDs.

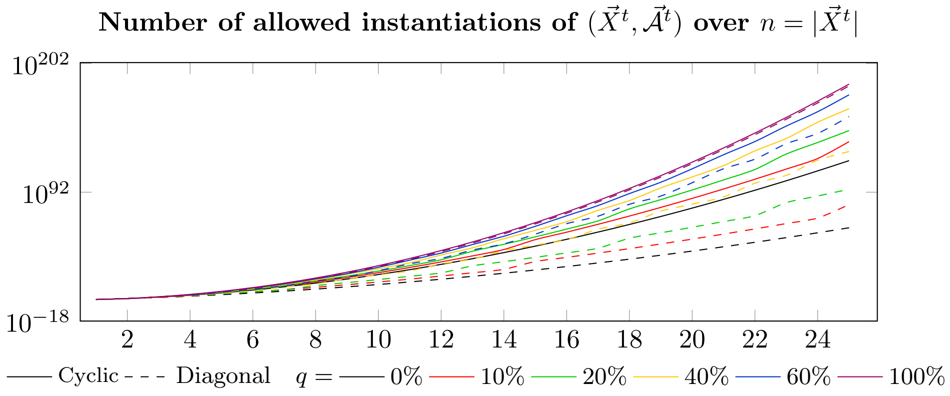

Figure 4.1. *Cyclic ADBNs ($\mathcal{N}^{\circlearrowleft}{}_{N,Q}$, solid) clearly outperform classic diagonal DBNs ($\mathcal{N}^{/}{}_{N,Q}$, dashed) in the number of allowed instantiations of $(\vec{X}^t, \vec{\mathcal{A}}^t)$. Note that for a full noisy-or network (100%) all possible graph structures ($\vec{\mathcal{A}}^t$ combinations) are allowed in the case of $\forall i \neg x_i$, which draws $\mathcal{N}^{/}{}_{N,N}$ near $\mathcal{N}^{\circlearrowleft}{}_{N,N}$. Still, even in this extreme case a cyclic ADBN outperforms a classic DBN by two orders of magnitude (semi-logarithmic plot).*

Often Boolean combination functions have undergone notable considerations in works by Henrion (1988), Srinivas (1993), and Antonucci (2011) introducing extensions to cope with imprecision. In Boolean combination functions, CPDs are procedurally created through individual probability fragments, associated with e.g., each arc and value of a dependence, and a (numerical) combination function. Each CPD entry is then created by numerically combining values of each arc associated with the value of a dependance. Often these values are seen in binary domains to be “true” and “false” in the sense of “present” and “absent.” Please note that these associations shall not be confused with activator properties. Further, one value, say “true,” might represent a present cause for one dependent, but an absent cause for another. To be precise, Heckerman and Breese (1996) describes a value of a variable that represents an absent cause as a “distinguished” state. Semantically, a distinguished state of a variable does represent an innocuousness property, but, as discussed throughout this chapter, it is not explicitly representable in a CPD. Still, a formal definition of innocuousness itself as “an inactive node does not cause any harm” is missing. Although, the counterpart “only an active node causes harm” is mentioned as an “amechanistic” property by Heckerman and Breese (1996). Heckerman and Breese (1996) and Zhang and Poole (1996) consider amechanistic properties in combination with explaining-away properties of Bayesian networks towards more efficient inference. They exploit that, once one active node is observed and is sufficient to cause harm, other potential causes are explained away and are less significant for any inference results. Still, amechanistic properties are not solely relevant for more efficient inference, as Zagorecki and Druzdzel (2006) show that amechanistic properties are beneficial for an easier parametrization of CPDs.

Cozman (2004) provides formal definitions and specifies properties of combination functions leading to an axiomization of the noisy-or function. Cozman (2004) formalizes an "accountability" property of combination functions, which is related to innocuousness properties to some degree. In his definition accountability is a property of CPDs that assures causes to origin from a modeled network, i.e., is the counterpart of "leaky" combination function, as, e.g., introduced by Henrion (1988). Namely, accountability formalizes that the probability of an effect, given all dependencies (causes) are in their distinguished state, is zero, e.g., $P(+x|\neg\vec{z}) = 0$. Still, $P(+x|\neg\vec{z}) = 0$ can semantically represent as well, e.g., that one *false*-dependence in $\neg\vec{z}$ "overrides" all other dependencies. This means that a CPD with $P(+x|\neg\vec{z}) = 0$ does not provide any semantic information about distinguished states of variables. Our formalization and definition of innocuousness(-contexts) explicitly identifies distinguished states and explicitly represents innocuousness properties in CPDs by specific parameter settings, neither requiring annotations nor novel representation formalisms.

4.4 Conclusion

In this chapter we have formalized an innocuousness property of random variables, which is often associated with Boolean combination functions for general CPDs. Based on a formalization with random variables taking the role of activators, we relax restrictions on ADBNs and give a quantitative evaluation of restrictions posed on such networks. Further, by providing a formal definition for innocuousness in CPDs, one gains the ability to formally represent that in specific contexts a dependency is causally irrelevant, opening new ways for more efficient inference and a higher causal accuracy in specifying CPDs.

Still, like in any other DBN, exact inference remains computationally intractable with respect to dimension complexity (number of state variables), and demands approximate inference techniques as introduced in Section 2.4. Considering that certain dependencies, i.e., arcs, are irrelevant in specific situations and a resulting BN might turn out to be singly connected, approximate inference techniques can heavily benefit from the newly defined, innocuousness properties.

While this chapter significantly increases the amount of allowed and supported instantiations per timestep, ADBNs still remain constrained. We have discussed that observations must enforce regularity, i.e., observations must be made under which only regular instantiations remain possible, i.e., only regular instantiations have a conditional probability larger 0. Note that Theorems 2.1 and 4.1 are based on instantiations, and observations must enforce that only regular instantiations remain a possible option. If observations are not sufficient to assure regularity, or are missing completely, an ADBN is not well-defined up to now. The following chapter addressed this problem by extending the theory of ADBNs towards models representing multiple structures and JPDs, where regularity is enforced intrinsically.

Chapter 5

Semantics of Bayesian Networks Revisited

For a large set of probabilistic graphical models, if not all, it is common to represent exactly one JPD, and, so far, situations, i.e., contexts from observations, have been discussed for ADBNs where as much partial structural information (deactive activator observations in $\vec{b}$) is evident to assure the same in an ADBN. Under every regular observation $\vec{b}$, exactly one topological ordering (under a common lexicographical ordering) of an equivalent (dynamic) Bayesian network exists, and this topological ordering holds for all instantiations conforming with evidence, as shown in Proof of Theorem 2.1. This means that inference is, so far, based on one well-defined (dynamic) Bayesian network defining one joint probability distribution. However, if sufficient structural information is not evident from observations, multiple structures are represented by an ADBN, each of which defining one joint probability distribution. This chapter shows that by moving away from a dogma that a (D)BN represents solely one joint probability distribution, an ADBN can be seen as a full extension of (D)BNs towards multi full joint probability distribution representations. Consequently, we show that *cyclic ADBNs are well-defined even if no observations are provided.*

If, in a specific context, Theorem 2.1 (and extensions) cannot be fulfilled, then an effective single structure for an equivalent DBN is not known, and an ADBN represents multiple joint probability distributions for every remaining, well-defined constellation of activator instantiations in the joint probability distribution. Note that no external constraints or conditions are given on the JPD by Proposition 2.4 and one closed-form formula represents all possible joint probability distributions of DBNs without a need of a case-by-case consideration. In this chapter, we revisit (A)DBN semantics s.t. not one single joint probability distribution is represented, but *multiple*. Note that this is significantly different from considering every case separately, namely every single structure separately. By Proposition 2.4 one formula encapsulates all full joint distributions and, consecutively, every well-defined sub-JPD (and therefore, sub-BN) is directly embedded in the global semantics of a dense intra-timeslice ADBN as a subset of entries in the full joint probability distribution.

The contribution of this chapter can be summarized as follows: By considering specific modeling approaches and a constraint of local CPDs, this chapter shows that ADBNs form (dynamic) probabilistic graphical models where partial structural information can be missing completely. In effect, we prove that ADBNs are well-defined DPGMs representing multiple structures as one, and, for the very first time, are a representation of multiple joint probability distributions. This means that one is able to represent joint probability distributions of joint probability distributions.

This chapter is structured as follows: Section 5.1 revisits the semantics of Bayesian networks and defines parameter settings for ADBNs in which no structural information need be supplied. As Bayesian networks are a special case of dynamic Bayesian networks (and vice versa), theory of ADBNs is extended towards classic Bayesian networks in Section 5.2, where the *Cry*-*Joke*-domain example is continued. We discuss obtained results and related work in Section 5.3 and conclude in Section 5.4.

5.1 Bayesian Networks of Bayesian Networks

In a situation where sufficient structural information is not available, multiple, regular and non-regular instantiations conform with evidence. Under the assumption that every timestep is regular, one knows that one of all remaining, possible regular instantiations must be effective, over which one is able to specify a prior belief. This means that, there exists a prior random distribution over all possible regular instantiations. Therefore, ADBNs can be extended towards extended ADBNs (eADBNs) in which no structural information need be evident from data or observations. In eADBNs, multiple topological orderings (even under a common lexicographic ordering) exist, each of them belonging to one well-defined Bayesian network. Then, an eADBN represents a Bayesian network of Bayesian networks as depicted in Figure 5.1, or rather, represents a distribution over multiple well-defined joint probability distributions. An eADBN shares familiar syntax and semantics with classic Bayesian networks, without case-by-case analyses of possible structures or introduction of external constraints, and an eADBN is defined as follows.

Definition 5.1 (Extended ADBNs, eADBNs)**.** *An eADBN is defined as a tuple* $(B_0, B_\rightarrow)$ *according to an ADBN (Definition 2.3). Additionally, random variables of a timestep* t*, i.e.,* $\vec{X}^t, A^t$*, are seen as influenced by one (meta) random variable* Ord^t*. Let each value of* Ord^t *represent a possible obtainable topological ordering* $ord_i^t \in \operatorname{dom}(Ord^t)$ *of random variables at time* t *under a common lexicographical ordering. Every topological ordering of random variables is obtainable by one minimal set of deactive activator instantiations (Proposition 2.3). Let* $\vec{a}^t_{\blacklozenge i}$ *be the minimal set of deactive activator instantiations to obtain a topological ordering at time* t *represented by* ord_i^t*. Then, let every CPD of activators* $a^t \in \vec{a}^t_{\blacklozenge i}$ *allocate all probability mass at its deactive value, i.e.,* $\forall a^t \in \vec{a}^t_{\blacklozenge i} : P(\neg a^t | ord_i^t) = 1$. ▲

In an eADBN, additional random variables are identified (or introduced) to ADBNs and certain CPDs of activator follow special parameter settings. These special parameter settings enable one to perform inference in ADBNs without the need of minimal observation sets, as an eADBN is well-defined without constraints, as stated by the following theorem.

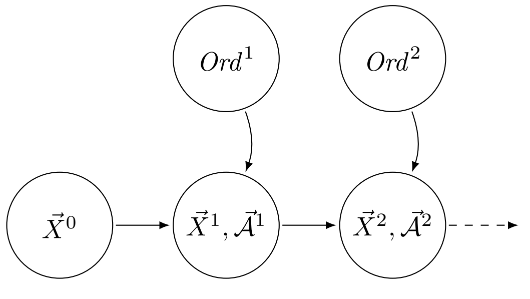
Ord^1
Ord^2
$\vec{X}^0$
$\vec{X}^1, \vec{\mathcal{A}}^1$
$\vec{X}^2, \vec{\mathcal{A}}^2$

Figure 5.1. *eADBNs are able to represent Bayesian network of Bayesian networks, i.e., represent a distribution of multiple well-defined probability distributions. Unlike ADBNs, eADBNs do not require any contextual, structural information. Specific parameter settings of $P(A^t_{XY}|Ord^t)$ preserve familiar Bayesian network syntax and semantics without requiring case-by-case analyses nor externally invoked frameworks.*

Theorem 5.1 (eADBN well-definedness)**.** *An eADBN is well-defined for every instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}, \vec{ord}^{1:t}$ of all random variables $\vec{X}^{1:t}, A^{1:t}, Ord^{1:t}$. Semantics as $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}, Ord^{1:t^\intercal})$ is well-defined and equivalent to DBN semantics given as the product of all locally defined CPDs.* ▲

Under Theorem 5.1 an eADBN is well-defined for all observation and instantiations, i.e., no minimal observation sets need be enforced. Therefore, one is able to reason about a structural identifying context itself, by using classic Bayesian network inference as, e.g., shown in Section 2.3. Note that, although cyclic dependencies are modeled and are not dissolved by activator contexts, no external constraints nor global normalization factors need be introduced, which still preserves the desired causal and local specifications, and the desired local interpretation of conditional probability distributions. Note further that not all possible structures need be considered separately (e.g., by an external framework), but a consideration of all possible structure constellations is intrinsically handled by the semantics of eADBNs, i.e., one joint probability distribution handles all structural "cases."

Proof of Theorem 5.1 (eADBN well-definedness). For *every* instantiation $ord_i^{1:t}$ of $Ord^{1:t}$ there exists an equivalent instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ of random variables $\vec{X}^{0:t}, A^{1:t}$ that is enforced by extreme allocations of probability masses in all local conditional probability distributions. For every equivalent instantiation $\vec{x}^{0:t}, \vec{a}^{1:t}$ there exists a topological ordering according to Theorem 2.1 and Proof of Theorem 2.1, and there exists a well-defined joint probability distribution $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})$. Every well-defined joint probability distribution $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})$ is seen as an entry $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|ord_i^{1:t})$ of a conditional probability distribution $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|Ord^{1:t})$. With a well-defined distribution over all possible orderings, i.e., $P(Ord^{1:t})$, a well-defined Bayesian network $\hat{B}$ is formed, defined by *two* random variables "$Ord^{1:t}$" and "$\vec{X}^{0:t}, A^{1:t}$," *one* prior probability distribution $P(Ord^{1:t})$ and *one* conditional probability distribution $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|Ord^{1:t})$ representing the joint probability distribution $P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}, \vec{Ord}^{1:t^\intercal})$ as the product of all locally defined CPDs, i.e.,

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}, Ord^{1:t^\intercal}) = P(Ord^{1:t}) \cdot P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|Ord^{1:t}) \ .$$

For an intra-timeslice dependency structure, $\hat{B}$ is seen as a dynamic Bayesian network $(B_0, B \rightarrow)$ with two random variables per timeslice as shown in Figure 5.1 with

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}, Ord^{1:t^\intercal}) = P(\vec{X}^{0:t-1^\intercal}, \vec{\mathcal{A}}^{1:t-1^\intercal}, Ord^{1:t-1^\intercal}) \\ \cdot P(Ord^t) \cdot P(\vec{X}^{t^\intercal}, \vec{\mathcal{A}}^{t^\intercal} | \vec{X}^{t-1^\intercal}, \vec{\mathcal{A}}^{t-1^\intercal}, Ord^t) \ ,$$

which is, for *all* instantiations, a well-defined DBN following Proposition 2.1. ■

Note that all derived approaches for exact inference and learning are not based on any well-definedness constraints and are solely based on the full joint probability distribution. This means that for inference or learning a well-definedness theorem is irrelevant and solely deals as a post-condition in order to be sure that results are correct. This has already been exploited in Chapter 4, where the introduction of innocuousness properties refined the well-definedness conditions, but all derived approaches remained applicable. The same holds for eADBNs, even if cyclic dependencies are not resolved by observations. From an inference perspective, the only addition is a new random variable Ord^t affecting some, or all, random variables of a timeslice without causing any cycles. Further, one could define activator random variables for every dependence of Ord^t and obtain a classical intra-timeslice ADBN, for which all derived approaches remain applicable.

The following example shows how the running example can be extended by a meta random variable, and then supports even an unobserved context of message transfer variables.

Example 5.1 (Regulatory compliance in eADBNs). *So far, at every timestep at least as much message transfers must be observed to assure regularity. Namely, at least $n^2/2$ message transfer variables M^t_{XY} had to be observed to be deactive to obtain a well-defined ADBN. With Definition 5.1 all queries are answerable without any contextual information about message transfers, and thus also fully supports predictive queries. We demonstrate the parametrization of* $\mathrm{dom}(Ord^t)$ *and $P(Ord^t)$ and a respective parametrization of activator random variables as an example on M^t_{CD}.*

In the running example with employees Claire, Don and Earl, one obtains six topological orderings under a common lexicographical ordering. Namely, one obtains $C \succ D \succ E, C \succ E \succ D, D \succ C \succ E, D \succ E \succ C, E \succ D \succ C, E \succ C \succ D$, which resemble the domain of Ord^t, abbreviated as $\mathrm{dom}(Ord^t) = \langle cde, ced, dce, dec, edc, ecd \rangle$. *For every value of* $\mathrm{dom}(Ord^t)$ *one is now able to specify a prior belief over every topological ordering as shown in Table 5.1. For example, if Claire is a supervisor of Don and Earl, topological orderings starting with C might be more likely, representing that it is more likely that the supervisor influences her employees and sends tainted messages to them. As stated by Proposition 2.3 every topological ordering can be induced by an instantiation of activator random variables, which are influenced by Ord^t in an eADBN. Considering M^t_{CD}, an adequate parameter setting of $P(M^t_{CD}|Ord^t)$ is given in Table 5.2, where the topological orderings $dce \in$* $\mathrm{dom}(Ord^t)$ *(i.e., $D \succ C \succ E$) and $edc \in$* $\mathrm{dom}(Ord^t)$ *(i.e., $E \succ D \succ C$) must enforce an allocation of all probability mass on $\neg m^t_{CD}$.*

Table 5.1. *Example for a distribution of $P(Ord^t)$ for the running example with arbitrary numbers α—γ with $\alpha + \beta + \chi + \delta + \varepsilon + \gamma = 1$. A numerical example instantiation is given, resembling that a topological ordering starting with C is more likely, e.g., that Claire is a supervisor of Don and Earl. Generation of* $\mathrm{dom}(Ord^t)$ *and $P(Ord^t)$ is straightforward and minimally invasive to the running example.*

Ord^t	$\mathbf{P}(Ord^t)$	Ord^t	$\mathbf{P}(Ord^t)$
cde	$\alpha = 0.3$	dec	$\delta = 0.1$
ced	$\beta = 0.3$	edc	$\varepsilon = 0.1$
dce	$\chi = 0.1$	ecd	$\gamma = 0.1$

Table 5.2. *Example for a CPD of $P(M^t_{CD}|Ord^t)$ for the running example with arbitrary numbers ι—μ. A numerical example instantiation is given based on an extension of the previous parameter $P(+m^t_{CD}) = \eta = 0.5$ and shows that a generation of $P(+m^t_{CD}|Ord^t)$ is straightforward and minimally invasive to the running example.*

Ord^t	$\mathbf{P}(+m^t_{CD}\|Ord^t)$	Ord^t	$\mathbf{P}(+m^t_{CD}\|Ord^t)$
cde	$\iota = \eta$	dec	$\lambda = \eta$
ced	$\kappa = \eta$	edc	0
dce	0	ecd	$\mu = \eta$

As we have demonstrated in this thesis, (cyclic) ADBNs arise naturally from domains by considering direct dependencies from a local point of view. Not always a meta variable Ord^t is immediately part of a domain, and Ord^t must be introduced to assure regularity even if sufficient structural information is not evident from observations. Still, the example shows that an introduction is straightforward, and is minimally invasive, as only CPDs of activator random variables must be modified in a simple schema. If intended, an extension of an ADBN towards an eADBN can be implemented automatically and an Ord^t variable need not be made explicit, if an inference engine is able to distinguish cases of insufficient structural information. Nevertheless, making Ord^t explicit has the benefit that a sufficiency check of regularity is not required and any Bayesian inference engine is intrinsically able to handle even a cyclic ADBN.

In an extended ADBN, activator random variables need not necessarily exist as independent random variables, as a topological ordering identifying random variable is sufficient. Still, the following remark shows that it is desirable to maintain individual activator random variables.

Remark 5.1 (Collapsed activator random variables)**.** *If a topological ordering identifying random variable (Ord) is present in a domain, activator random variables A^{st}_{ij} can collapse with Ord^t, i.e., Ord^t represent each and every A^{st}_{ij}. While this leads to a well-defined A(D)BN, it introduces a significant modeling overhead, as not all activator random variables are relevant for one X^t_i, i.e., large subsets of* $\mathrm{dom}(Ord^t)$ *identify the same parameter in a local CPD of X^t_i. Only random variables included in the vectors $\vec{A}^{st}_i, s \leq t$ are relevant for X^t_i.*

For example, in a dense intra-timeslice ADBN with n *state variables, one obtains* $n!$ *topological orderings of state variables. If activator random variables are collapsed with* Ord^t*, then every CPD specification of a state variable* X_i^t *must consider* $n!$ *cases of* $\mathrm{dom}(Ord^t)$*. With explicit activator random variables, every CPD specification of a state variable* X_i^t *must solely consider* 2^{n-1} *cases of* $\mathrm{dom}(\vec{A}_i^t)$*.*

Still, for one cyclic dependency, i.e., two potential topological orderings and two activator random variables, no overhead is introduced as shown in the following section as a continuation of the introductory Example 1.5 regarding the *Joke*-*Cry*-domain. As mentioned in the beginning, the *Joke*-*Cry*-domain, in fact, does represent an eA(D)BN, and is continued in the following section.

5.2 Extended Activator Bayesian Networks

We motivate the identification and exploitation of activator random variables in *dynamic* BNs, as they are often associated with cyclic dependencies and—from another perspective—for reasoning over time. Of course, also static "timeless" Bayesian networks benefit from activator conditions of random variables in order to model processes where certain dependencies change depending on other contexts. If roles of causes and effects are context-specific then cyclic dependencies are needed, which are permitted in activator Bayesian networks and defined as follows.

Definition 5.2 (Activator Bayesian networks, ABN)**.** *Unrolling the first two timeslices of a cyclic ADBN yields a classic BN with cyclic dependencies. Such a cyclic ABN represents a static Bayesian network in which local structures are not known in advance and dependent on specific contexts. This is beneficial for Bayesian networks in which roles of cause and effects are not uniquely identifiable, and in which dependencies are sensitive to a context. An ABN's semantics is well-defined by the unrolled ADBN's semantics as defined in Theorem 2.1 or in Theorem 5.1. For the case that a topological ordering identifying random variable is present in an ABN, i.e., an ABN's semantics is well-defined under Theorem 5.1, we speak of an extended ABN (eABN).* ▲

Context-specific influence directions are often found in human emotions caused by mutual interaction and are required to model a causally correct knowledge base for an artificial intelligence as the introductory Example 1.5 shows. In fact, the *Cry*-*Joke*-domain from Example 1.5 represents an example for an extended ADBN (Definition 5.1) as well as an eABN (Definition 5.2). The following example shortly repeats the domain and demonstrate the modeling approach as an eABN. For convenience, the referenced Figure 1.2 is given again in Figure 5.2.

Example 5.2 (Extended activator Bayesian network)**.** *To repeat, people might start crying from laughter because of hearing a (very) good joke. However, assuming only this causal direction at, e.g., a funeral, is a macabre assumption—the reasons why jokes are possibly told at a funeral are to possibly cheer up sad and crying people. As discussed in the caption of Figure 5.2 only modeling one causal direction is not an option.*

A cyclic dependency between both random variables Cry and Joke in Figure 5.2 is causally required, in order to capture a true causality between both random variables, whose direction is only identifiable in a context of another random variable Place. Considering Figure 5.2 as an extended eABN, Place represents meta-random variable Ord. In fact, in this example Ord is not a meta-variable, but is part of the domain: Seeing the domain of Place as $cry - worthy$ *places like a funeral or wedding, and usually happy places like a festival party or comedyclub, directly represents all possible topological orderings of the graph. Note that activators* $A_{Cry\,Joke}$ *and* $A_{Joke\,Cry}$ *are collapsed with Place in this example.*

By modeling the domain as an eABN, one includes all desired influences and dependencies as seen from a world-representation point of view: One can cry from happiness and from hearing very good jokes in the surroundings of happy, comfortable places, but people tell jokes to warm-up a chilling atmosphere among crying, sad people at darker places. Naturally, the cyclic eABN, as shown in Figure 5.2, also covers that at a happy place, a mood is usually happy, from which it is less likely to cry and it is more likely that a good joke was told. With an eABN all possible relationships can be included in one general model, and it is well-defined even if the context of Place is unobserved, i.e., no single causality is identifiable. As all potential models are intrinsically represented by one eABN, one is able to reason about Place—the structural identifying context—as well, and, effectively, inference is based on all possible models at the same time.

In this example, we represent two full joint probability distributions over which a distribution (P(Place)) is modeled. The semantics of this eADBN is a distribution based on the prior random distribution of Place over a full joint probability distribution over all random variables Place, Cry, Joke, Mood. Note that both represented full joint probability distributions are partially overlapping, as they represent full joint probability distributions over the same *random variables with common CPDs.*

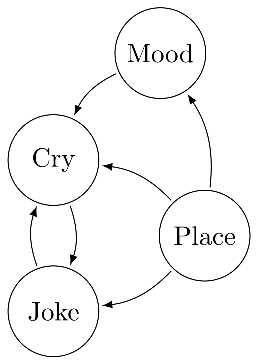

Figure 5.2. *Modeling causal dependencies often requires cyclic dependencies. If one is supposed to reason over Crying, potential influences* of *Joke or* on *Joke must be considered. Without a link between Joke and Cry, the only possible explanation for crying at a party is that the person is Mood = sad or a person always cries at parties—usually a wrong assumption. And without a link from Cry to Joke, the only explanation for a Joke being told at a funeral is the funeral itself—usually a quite macabre assumption. Repeated Figure 1.2 from Page 8.*

This example shows that not only activators exist naturally in domains, but that topological ordering identifying random variables occur from local, causal modeling approaches, as well. As shown in the previous chapters, the example represents a well-defined probabilistic graphical model, neither requiring external reasoning frameworks to handle every specific context of *Place* separately, nor requiring global normalization factors due to cyclic graph structures. We have shown that various algorithms for exact and approximate inference and learning of A(D)BNs follow familiar schemes and that the identification of activator random variables does not introduce any significant overhead. Further, this chapter shows that not even restrictions on instantiations or observations are enforced on A(D)BNs by adequate modeling techniques. Considering all of above, we feel confident to say that (dynamic) Bayesian networks can be based on cyclic graphs.

5.3 Discussion and Related Work

Extending ADBNs towards eADBNs is an interesting and tempting modeling approach, as no demands are made on observations. This is highly beneficial for situations where it is clear in advance that no sufficient observations will be available. However, we emphasize that an extended ADBN does constrain an ADBN as well, as an eADBN hardcodes the set of regular observations in the form of Ord^t and every other observation is deemed impossible, i.e., has probability zero. As we have shown in Chapter 4, constraints of well-definedness by Theorem 2.1 are relaxed by Theorem 4.1, *without* any modification to the running example at all. Chapter 4 has shown that without any other algorithms or constraints, observations that were deemed non-regular by Chapter 2 were, in fact, regular by considering innocuousness properties as done in Chapter 4. If one would have designed Ord^t and respective $P(A^t_{XY}|Ord^t)$ for Theorem 2.1, the complete model had to be revised based on Theorem 4.1. In summary, we emphasize that if observations can be trusted in advance to be regular, inference can be performed without the need of any regularity constraints, checks or modeling approaches. In fact, apparently non-regular observations might prove to be regular at a later point (after acquiring more observations, or, as discussed below, at a later timepoint t of the model), and previously obtained results will be well-defined without the need of any recalculation.

As an example for future work on regularity and well-definedness theorems for ADBNs, consider that even Theorem 4.1 can be relaxed further by including *persistency*, i.e., $P({+x}^t|{+x}^{t-1}, \vec{z}) = 1$. Given persistence, an observation of, say, ${+x}^{10}$ may let an activator A^1_{XY} at time 1 stand in its innocuousness-context ${+x}^1$ leading to a regular observation at time 1. In this situation, filtering distributions between time 1 and 10 may be spurious, but become well-defined again after timeslice 10. Respectively, smoothing distributions considering evidence beyond timeslice 10 will lead to well-defined results for timeslices 1 to 10. Note that, this situation is intrinsically handled by an ADBN, but would be cumbersome to encode in an eADBN for dependances of $Ord^{1:t}$ in respective CPDs.

Prior to the contribution of this chapter, in an ADBN, a sufficient amount of structural information had to be evident at every timestep to assure regularity. This means that one could have mimicked ADBNs by an external reasoning framework that analyzes each timestep and, procedurally, creates a required (dynamic) Bayesian network. While such an external reasoning framework would have a large overhead (one model is required for every possible context situation) and contradicts Pearl's intention of promoting Bayesian networks as a first-class world representation, sound results would have been obtained. However, not any external structure-analyzing framework can comprehend an eADBN, as no single model becomes evident. One eADBN represents multiple models and multiple full joint probability distributions, and inference is based simultaneously on all models. Note that this is significantly difficult to implement in an external framework, as "static" edges between models represent the *same* conditional probabilities that are not allowed to be counted twice. This means that inference results are not simply weighted for each model by the prior belief of each model, but the full joint over all random variables including the prior belief over each model must be considered, which is intrinsically embedded in an eADBN.

Moreover, by using eADBNs, one is able to represent every non-stationary DBNs (Robinson & Hartemink, 2008) or time-varying DBN (Song et al., 2009). Robinson and Hartemink (2008) and Song et al. (2009) take a fundamentally different perspective on changing structures, where changing structures represent an evolutionary change of a structure over time during a long evolutionary process and structures are slowly modified by a set of possible actions. As discussed earlier, this is different from a view taken by ADBNs: an ADBN changes rapidly over time, namely at *every* timestep depending on a specific context. In fact, ADBNs succeed non-stationary DBNs in their expressiveness by capturing all possible structures in one model, where a context specifies a needed substructure. By discretizing an evolution of a structure in a non-stationary DBN into discrete epochs, where during one epoch a structure remains constant, i.e., every modification action of a structure represents a different epoch, one obtains a novel domain random variable $Epoch^t$. Notwithstanding, $Epoch^t$ can dependent on $Epoch^{t-1}$ to represent an evolutionary character. Then, $Epoch^t$ represents an Ord^t, as well as, all activator random variables A^t as one. This means that all activators are collapsed, and $Epoch^t$ directly represents a context of a required structure during each epoch. With an obtained eADBN even inference in an unknown epoch is possible, and a non-stationary ADBN is represented in classic, familiar DBN syntax and semantics.

This chapter revisits semantics by considering all possible semantics implied by a model, i.e., all represented joint probability distributions at once. An alternative would be, in the case of insufficient observations, to find a minimally invasive modification of observations to assure regularity under a given theorem. While a modification of an observation is discussable and can, at best, lead to approximate results, not considering all possible $n!$ topological orderings, is significantly more efficient. Moreover, a modification of observations helps handling non-regular observations to a limited extend, as discussed in Chapter 7.

As discussed earlier, Ord^t represents a prior belief over potential structural candidates and is directly related to classical structural learning. Heuristics employed, e.g., by Friedman et al. (1998), to optimize and find potential structural candidates punish overly complex structures. Such a heuristic, for itself, immediately represents a form of belief over the set of potential structural candidates, as well. Therefore, learning eADBNs directly includes prior beliefs over network structures intrinsically in the model in one formalism. In fact, incorporating beliefs over network structures for learning Bayesian networks has been investigated by Friedman and Koller (2003), but in separate, external formalisms and models.

Acid and de Campos (2003) consider classes of equivalent graphs when searching a space of structural candidates, which is also incorporated by Friedman and Koller (2003). In fact, our definition of $Ord^{1:t}$ considers equivalent graphs intrinsically as well, as the domain, i.e., the space of potential structural candidates, is not dictated by the number of DAGs formable by the random variables, but by the number of topological orderings of the state variables. For example, under one topological ordering, say, $C \succ D \succ E$, eight different DAGs are representable, over which a prior belief according to the probability distribution $P(M^t_{CD}, M^t_{CE}, M^t_{DE}|CDE)$ exists.

To the best of our knowledge, an eADBN is the first probabilistic graphical model that is able to represent multiple joint probability distributions in one model. Further, to the best of our knowledge, an eADBN is the first probabilistic graphical model that is able to intrinsically represent a distribution over its own network structure and is able to base inference on all structural candidates at the same time.

5.4 Conclusion

This chapter extends the theory and discusses roles of regularity and well-definedness of ADBNs. We show that structural information need neither be evident nor be supplied by observations for eADBNs. This is beneficial for applications where one is not able to proactively assure that observations, and further, *regular* observations will be available.

Future work is dedicated to combine ADBN learning and eADBNs towards learning (dynamic) Bayesian networks under unknown and changing structures. As a long term goal, and without the need of heuristics and "external" optimization of structures, learning eADBNs can replace classic learning approaches with an intrinsic self-contained approach towards learning (dynamic) Bayesian networks under incomplete data and unknown structures.

In summary, so far, we have shown that ADBNs are DPGMs that allow a causal specification of dependencies while maintaining local semantics of CPDs, even if cyclic dependencies arise, which are only resolvable once a specific context is known. We have shown that their semantics as the JPD over all random variables is similarly defined to classic (D)BNs as the product of all locally defined CPDs and that algorithms with familiar schemes exist for finding exact, as well as, approximate solutions to common query answering problems. Further, learning ADBNs remains applicable, even from incomplete datasets where structural information is (partially) missing and this chapter extends theory towards models for which no structural information need be provided. Theory has been substantiated by a running example on taintedness of employees, for which we show in the following Chapter 6 that it is immediately applicable to a real-world problem in security, and that ADBN's local semantics and causal specifications solve a pertinacious problem in security analyses.

Chapter 6

Protecting Companies with (Dynamic) Bayesian Networks

Understanding and assessing potential impacts relevant for a company or mission is a pertinacious problem and a novel research area. For example, a local impact caused by an event on a distant node, say, a user workstation, might lead to a causal chain of operational failures, leading to an impact on a company, as some critical devices are affected required in critical business processes. Such assessments are frequently called *mission impact assessments.*

In order to perform a mission impact assessment, one must understand dependencies of a mission on various involved resources, and one requires an approach to "spread" and "propagate" locally created impacts onto higher goals, e.g., a space mission or a company's business goals. Early approaches attempting to solve mission impact assessment problems use ad-hoc methods involving newly established algorithms. We argue that such newly created algorithms suffer from multiple discrepancies, which we categorize into four different groups: **(1)** An expert must first fully understand and be trained in a system before he can assess configurations and parameters. We say, such systems do not provide a context-free assessment. **(2)** Obtained results from a system require a steep learning curve for interpretation and easily lead to overfitting by a dulling due to learned reference values. This means, results are not bias-free and require knowledge about a system. **(3)** During configurations, experts are forced outside their expertise, leading to potentially inaccurate specifications. We argue that it is favorable to accept disagreement from multiple knowledge sources instead of enforcing the definition of one allegedly congruent knowledge base. Finally, configurations (compare Problems 1 and 3) are assessed by a possibly overtrained expert and might be inaccurate, but parameters are not verifiable nor can be validated by an independent third party. This means that **(4)** obtained results from a newly created algorithm must be validated against a ground truth. Ground truths for occurred events and their exact impact on a mission are often not available in large quantities or are confidential.

To put things into perspective, a non-context-free system requires an expert to understand how an evaluation reacts to a parameter of "5" and how to a parameter of "3"—without the context of the complete framework, such values do not have any meaning and are neither verifiable, validatable, nor are understandable.

Further, an end-user becomes biased from interpretation of received results: With an unclear definition of an end-result, e.g., “yellow,” “3,” or “severe,” an end-user intrinsically adapts over time to “normal” results and becomes biased, i.e., a reported “severe” “red” error of category “5” is first taken serious, if it persists for an hour. These are the same emerging problems as outlined by Example 1.4.

In this chapter, we present an approach towards probabilistic mission impact assessment (probabilistic MIA) based on a probabilistic graphical model. We consider three views from three experts from different expertise and combine them inside a probabilistic graphical model. Local and global semantics of the defined probabilistic graphical models assure a context-free assessment of defined parameters, which are assessable and can be validated by themselves without knowing their later use. Based on the probabilistic models we reduce the novel problem of mission impact assessment onto a well-understood problem: probabilistic inference. Further, obtained results are understandable using common sense and do not suffer from biased interpretations. While the presented probabilistic MIA delivers great results in two real world use cases, it lacks the ability to consider persistence of impacts, e.g., the persistence of already caused harm to the surrounding even though the initial impact source has been eliminated. In the same sense as persistence is “added” to Bayesian networks by dynamic Bayesian networks, we extend probabilistic mission impact assessment to dynamic mission impact assessment in this chapter and show that cyclic ADBNs are immediately required.

The contribution of this chapter can be summarized as follows: By introducing a well-defined probabilistic graphical model for mission impact assessment, we are able to reduce impact assessment on a well-defined mathematical problem, which allows for a validation of results at data level and does not require deep training of experts. By resorting to local conditional probability distributions one is able to integrate widespread knowledge from different expertise into one sound model. This is useful for applications, where qualitative assessments are required and perpendicular views from multiple experts onto a problem must be brought inline. As a long-term goal, this provides the basis for an automated response system based on a mathematical well-defined model for risk and impact assessments in a predictive and retrospect analysis over time in changing and dynamic environments by the use of dense intra-timeslice ADBNs.

This chapter is an extension of an approach that has been presented at the NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact held in Istanbul, Turkey during June 2015 and at the Workshop on Cyber Defence and Security in Brussels, Belgium in September 2015. Valuable comments by participants and reviewers have been incorporated, and the approach has been well perceived by end-users and security experts. The underlying approach of this chapter, outlined in Section 6.1, is employed as part of the PANOPTESEC integrated research project (GA 610416) funded by the Seventh Framework Programme (FP7). In cooperation with the Panoptesec’s use case partner, we apply the presented approach to real data in Section 6.2.

The remainder of this chapter is structured as follows: Based on a well-defined probabilistic graphical model, we develop in Section 6.1 a mathematical model for mission impact modeling based on views from different experts. We introduce a notion of temporal aspects to cover dynamic environments to a certain degree and propose an independent-timeslice model assessing impacts at independent points in time, e.g., at independent short-, mid- and long-term time points. Based on this model, we discuss mission impact assessment as a formalized problem from probabilistic views in Section 6.2, apply the introduced independent-timeslice mission impact assessment in two real world use cases involving business-, IT-, and security experts from different domains and show that the approach delivers satisfying and greatly accepted results.

We extend our work towards dynamic mission impact assessment for realtime and forensic analyses in Section 6.3 by an extension of the presented independent-timeslice model towards completely dynamic probabilistic mission impact assessments for rapidly changing environments and time-dependent analyses at a, if intended, nearly continuous time granularity. We show that this extension directly represents and demands a cyclic dense intra-timeslice ADBN closely resembling the taintedness domain. We discuss and propose various approaches for such an extension and show how derived independent-timeslice models can be reused directly in future work. In Section 6.4 we discuss related work and identify common discrepancies and benefits of existing approaches. We conclude in Section 6.5.

6.1 Dependencies and Impacts

In the following, we take a view from different perspectives towards mission impact assessment. We consider three views from three experts from different expertise. We do not enforce an expert to overlook assessments from other experts and expertises (local assessment), and further, do not require that an expert understands or is trained on how his assessment is used inside algorithms and frameworks (context-free assessment). Based on a probabilistic model, we are able to include all three views directly into one consistent model bridging semantic and technology gaps. Based on this model a well-understood probabilistic inference problem is evident that assesses a mission impact from widespread events towards a bias-free result.

Every expert defines a different dependency model, where every modeled entity represents a random variable and dependencies between entities are represented by local conditional probability distributions (CPDs). As discussed and demonstrated in earlier chapters, local semantics of CPDs enable one to perform a context-free assessment of parameters, i.e., dependencies.

Remark 6.1 (Impact)**.** *We use an abstract term of "impact" in our work in the sense of "not fully operating as intended." The underlying meaning of "intended operation" depends on the use-case of a model.*

6.1.1 Mission Dependency Model (Business View)

In order to perform a mission impact assessment, one must understand the good one aims to protect, e.g., a mission or a company. In the field of business intelligence, a complete company or organization, i.e., the good one aims to protect, is modeled as a conglomeration of *business processes.* Commonly, business processes are modeled using the business process modeling notation (BPMN) and a business process is modeled as a (dependent) collection of tasks. This modeling approach is well accepted and is as well used by, e.g., de Barros Barreto et al. (2013), Albanese et al. (2013), and Musman et al. (2010). Figure 6.1 shows a sketch of a BPMN model used throughout this chapter.

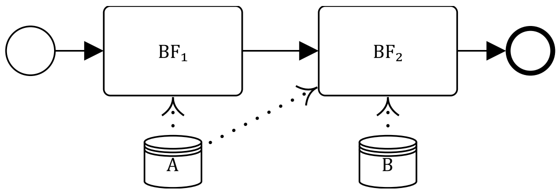

Figure 6.1. *Example BPMN 2.0 model sketch for the BP_1 business process shown in the dependency model of Figure 6.2.*

Designing such BPMN models is handled manually by an expert from a company, or by an external business consultant having a precise expertise in the understanding of business analysis. The business analysis is performed on a pure business perspective and stops at a first "resource" level. For example, from a business perspective, only a web-frontend may be identified, but not complex dependencies on databases, computation clusters and backup devices. In order to not force a business analyst outside his expertise, we intend to accept an inaccurate identification of resource-devices in the mission dependency model and focus on reflecting the information as directly described by an expert.

Therefore, we extend a model by Jakobson (Jakobson, 2011) and model mission dependencies as shown in Figure 6.2 as a graph of *mission nodes*. We model a *company* as being dependent on its *business processes*. A business process is again dependent on one or more *business functions*. *Business resources* provide business functions to some extent. Identified business resources are part of an infrastructure perspective and may not precisely reflect the operational relevance of resources. Continuing the previous example, an expert models a business function "provide access to customer data" and may identify a web-frontend as the directly mission critical device, i.e., business resource. Notwithstanding, the web-frontend is only a small part of providing the access to customer data, besides various central database servers, computation clusters analyzing customer behavior, backup devices and load balancing servers. Still, it cannot be expected that a business analyst with a complete view on business processes of a company can overlook these complex technical dependencies. Moreover, when consulting multiple experts their view might disagree which of those resources is in fact the most critical. Therefore, we introduce in the following section a resource dependency model, which is used to automatically analyze intra-dependencies and allows experts to identify any of those involved resources that seem to be relevant for them.

Similarly to BPMN models, mission dependency models, e.g., as shown in Figure 6.2, are modeled manually directly by experts, for which we describe a use-case study in Section 6.2.2. Manual workload remains low, as these global business dependency models are likely to remain static over long periods of time, whereas changes reflect themselves at a resource level, discussed in the following section. Moreover, comparing Figure 6.2 to the original BPMN model in Figure 6.1, relations between BPMN entities and mission nodes become evident: Each BPMN model represents one business process inside a company, where BPMN tasks represent business function. Likewise, BPMN tasks access data from data stores, representing immediately critical business resources. In consequence, mission dependency models can also be automatically extracted from BPMN models if such are already present for a company.

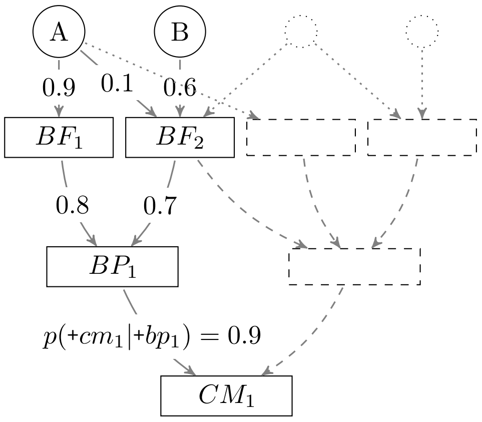

Figure 6.2. *Mission Dependency Model. Values along edges denote individual conditional probability fragments. For Example 6.1 only the solid entities are used. Consequences and further attributes are omitted in this figure.*

In a mission dependency model, every dependency, i.e., edge in Figure 6.2, represents a local conditional probability. Every conditional probability describes a probability of impact if a dependency is impacted. For example, the probability of the business-function BF_1 (see Figure 6.2 and 6.1), e.g., “provide access to customer data,” failing, given the required business-resource A, e.g., “customer-data-frontend,” fails is 90%. The local semantics of local conditional probabilities permit understanding parameters using common-sense (e.g., “in 9 out of 10 cases, customer data was not accessible for employees during frontend-server maintenance”). By that, all assessments of parameters can be directly validated (instead of holistic results) by an expert or through ground-truth. As these assessments are directly understandable by themselves and do not require reference values, an expert is able to take a local perspective on his assessments and must not overlook parameter assessments made by other experts in other parts of a mission dependency model, which may be outside his expertise. By the use of the conditional probabilities, every entity of a mission dependency model then represents a random variable X, where the event $+x$ represents the case that node X is operationally *impacted* and $\neg x$ represents that X is operating as intended, i.e., no impact is present.

Notwithstanding, a business entity may depended on multiple entities, e.g., a mission critical device may be a central database server or a node of a computational cluster. Formally, then, given a random variable X (e.g., a business process) that is dependent on multiple random variables $\vec{Z}$ (e.g., a set of business functions), the dependencies of X on all $Z \in \vec{Z}$ are represented by one conditional probability distribution (CPD) $P(X|\vec{Z})$ of node X. For ease of parametrization of these distributions we use commonly known combination functions, such as noisy-or and noisy-and. Hence, every node in a mission dependency model is a random variable, and edges define local CPDs associated to them, according to the following definition.

Definition 6.1 (From dependencies to distributions)**.** *We render every dependency of random variable Y on X as an* individual *conditional probability $p(+x|+y)$ and $p(+x|\neg y)$. Such individual conditional probability are* fragments *of* *a complete conditional probability distribution (CPD) and are therefore denoted in lowercase. To acquire the local CPD $P(X|\vec{Y})$ of node X from* all *its individual dependencies on nodes $Y \in \vec{Y}$, we employ noisy-and and noisy-or combination functions, as, e.g., described by Henrion in (Henrion, 1988).* ▲

By the use of combination functions, not complete conditional probability distributions must be parametrized, but solely single fragments of a distribution for every dependency, or edge, in a model, by which the number of parameters needed is significantly reduced. For the scope of this work, we consider non-leaky combination functions. Non-leakiness implies that a source for an impact must origin from inside a model and cannot occur "from nowhere," i.e., $P(+x|\neg\vec{y}) = 0$, and thus $p(+x|\neg y) = 0$ is fixed for every dependency. We believe that for most mission nodes, a suitable combination function is noisy-or, representing that every impact on a dependence might lead to an impact on the dependent node, i.e., every impacted dependence is a sufficient cause for an impact. If dependencies are laid out completely redundantly, a noisy-and combination function can be used for mission nodes to directly reflect redundancies. Notwithstanding, if an expert feels confident to do so, CPDs can be designed directly. Consequently, one obtains a formal definition of a mission dependency model as follows.

Definition 6.2 (Mission Dependency Model)**.** *A mission dependency model M is a directed acyclic graph (DAG) as a pair $\langle \vec{V}, \vec{E} \rangle$ of vertices $\vec{V}$ and edges $\vec{E}$. Vertices $\vec{V}$ are random variables (Notation 1.1) and are categorized according to their semantic as business-resources ($\vec{BR}$), -functions ($\vec{BF}$), -processes ($\vec{BP}$), and -company (BC). For the scope of this work, we consider that a business dependency model is created for a single BC. The ordering $BR \prec BF \prec BP \prec BC$ represents the strict topological ordering of graph M. Every edge $E \in \vec{E}$ represents a dependency. Let $V \in \vec{V}$, then let $\vec{E}_V \subseteq \vec{E}$ be the set of edges directed to V, and let $\vec{D}_V$ be the set of vertices from which $\vec{E}_V$ origin, i.e., $\vec{D}_V$ is the set of dependencies of V. For every vertex $V \in \vec{V}$ a conditional probability distribution (CPD) $P(V|\vec{D}_V)$ is given, or, alternatively, a combination function is given for V and edges $E \in \vec{E}_V$ are associated with conditional probability fragments s.t. a $p(+v|d)$ is given for all $d \in \mathrm{dom}(D), \forall D \in \vec{D}_V$.* ▲

Definition 6.2 solely considers inter-layer dependencies, and excludes intra-layer dependencies, e.g., we exclude dependencies of a business function onto another business function. We argue that such dependencies are resolvable in a lower level and by an adequate specification of associated CPDs. With Definition 6.2, a mission dependency model represents a probabilistic graphical model, and, in particular, a Bayesian network, as initially defined by Theorem 1.1 in Chapter 1, which represents a joint probability distribution over all entities in the dependency model plainly as the product of all local conditional probability distributions. The local semantics of Bayesian networks permit an expert to locally interpret individual parameters, i.e., to locally interpret individual probabilities of CPDs. We will preserve these local semantics for our presented probabilistic mission impact assessment, i.e., we preserve a direct and local understandability to all conditional probabilities, e.g., $p(+x|+y) = 67\%$. The following example shows the intention of using probabilistic dependency models and preserving the local semantics for mission impact assessments.

Example 6.1. *Following the mission dependency model depicted in Figure 6.2 (excluding dashed entities), a Bayesian network is evident representing a joint probability distribution (JPD) over all random variables as*

$$P(CM_1, BP_1, BF_1, BF_2, A, B) =$$
$$P(CM_1|BP_1) \cdot P(BP_1|BF_1, BF_2) \cdot P(BF_1|A) \cdot P(BF_2|A, B) \cdot P(A) \cdot P(B) \ ,$$

i.e., the product of all locally defined CPDs. $P(BP_1|BF_1, BF_2)$ *and* $P(BF_2|A, B)$ *are obtained by an employed combination function based on fragments* $p(+bp_1|+bf_1)$, $p(+bp_1|+bf_2)$ *and* $p(+bf_2|+a)$, $p(+bf_2|+b)$ *respectively. Local semantics of conditional probabilities, e.g.,* $P(BF_2|A, B)$ *and respectively* $p(+bf_2|+a)$ *are understandable* without *a need to consider the value of, say,* $p(+bp_1|+bf_1)$*.*

One obtains the probability of impact $+cm_1$ *onto the company* CM_1 *from, say, an observed impact on* $A = +a$ *and none on* $B = \neg b$ *by marginalization from the JPD as*

$$P(+cm_1|+a) = \alpha \cdot \sum_{BP_1} \sum_{BF_1} \sum_{BF_2} P(+cm_1, BP_1, BF_1, BF_2, +a, \neg b) \ ,$$

with a normalizing factor α *s.t.* $\sum_{CM_1} P(CM_1|+a) = 1$*. Later, we will define exactly this probability of impact onto a company as a well-defined mission impact assessment. An obtained result, e.g.,* $P(+cm_1|+a) = 20\%$ *is defined to be formally correct, given all CPDs are validated to be correct. This has the advantage that obtained results can be reported to higher authorities without disclosure or explanation of used algorithms or approaches.*

The example shows how a mission impact can be defined based on a probabilistic inference problem. An obtained result of, say, 20% is understandable without a context, i.e., one does not require indepth knowledge of an originating attack or needs to understand how this assessment is obtained. Further, a probability of 20% is interpretable unbiasedly, i.e., every person should come to a similar conclusion on how severe this assessment is.

The local semantics of mission dependency models, and the direct relationship to commonly known entities in BPMN models, intuitively allows business experts to model (partial) mission dependency models directly. When given information from multiple sources, e.g., multiple BPMN models and information gathered from different experts from different expertises, a common problem is eminent: Experts frequently use different nomenclatures, descriptions, languages and references for common entities, inevitably leading to semantic overlaps between information sources due to semantic and terminology gaps. Ergo, a naive concatenation of extracted and gathered business dependency model inevitably leads to duplicate entities and incorrect impact assessments. To obtain well-defined results, i.e., to obtain a solid and consistent business dependency model from multiple sources and experts, a semantic normalization and merging is required for business dependency model. Motzek, Geick, and Möller (2016) deeply discuss and propose a solution to the semantic normalization and merging problem of mission dependency models.

As mentioned above, a mission dependency model directly reflects expertise of business experts, and remains directly and locally understandable. However, an identification of resources might be too naive or operationally imprecise.

Moreover, multiple experts might disagree on the identification of critical devices as discussed previously. Therefore, the following section discusses a resource dependency model, which covers transitive and indirect dependencies to cover this inaccurate and disagreeing information.

6.1.2 Resource Dependency Model (Operation View)

Identified critical resources may be inaccurate, or rather, may be part of a complex dependency network involving various other resources, which are beyond the scope of knowledge of various experts. For example, a business expert might identify a frontend web-server for accessing critical data involved in a business process. While this does accurately represent a business perspective, the web-server may only be a small part of a complex process from an operational perspective: variously involved databases fed by computational clusters and interfaces are equally, or even more, important. However, deeply understanding all transitively involved resources exceeds the expertise of a business expert; in fact, it might even be unknown to an IT specialist. Nevertheless, all transitively, indirect and direct resources must be covered in order to provide an accurate mission impact assessment, which is why we propose to learn these dependencies automatically from data in the following.

Consequently, we introduce a resource dependency model covering dependencies between individual resources, which can be, e.g., individual ICT servers, ICS devices, software components or, in other use cases, manufacturing robots, suppliers, soldiers or vehicles. Further, a resource dependency model may consist of heterogeneous resources, e.g., may represent in one model intra-dependencies between employees, intra-dependencies between ICT devices, as well as inter-dependencies between employees and devices. We follow the same probabilistic approach as before, i.e., every dependency between two resources represents a local conditional probability of impact, given the dependence is impacted, as shown in Figure 6.3, and every resource represents a random variable. Thus, a resource dependency model is formally defined as follows.

Definition 6.3 (Resource Dependency Model)**.** *A resource dependency model R is a directed graph as a pair $\langle \vec{V}, \vec{E} \rangle$ of vertices $\vec{V}$ and edges $\vec{E}$. Every edge $E \in \vec{E}$, from vertex $X \in \vec{V}$ to $Y \in \vec{V}$ represents a dependency, and is associated with a conditional probability fragment $p(+y|+x)$. Vertices $\vec{V}$ are random variables and represent resources in an infrastructure, where a subset of vertices semantically correspond to vertices of a corresponding mission dependency model M. Let $V \in \vec{V}$, then let $\vec{E}_V \subseteq \vec{E}$ be the set of edges directed to V, and let $\vec{D}_V$ be the set of vertices from which $\vec{E}_V$ origin, i.e., $\vec{D}_V$ is the set of dependencies of V. For every vertex $V \in \vec{V}$ a conditional probability distribution (CPD) $P(V|\vec{D}_V)$ is defined by a non-leaky noisy-or combination of all conditional probability fragments of associated edges in $\vec{E}_V$. V is not contained in $\vec{D}_V$, i.e., a resource V is not dependent on itself.* ▲

The definition of a resource dependency model is similar to the definition of a mission dependency model (Definition 6.2), and does represent a probabilistic graphical model as well, but does not introduce constraints of acyclicity, i.e., a resource dependency model can contain cyclic dependencies. Hence, a resource dependency model is not a Bayesian network. We preserve desired local semantics of parameters, i.e., local conditional probabilities, by an exploitation of employed noisy-or combination functions, as described by Motzek and Möller (2017) and Motzek et al. (2015).

Using two individual models, one from a business perspective and one from an operational perspective, is highly beneficial, as knowledge from different experts is included directly and is used to accept potential disagreements: If one model had to be agreed on from both perspectives, the identified web-server would be disputed as it is not clear which resources are directly mission critical and to which depth all dependencies have to be covered. In the worst case, too many vaguely relevant resources would be identified, or, too few resources would be identified. By the use of two models, each perspective is correctly represented without a need to make any compromise.

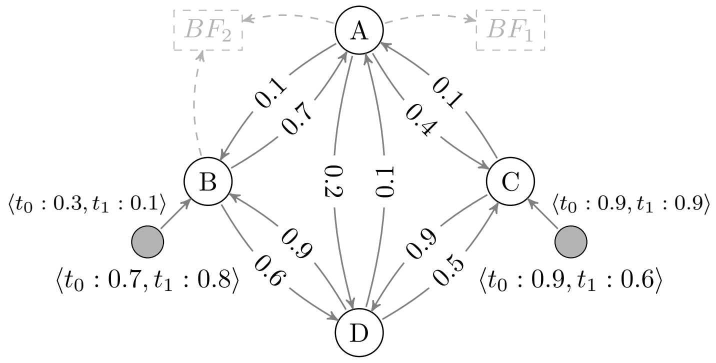
BF_2
A
BF_1
0.1
0.7
0.2
0.1
0.4
0.1
B
C
$\langle t_0 : 0.3, t_1 : 0.1\rangle$
$\langle t_0 : 0.9, t_1 : 0.9\rangle$
0.9
0.6
0.9
0.5
$\langle t_0 : 0.7, t_1 : 0.8\rangle$
$\langle t_0 : 0.9, t_1 : 0.6\rangle$
D

Figure 6.3. *Resource dependency model. Dependencies between B, C would also be possible. Conditional probability fragments are marked along the edges. Grey nodes represent external shock events leading to local impacts. The time-varying conditional probability of local impact given an instantiated external shock event is given next to the edge and the time-varying shock event's prior random probability is given below it. Connections to the mission dependency model are sketched in dashed gray.*

As mentioned earlier, and in contrast to the mission dependency model, assessing resource dependencies is not manageable by hand. Complex operation structures render a manual dependency analysis infeasible and error prone. Further, dynamically adjusting infrastructures (e.g., as found in IT-cloud use cases) make it even unknown to an expert to identify exact dependencies. However, we argue that an expert is able to validate a presented resource dependency model for plausibility. Therefore, heuristics are employed based on exchanged information amounts, e.g., traffic analyses in an IT use case, to learn possible resource dependencies. As long as a resource only consumes relevant information for its purpose, then every information transfer must motivate some dependency. Moreover, collecting traffic-information is a reasonable and feasible effort. Further, under the assumption of *per node* equally distributed entropy and encoding of consumed information, a dependency, i.e., a conditional probability, must be a function of consumed information bits. We, thus, reduce an infeasible effort for an expert of identifying all dependencies by hand onto finding a heuristic, or rather, validating of a generated dependency model. While a validation of a resource dependency model is expensive, it is a reasonable effort for highly critical infrastructures or operations. The following example demonstrates an approach for an automatic generation of a resource dependency model in an ICT use case.

Example 6.2. *It is reasonable to assume that it is feasible to acquire information about exchanged information at a logical ICT device level covering virtual machines as individual devices. Often, more granular data, e.g., on software layers, is not acquirable, and exact dependencies between software components are not identifiable. Still, multiple software applications running on one device are very likely dependent on each other, and an impact of one software component will lead to an impact of other software components. Ergo, we say that dependencies at network device level are coarse enough, and assume that every device drives one purpose that might be fulfilled by multiple software components.*

For example, a workstation X consuming different query results from multiple databases will distribute gained and processed information from such queries to other devices. The percentage of received traffic $T_{Y_i,X}$ from every database Y_i towards the total received traffic gives a good guideline for the conditional dependency between them as $p(+x|+y_i) = \frac{T_{Y_i,X}}{\sum_i T_{Y_i,X}}$. In general, this heuristic must apply to all cases where all dependencies of a resource provide similarly encoded data with similar entropy. An implementation of this heuristic in an ICT use case is directly provided by, e.g., Wireshark (Combs & The Wireshark Foundation, 2016). Wireshark allows one to capture raw network-traffic over long periods of time and, postponed, to analyze recordings, e.g., by their conversation statistics. Such conversation statistics directly provide the relative amounts of data transferred $T_{Y_i,X}$ from IP address Y_i towards X. Aggregating this data directly leads to the assessments for all conditional dependency assessments $p(+x|+y_i) = \frac{T_{Y_i,X}}{\sum_i T_{Y_i,X}}$ in a resource dependency model. In order to cope with dynamically changing IP addresses it is beneficial to utilize external network inventories providing exact matching between IP addresses and unique identifiers, or to use name resolution in a network. Moreover, it is highly beneficial to not record network-traffic payload, by which one significantly reduces memory requirements.

In Section 6.2 we show that the straightforward approach outlined in Example 6.2 delivers highly satisfying results for two real world use cases and discuss an implementation procedure. However, the heuristic presented by Example 6.2 may fail once a resource consumes irrelevant data, e.g., 5TB of cat pictures from a local file server. Depending on a network or company characteristics other heuristics might be appropriate, e.g., derivation from a mean received amount of data or a mapping onto a σ distribution.

In addition to modeled ICT-devices in Example 6.2, human resources may be modeled as well representing dependencies of employees on data access, manual data acquisition and complex dependencies between multiple employees. By analyzing access log files, and, if permitted, analyzing call and mail metadata, such dependencies may be automatically learned by using the same heuristic as in Example 6.2. Note that the presented mission impact assessment is not limited to ICT use cases, but as well suited for medical or military domains, where a resource dependency model may not represent ICT devices, but persons, patients, or vehicles, and mixtures between them. For example, a resource dependency model of a shipping company is likely to consist of various autonomous robots, multiple employees scheduling ships and berths, and employees scheduling suppliers and subcontractors.

The approach outlined in Example 6.2 generates one model for the complete period of observed network traffic, i.e., represents a general model of all evolution phases if a network changes over time. By periodically repeating the approach outlined in Example 6.2, a model incrementally adapts to changing, dynamic environments, and a differential analysis to a fixed time-period reference point (e.g., a monthly-generated model) can be used to cope with context drifts. In that sense, the here discussed model can still be used in slightly dynamic environments. Later in Section 6.3, we present an approach towards completely dynamic probabilistic mission impact assessments for rapidly changing environments and time-dependent analyses at a, if intended, nearly continuous time granularity.

Mission dependency models and resource dependency models directly represent an infrastructure and already define a probabilistic graphical model. What is yet missing to perform a mission impact assessment using this model is a source of potential impacts addressed in the following subsection.

6.1.3 Local Impacts (Security View)

A third view involves a security expert able to assess local consequences of events. In the style of reliability analyses using Bayesian approaches, as, e.g., early work by Torres-Toledano and Sucar (1998), we model external shock events inside a network. Informally, an external shock event (SE) represents a source for an impact and is attached to a node in a resource dependency model, i.e., a SE threatens a node to be impacted. By representing SEs as random variables, one gains the ability to include uncertainty about the existence of SEs and uncertainty about whether a present threat leads to an impact on a node. Formally, a shock event represents a random variable as well, and is defined as follows.

Definition 6.4 (External Shock Events). *An external shock event SE is a random variable and let $\vec{SE}$ be the set of all known external shock events. An external shock event $SE \in \vec{SE}$ might be present ($+se$) or not be present ($\neg se$), for which a prior random distribution $P(SE)$ is defined, i.e., SE is a prior random variable. Every vertex V of a resource dependency model R might be affected by one or more external shock events $\vec{SE}_V \subseteq \vec{SE}$. In case an external shock event is present ($SE = +se, SE \in \vec{SE}_V$), there exists a probability of it affecting node V, expressed as a local conditional probability fragment $p(+v|+se)$. If an external shock event exists and it is not inhibited, we speak of a* local impact *on V. In the case that the external shock event is not present, i.e., $\neg se$, it does not affect random variable V and we write $p(+v|\neg se) = 0$. Every individual conditional probability fragment from an external shock event is treated in the same noisy-or manner as a dependency towards another node, and thus, multiple shock events can affect one node and one shock event can affect multiple nodes.* ▲

According to Definition 6.4, the presence of an external shock event can be known (observed) or can be unclear and is assessed probabilistically through its prior random distribution $P(SE)$. We denote the set of observed external shock events (known presence) as a set of instantiations $\vec{se}_o$ of observed random variables $\vec{SE}_O \subseteq \vec{SE}$. This is highly beneficial for applications, where the actual presence of impact-sources is uncertain ($P(SE)$), and where evidence of existence and impacts is available, i.e., where SEs are observable ($+se \in \vec{se}_o$). Classically, a local impact can also be seen as an observation of an impacted node, i.e., $+x$.

However, in a probabilistic approach, a cause of an observation must be evident from and and be modeled inside the network. In consequence, the cause of an observation of an impacted node $+x$ cannot origin from an external source, and other network-nodes are "blamed" for the observed impact. By introducing external shock events one gains the ability to model "soft evidence" of local impacts, i.e., one is unsure whether an external shock event exists, and is unsure whether it might actually lead to a local-impact and affect a node's operational capability from external sources. Nevertheless, observations provide valuable information about transitive impacts and are further discussed after the following definitions and examples.

Assessing the existence of external shock events and the implications of present shock events is likely to not remain stationary over time. To address a degree of variance over time, we introduce the concept of temporal aspects of external shock events:

Definition 6.5 (Temporal Aspects)**.** *We define a temporal aspect of an external shock event. We employ the idea of abstract timeslices in which the effect of an external shock event changes. Every abstract timeslice then represents a duplicate of the network- and mission dependencies with a different set of local conditional probabilities and prior probabilities of local impacts. We denote time-varying probabilities in a sequence notation as $\langle t_0 : p_0, \ldots, t_T : p_T \rangle$, with $T+1$ abstract timeslices. In every abstract timeslice i, varying local impacts take their respective conditional or prior probability p_i defined for its timeslice t_i.* ▲

With Definition 6.5 an independent model is created for each timeslice, where impact assessments of time t_i are independent of assessments from time t_{i-1}. In the following, we call this the *independent-timeslice model.* This idea is extended in Section 6.3 towards a fully dynamic impact assessment, where entities of timeslice t_i *depend* on entities of timeslice t_{i-1}, i.e., a resource dependency model is a time-dependent model evolving over time with time-dependent, "conscious" nodes allowing for retrospect and predictive analysis of potential mission impacts. This extension will demands cyclic ADBNs as derived in previous chapters.

Every local impact represents a potential threat and can be, for example, a consequence of a present vulnerability, a countermeasure, a failure or an attack. It lies in the expertise of a security operator to assess a potential *local* impact of those threats. Due to locally viewed CPDs based on combination functions from probability fragments, an expert does neither need to have any expertise in resource dependencies nor an understanding of missions to do so. Further, an assessment of local impact probability can be formally validated through experiments or be grounded on common sense. The following examples demonstrate how to employ external shock events in an ICT security context and outline the merits of local assessments.

Example 6.3 (Response Plan Side Effects)**.** *We employ mission impact assessment to achieve a* qualitative *assessment of potential negative side effects of a proposed response plan to an ongoing or potential attack. We see a response plan as a collection of individual actions affecting a network. E.g., a shutdown of a server might easily reduce the surface of a potential attack. Still, if a critical resource is highly dependent on that server, it might impact a mission even more severely than a potential attack. We consider three mitigation-action types and transform them into external shock events, possibly leading to local impacts.*

The first mitigation action, i.e., an external shock event, is a ***shutdown****. Obviously, if a node is shut down (+se: the external shock event is present) it is easy to see that the probability of local impact, given the shutdown of node* X*, is* 1*, i.e.,* $p(+x|+se) = 1$.

Secondly, employing a ***patch*** *on a node* X *might produce collateral damage as well. During installation of the patch, there exists a (low) probability of immediate conflict, e.g., a flat assumption of* 10% *or a measure published by the software vendor. In a mean time, a patch might enforce a reboot of a network device. This leads to a temporal shutdown and might lead to hardware failure. Finally, after a successful reboot, a replacement of hardware, and/or a restore of a previous backup, the network device will fully resume its operational capability. Using temporal aspects, one is able to model a patching operation in three abstract timeslices and define the local impact probabilities of this external shock event to be* $p(+x|+se) = \langle t_0 : 0.1, t_1 : 1.0, t_2 : 0.0\rangle$.

Our third considered mitigation action is the restriction of a connection from node X *to node* Y*, i.e., a new* ***firewall*** *rule. From a technical perspective this operation forbids a transfer of data that might have been crucial for the operational capability of a node* Y*. Therefore, a firewall rule leads to an operational impact on* Y*. As a connection between two devices resembles a dependency, one must further actually remove this dependency. Otherwise, one would infer further impacts over a dependency that was prohibited and already assessed locally. To do so, one transforms a prohibited dependency to an observed external shock event +se s.t. the local conditional failure probability* $p(+y|+x)$ *becomes a local impact probability* $p(+y|+se)$*. Another approach, decidable by a security operator, would be to accumulate an amount of prohibited information flow for every node affected by a firewall rule, and to add one unified local impact for all prohibited connections per random variable.*

These external shock events are deliberately executed in the domain, i.e., their presence is fully known and controlled. Therefore, we model their prior probability to exist as a tautology, i.e., $p(+se) = 1$*, and observe the presence of mitigation actions, i.e., all modeled shock events* $\vec{SE}$ *represent the observed events* $\vec{se}_o$.

This example shows how executed actions are modeled as external shock events for an assessment and is applied in a real world use case outlined in Section 6.2.2. The assessments of *local* impacts are highly beneficial for the example, as not enough, if even any, data is available that allows for an analysis of potential impacts on a company related to executed individual mitigation actions. Without a context-free and bias-free assessment, one needs to generate data for learning and validation of an algorithm: Every mitigation action, and every mitigation action combination, must be executed on all network resources multiple times and an impact onto a company must be assessed. To obtain statistically sound results, we believe such tests must be executed, at least, several thousand times. Frankly, it is easy to imagine that a company would not exist anymore before experiments have finished. In our approach, only local assessments of mitigation actions are required which are validatable locally by using common sense or by using small local experiments, without a need to validate a global assessment. In our approach, if expert validation is not available or seen as not sufficient, for example, the impact probability of a patching operation on a resource can be validated by small local experiments on single, automatically deployed instances of virtual machines, which deploy and execute the patch and evaluate its outcome.

A second use case is motivated from an opposite perspective. While response plans discussed in Example 6.3 are intentionally executed actions on an environment, i.e., local impacts are triggered internally, the following use case considers impacts triggered by external sources, e.g., an adversary.

Example 6.4 (Vulnerability Impact Assessments)**.** *In a cyber security context, vulnerability advisories represent notifications of potential flaws in systems or software. It is not always known if a vulnerability is actually "exploitable," meaning, if a flaw can actually be exploited to cause harm to a system. Further, the expected amount of potential harm can be difficult to assess and depends on local configurations of components or further environment constraints. Moreover, inferring if a vulnerability is actually present on a local system requires a deep analysis of local software configurations. These facts motivate to model vulnerabilities as (partially observed) external shock events.*

A vulnerability represents one external shock event SE_V, which affects multiple nodes $\vec{X}$. The prior probability distribution of an external shock event SE_V, i.e., $P(SE_V)$, then represents (i) a probability of existence $P(SE_V^e)$, e.g., it affects all nodes running software Q, but it depends on numerous further uncheckable constraints if vulnerability V actually affects this configuration and represents (ii) the probability of exploitability $P(SE_V^x)$ if a (publicly known) exploit exists for a known vulnerability. $P(SE_V^x)$ is very likely to vary over time for which one can employ abstract timeslices. The prior probability distribution $P(SE_V)$ is then obtained by $P(SE_V)_t = P(SE_V^e) \cdot P(SE_V^x)_t$ and can be extracted from CVSS scores based on access complexity, authentication and access vector attributes, where access complexity is likely to vary, i.e., decrease, over time. Moreover, the prior random distribution $P(SE_V^e)$ of each external shock event provides the ability to include uncertainty whether an individual shock event does exist or not.

Given an exploitable presence of a vulnerability on a node X, a local impact might be created, i.e., $p(+x|+se_v)$. Likewise, this probability represents the expected harm of a successful exploitation of a vulnerability on a node and can vary for every individual node. A required probability fragment for an impact *$p(+x|+se_v)$ is obtainable by a noisy-or combination of the CVSS attributes for confidentiality-, integrity- and availability impact.*

All parameters, i.e., $P(SE_V^e)$, $P(SE_V^x)$ and $p(+x|+se_v)$, are qualitatively assessable and understandable for an expert, who can be assisted, or even be replaced, by an automatic extraction from public vulnerability advisories. Generally, abstract timeslices could be used to model a vulnerability in different dimensions, e.g., C, I, A. We refrain from this idea and keep the nomenclature of a general impact *on a node.*

Examples 6.4 and 6.3 demonstrate the use of external shock events to include knowledge about a resource's impact from external perspectives. Notwithstanding, similar knowledge is obtained from sensors inside a network, e.g., intrusion detection systems (IDS). Alerts obtained from, e.g., IDS represent internal information and are seen as *observations* of random variables, as discussed above. In order to directly represent this knowledge as external shock events, every raised alert is seen as one external shock event, with a prior random distribution $P(SE_A)$ representing the certainty of the raised alarm and $P(+x|+se_a)$ representing the severity, i.e., most likely 1, of the raised alarm.

As discussed above, a raised alert may provide transitive information about other resources as well, and we continue the discussion of observations and their implications in probabilistic mission impact assessments in Section 6.3.

Modeling vulnerabilities in a probabilistic model is significantly different from existing approaches to include vulnerability advisories to raise situational awareness, as, e.g., generating attack paths (cf. Jha, Sheyner, & Wing, 2002; Ou, Govindavajhala, & Appel, 2005). Attack paths try to address the problem how an attacker might actually compromise the network, i.e., such approaches try to simulate an attacker. In contrast, we intend to raise an amount of situational awareness that provokes a proactive removal of potential impact sources. In fact, examples like StuxNet (see, e.g., Langner, 2013) have shown that actual attack paths are only loosely based on an interaction of vulnerabilities, and that vulnerabilities rather represent first stages of attacks. Further, we argue that global effects of local vulnerabilities are hardly foreseeable by any expert. In our probabilistic model, only local consequences of exploited vulnerabilities must be addressed, and transitive effects are (automatically) assessed due to the resource dependency network. This means, one considers what all could happen *locally* and one does not try to find an actual path of an attack or somehow assess global effects at once. An actual use case demonstration for this example is given in Section 6.2 and we show that one obtains an assessment that is understandable directly and bias-free, namely "there exists a x% probability of compromise to our company" instead of "there exists an attack path over cve-xy, cve-zt, cve-ix, cve-po," which is only understandable to a security expert and whose implications are unclear for any non-security expert with indepth domain knowledge. The mathematical foundations for a context- and bias-free probabilistic mission impact assessment from the defined dependency models are introduced in the following sections.

6.2 Applied Probabilistic Mission Impact Assessment

In this section we discuss a probabilistic mission impact assessment from a mathematical perspective and apply it in two real world use cases related to cyber security.

Given a mission dependency model, a resource dependency model, and a set of external shock events, one obtains a probabilistic graphical model. Informally speaking, in the resource dependency model, some nodes are threatened by (observed) external shock events, and, as nodes are dependent, a threatened node might again threaten another node, leading to a "spread" or "propagation" of impacts. We say, a node is threatened by an external shock event *transitively.* In the end, there exists a probability that even a business process or the complete company (mission) is threatened transitively by various external shock events. To recall, to be threatened by an external shock event might lead to an impact; and it is a well-defined problem of calculating this "might"-probability of being impacted due to an external shock event, which is what we call the mission impact assessment. The following definition on mission impact assessment has already been applied in Example 6.1.

Definition 6.6 (Mission Impact Assessment, MIA)**.** *Given a mission dependency model M, a resource dependency model R and a set of external shock events $\vec{SE}$, a mission impact assessment of a mission node MN is defined as the conditional probability of a mission node $MN \in M$ being impacted (+mn), given all observed external shock events $\vec{se}_o$, i.e., $P(+mn|\vec{se}_o)$, where the effects of local impacts due to all $\vec{SE}$ are mapped globally based on mission-dependency and resource-dependency graphs. Note that $\vec{se}_o$ includes present (+se) and absent ($\neg$se) shock events and that some shock events are unobserved, i.e., are assessed probabilistically through their prior random distribution $P(SE)$. The task of obtaining $P(+mn|\vec{se}_o)$ is defined as the* MIA problem. ▲

Given Definition 6.6 it is the task of an mission impact assessment to solve the MIA problem, i.e., to obtain the probability $P(+mn|\vec{se}_o)$.

From a probabilistic point of view, a sound definition of an overall joint probability distribution as demonstrated in Example 6.1 is required. As demonstrated, this is given for the mission dependency graph, because it is a directed acyclic graph and represents a Bayesian network. However, in the resource dependency graph an acyclicity constraint cannot be assumed and Bayesian network semantics are not well-defined for cyclic graphs. One could transform a network dependency graph to a Markov random network, which, however, due to a needed global normalization factor, destroys an intended local view, i.e., local semantics, on probabilities. Motzek and Möller (2017) show that the MIA problem in the independent-timeslice probabilistic graphical model is solvable by an exploitation of noisy-or and noisy-and CPDs. Under noisy-and and noisy-or assumptions, the problem reduces to a commonly known problem of probabilistic satisfaction, as, e.g., used by de Raedt et al. (2007). Motzek and Möller (2017) employ a linearly scaling Monte Carlo simulation to obtain an approximate solution to the MIA problem and show that results are highly applicable to real world use-cases, which are given in the following subsection. In fact, the independent-timeslice received great acceptance at the NATO IST-128 Workshop on Cyber Attack Detection, Forensics and Attribution for Assessment of Mission Impact held in Istanbul, Turkey during June 2015, at the Workshop on Cyber Defence and Security in Brussels, Belgium in September 2015, and at the NATO IST-148 Symposium on Cyber Defence Situation Awareness, in Sofia, Bulgaria in October 2016, even by attendees from different fields of applications. Attendees were delighted that local CPDs and probability fragments allow an easy parametrization leading to directly understandable results.

In Examples 6.3 and 6.4 we discuss two application fields for the introduced probabilistic mission impact assessment, and give examples of automatically generated and learned resource dependency models, as envisioned in Example 6.2. In the following two subsections, we apply all examples to two real world use cases and demonstrate that the approach is directly applicable, delivers satisfying results and is greatly accepted by experts.

6.2.1 SMIA Challenge

In 2011 Teodor Sommestad et al. (Sommestad & Hunstad, 2013) conducted an experiment at the information warfare lab of the Swedish defense research agency, which gives us the opportunity to demonstrate Example 6.4 for vulnerability impact assessment. In the 2011 experiment, codenamed SMIA2011, a complete network consisting of multiple domains was set up containing multiple ICT servers, clients, mailservers, firewalls, ftps, webservers, even SCADA server, etc. User behavior was simulated inside each domain by action scripts, e.g., checking webservices, emails and downloading files. Intrusion detection systems provided information to one team in charge of monitoring the complete network and noting suspicious behavior. Another team was in charge of infiltrating the network. Both teams carefully documented their approaches and all network traffic was recorded over six days.

In summary and most noteworthy, the attacking team was able to ex-filtrate all mail messages from mail servers $m_1 - m_5$ and change parameters of a SCADA server *fanuc*. For our analysis, we consider these servers as mission critical resources, i.e., $\vec{BR} = \langle m_1, \ldots, m_5, \mathit{fanuc} \rangle$. While some attacks built on each other, the most impacting attacks were almost uncorrelated. Further, vulnerabilities actually played an insignificant role during attacks. As our approach does not build up on actual attack sequences, but rather considers vulnerabilities as points of interest, we evaluate if our approach is able to raise a significantly high enough situational awareness to be concerned about a compromise, i.e., impact, on the identified mission critical systems.

In more detail, the attacking team firstly discovered a misconfigured service running on one mailserver m_1, which allowed the attackers to extract a (encrypted and later decrypted) password file. By decrypting the password file from m_1 the attackers further gained a privileged ssh connection on m_1, which was exploited for tunneled attacks to two hosts f_6 and o_6 that were not reachable previously. Exploitation of a known vulnerability on f_6 and o_6 gave a remote shell that revealed further user-passwords. The extracted passwords allowed downloading all mailboxes from all domains, most likely due to reused passwords in multiple domains. Another vulnerability was exploited directly on another host h_a. While revealing new passwords, no further attacks built up on it. Nevertheless, it could have been an excellent starting point for the following and most interestingly attack: The attacking team was able to completely manipulate all employed firewalls, providing complete access to any node on any domain. Firewalls were only secured by a simple password (*"password"*), but it could have been extracted in one of the previous attacks. Due to the broadened reachability, the attackers had free access to a (otherwise completely unsecured) SCADA server, on which they successfully changed various parameters. Note that most attacks were not executed through any publicly-known software-vulnerability and, thus, would not have been detected by any analysis of such vulnerabilities alone. In fact, no publicly-known software-vulnerabilities were found on any mission critical resource, and no exploits were launched against them. Achilles' heel lied in a set of badly configured resources and a false sense of security in a presumably demilitarized zone. We believe that it is impossible to accurately identify and devise these chain of events beforehand and to detect these configuration flaws in any automatic or manual expert-driven approach.

In the SMIA2011 experiment all occurring traffic was recorded over multiple days and thankfully provided to us, which allows us to obtain a resource dependency model for this experiment, visualized in Figure 6.4. To learn a resource dependency model automatically, we follow Example 6.2 using wireshark and consider each IP as an individual resource, as dynamically assigned IPs remained static over the experiment. We solely utilize recorded traffic of the first day prior to any severe actions carried out by the attacking team (such as flooding the IDS and changing firewall configurations). An obtained dependency model seemed plausible, but dependency degrees showed up to be imbalanced: The amount of analyzed traffic of the first day was short and user scripts did not generate a realistic amount of traffic. Further, no operator was simulated to control or monitor the SCADA server, which is why it did not appear in the dependency model analysis. To overcome these circumstances, a minimal dependency probability of 5% is assumed and the SCADA server is manually modeled to be dependent on an operator from the same domain and vice versa. Note that these manual corrections are exactly foreseen in our dependency model. A domain expert is assisted by a heuristic delivering a locally interpretable model, which is, if needed, corrected and consecutively validated in his expertise.

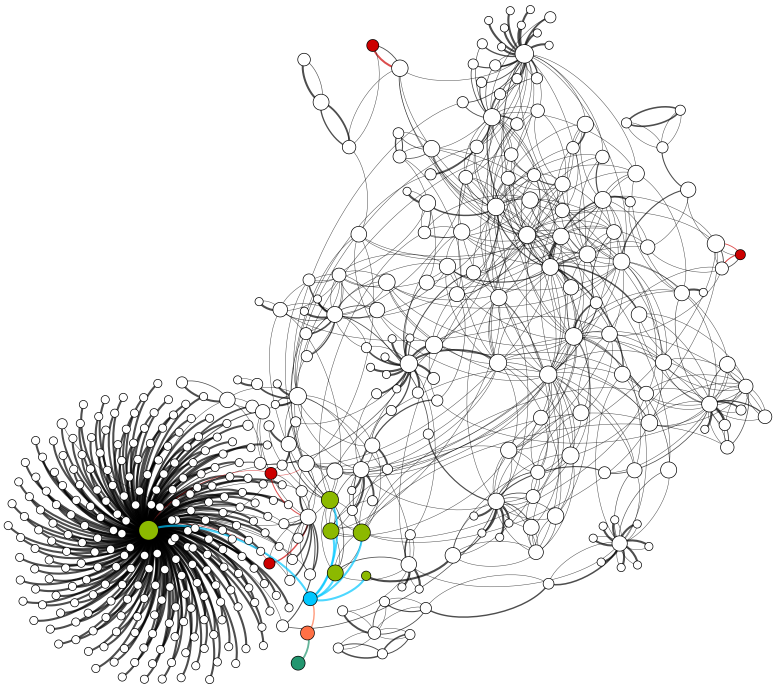

Figure 6.4. *Resource dependency model extracted from one day traffic captures for the SMIA challenge use case. Red nodes are directly impacted by vulnerabilities and affect other nodes transitively (first-step edges highlighted in red) and are remotely placed from mission critical devices (green). Thicker and darker edges represent higher dependency degrees. For visualization some insignificant dependencies are removed. In fact, the generated network is far from a fully meshed network (only 1% of all possible edges are extracted.) Number of nodes $n_N = 475$, number of edges $n_E = 2431$. Visualized using Gephi (Bastian et al., 2009).*

External shock events are modeled as described in Example 6.4. In total, attackers exploited three different vulnerabilities on four hosts (shown in red in Figure 6.4): *CVE-2010-0478* on some insignificant node h_k, *CVE-2008-4250* on h_a and *CVE-2003-0352* on f_6, o_6. These vulnerabilities represent the external shock events $\vec{SE}_V = \langle CVE_{478}, CVE_{4250}, CVE_{352} \rangle$. Three resources (h_a, o_6, f_6) of the compromised hosts are part of the domain containing m_1 and one (h_k) is part of the domain containing scada server $fanuc$. For all vulnerabilities, exploits are publicly known and integrated into various frameworks, e.g., metasploit, which is why we assume $P(+cve) = 1, \forall CVE \in \vec{SE}_V$ for all of them representing that one is certain that each vulnerability is present and exploitable. Local impact probabilities are adapted frankly according to their respective CVSS score divided by 10, e.g., $p(+f_6|+cve_{352}) = 0.75$. For simplicity we do not employ abstract time slices in this use case and no observations are made in this use case, i.e., $\vec{se}_o = \emptyset$.

Based on the defined local impacts, the resource dependency model and the mission dependency model, one obtains that there exists a probability of $P(+m_1) = 23.4\%$ of impact, i.e., compromise, of the firstly attacked mail server m_1 and that there exists a $P(+fanuc) = 7.8\%$ probability of compromise of the discussed SCADA server *fanuc*. Both servers were in fact compromised, but through ways unforeseeable by any software-vulnerability focused analysis. We argue that a probability in these ranges cannot be ignorable by any person confronted with these results. Even the other compromised mail servers, part of domains without present shock events, are assessed to be impacted with probabilities of $P(+m_2) = 8.1\%$, $P(+m_3) = 8.1\%$, $P(+m_4) = 10\%$ and $P(+m_5) = 7.6\%$. This shows that our MIA delivers reasonable and accurate results, as one obtains non-negligible impact probabilities raising one's situational awareness for all six servers that were compromised.

Considering all critical resources $\vec{BR}$ to provide, each, one business function $BF_i \in \vec{BF}$, which are equally part of one business process BP_1 of a, say, cloud company CM_1, an impact probability of $P(+cm_1) = 38.3\%$ on the company's mission is assessed. A probability in this range is not dismissable in any case and does not require reference results of previous impact assessments, e.g., it can be directly compared to tossing a ⚀ or ⚁ on a six-sided dice ⚅. Additionally, this mathematically grounded probability directly measures the criticality of this situation: with an objective measure of this cloud company, e.g., a monetary value expressing the value of the cloud company, one obtains an expected (monetary) loss for this situation originating from this probability of impact.

Moreover, our approach is completely transparent and understandable throughout from modeling over its usage up to obtained results. Every defined parameter can be grounded on expert assessments or historical evaluations and, finally, a produced assessment of "Due to a set of local, widespread impacts, there exists a probability of 38.3% that our mission will be compromised" is understandable and not dismissible. We argue that this assessment is eligible for reporting along a command-chain and understandable by every instance. In contrary, a non-bias- and non-context-free assessment of "vulnerabilities lead to a mission impact of 25764.324" does only make sense for a deeply trained expert or given reference results and is not suitable for any reports.

This example further shows the benefits of local assessments: Every parameter in the form of a conditional probability fragment is understandable and validatable by itself locally, and these parameters immediately define a well-defined probabilistic graphical model. By reducing mission impact assessment to a probabilistic inference problem in this model, obtained results are understandable without any reference values of other assessments, and are defined to be correct based on the correctness of the local parameters. While all parameters, considered for themselves, are understandable, the global implications of them are hardly overseeable. Still, the approximation algorithm derived by Motzek and Möller (2017) defined obtains these assessments in the range of seconds.

The following subsection discusses a further use case for a context-free mission impact assessment, utilizing observations and temporal aspects.

6.2.2 Acea Use Case

The Panoptesec integrated research project aims to “deliver a beyond-state-of-the-art prototype of a cyber defence decision support system” (Panoptesec DOW, 2013). As part of this project, we employ the derived probabilistic mission impact assessment for both use cases as outlined in Example 6.3 on response plan assessments and Example 6.4 for a vulnerability impact assessment. We are able to apply and test the complete approach in the Panoptesec’s use case partner, Acea SpA, Italy’s largest water services operator and one of the largest energy distribution companies in Italy (Acea SpA, 2016). To deploy a probabilistic mission impact assessment, all three models were created in cooperation with experts, as presented in the following.

In a team session of two business experts from the company and one IT specialist, a complete mission dependency model was identified for Acea ARETi in less than three hours. ARETi is a division of Acea SpA in charge of distributing and controlling energy to the city and vicinity of Rome. Admittedly, business experts showed to be reluctant to give assessments of CPDs, but intrinsically were able to understand all parameters. Given a set of choices, experts quickly agreed on an assessment and validated a complete mission dependency model, which is displayed in Figure 6.6 in its general structure in combination with the resource dependency model.

The most important objectives, i.e., business processes, of ARETi are to distribute and control high level voltage, whose dependency decompositions are shown in Figure 6.5. These business processes require four business functions provided by 16 mission critical resources. For distribution of high level voltage (business process BP_D), remote terminal units (RTUs) are required, which are remotely placed actors for switching power. Around 50 RTUs are geographically scattered in the vicinity of Rome and accessed via various communication links, e.g., by GSM. In consequence, these communication establishing devices (further called proxies) as well as the individual RTUs provide business function BF_R. Naturally, a high dependence between RTUs and their proxies exists, which is automatically learned and assessed in the resource dependency model (compare the two “clouds” of devices in the lower right of Figure 6.6). Hence, it is sufficient to solely identify involved proxies (i.e., central nodes of the clouds in Figure 6.6) in a mission dependency model, and not all individual RTUs need be identified. This is highly beneficial for this application, as individual RTUs are frequently replaced, which is automatically captured by an incremental re-learning of the resource dependency model and, thus, must not be made explicit in the mission dependency model. Two of these proxies (p_1, p_2) are optional and were assessed with a probability fragment of $p(+bf_r|+p_1) = p(+bf_r|+p_2) = 0.8$ each. Remaining proxies p_3, p_4, p_5 are required by business function BF_R with a probability fragment of $p(+bf_r|+p_3) = p(+bf_r|+p_4) = p(+bf_r|+p_5) = 0.9$ each.

The central intelligence between controlling (business process BP_C) and distributing power (BP_D) is provided by multiple SCADA servers (business function BF_S), which control and manage the individual RTUs, which are monitored by human machine interfaces (business function BF_H). Note that it is sufficient to solely identify the HMI clients (two in this case) in the HMI business function, as it is the directly critical device for the purpose of HMI; transitive dependencies of the HMI clients on SCADA servers, RTUs and proxies are automatically covered through the resource dependency model and probabilistic inference. As both HMI clients (h_1, h_2) allow almost full control over the complete process, both were assessed with a conditional probability fragment of $p(+bf_h|+h_1) = p(+bf_h|+h_2) = 0.9$. All SCADA servers of business function BF_S are laid out redundantly, which is replicated in the mission dependency model by using a noisy-and combination function. As controlling requires a working ICT environment, business experts identified an “ICT” business function BF_I consisting of one NTP-, three archival FTP-, and one web-server (business resources $\vec{BR_i}$). As these devices are not dramatically critical to provide this business function, their probability fragments were assessed each to $p(+bf_i|+br_i) = 0.4, \forall br_i \in \vec{BR_i}$. BF_I is only marginally required for controlling BP_C and was consequently assessed with a probability fragment of $p(+bp_c|+bf_i) = 5\%$. Dependency degrees for remaining business processes and functions, i.e., conditional probability fragments, were assessed as shown in Figure 6.5. Further details on these business processes, such as further redundancies and objectives of individual resources are omitted here for confidentially, and only shown in their general structure in Figure 6.6. In Figure 6.6 two other business processes are indicated with additional business functions, which are not discussed here for the same reason.

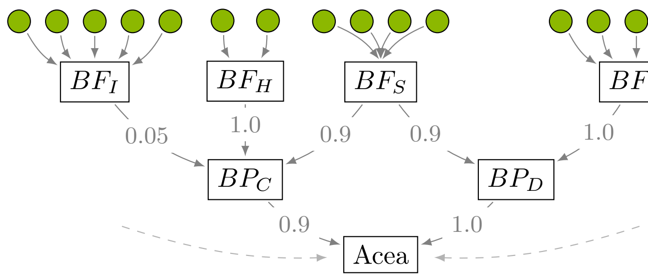

Figure 6.5. *Small exert of two business process involved in distributing (BP_D) and controlling (BP_C) high level voltage in the vicinity of Rome by ARETi and their four required business functions (BF_H, BF_S, BF_R, BF_I) provided by 16 mission critical devices (green). Business resources providing BF_S are laid out redundantly. Compare with the corresponding resource dependency model and full mission dependency model shown in Figure 6.6.*

Further, in the team session, problems outlined in Section 6.1.1 were evident, where different experts, in fact, used different nomenclatures and languages to refer to the same entities. For example, some experts referenced resources by hostnames, whereas others used IP addresses or some referenced mission nodes using abbreviations and others used literary descriptions. (Motzek et al., 2016) deeply discuss these challenges of an eminent semantic normalization and merging problem of mission dependency models.

A resource dependency model, depicted in Figure 6.6 is automatically learned from recorded traffic in a redundant, backup environment with emulated behavior of SCADA devices as outlined in the Example 6.2 every hour. A scheduled task iteratively collects traffic metadata using Wireshark for fifty minutes. These recordings exclude payloads and solely capture header information from Ethernet-

and IP-frames, such as MAC-address, IP-address, frame length and tcp/udp ports. Analyzing and aggregating this information has low and constant storage- and memory requirements, which allows one to constantly aggregate these information and generate a cumulative model over time. Based on these raw statistics, we periodically generate a resource dependency model as described in Example 6.2 every hour. By doing so, the model constantly adapts to changing environments within an hour. In fact, an external IT specialist consultant validated the extracted model to be reasonable for the company. However, we admit that not all individual probability assessments were validated, but critical dependencies were validated to be included and to bear a reasonable dependency degree. As discussed above, large amounts of RTUs are present requiring a remote connection. For this reason, dependencies on routing equipments are highly critical in this domain. In order to cover these routing resources, traffic is not solely analyzed on a logical level, but also on a physical level. To be precise, every connection is established between two logical devices, e.g., identified by two IPs, through two physical devices, e.g., identified by two MAC addresses. Through the use of a global inventory of all IT related resources, IPs and MACs are mapped towards unique identifiers and a dependence of each resource on its communication-establishing device is added. For example, say, a traffic recording includes a connection from MAC_1, IP_1 to MAC_2, IP_2. Say, IP_1 maps to ID_1, IP_2 to ID_2, MAC_1 to ID_3, and MAC_2 to ID_4; then a dependency of ID_2 on ID_1 is added, as well as a dependence of ID_1 on ID_3 and of ID_2 on ID_4, through a flat assumption of $p(+id_1|+id_3) = 0.9$. By doing so, one considers that a potential impact on a router may directly affect all resources communicating over aforesaid router, i.e., that communication may be spoofed, compromised, or prohibited.

In addition to a vulnerability assessment, response plan assessments are highly important for Acea SpA and the Panoptesec project. Assessing how responses to cyber-attacks affect an environment is a completely novel problem, no large datasets are available, and one must rely on a validation by an expert. The Panoptesec system proposes response plans automatically in reactive- and proactive-situations based on their effectiveness against attacks and financial benefits. However, these assessments and proposals are not necessarily in line with a company's goals. For example, a shutdown of highly critical node will certainly eliminate all attacks targeted towards that node and is financially highly attractive, as this response plan does not involve almost any cost. However, this response is catastrophic when considering implications on the mission, i.e., company, as clearly the mission cannot be accomplished anymore. To avoid sacrificing missions for a false security of security, the Panoptesec system employs the presented mission impact assessment using impact definitions by Example 6.3 to obtain *operational* impact assessments for each individual response plan. Mitigation actions $\vec{MA}$ in a response plan RP then represents an external shock event $\vec{SE}$, i.e., random variables with associated, time-varying conditional probabilities. All of these shock events are supposed to be deliberately executed. This means that their existence is known and observed, i.e., $\vec{se}_o = +\vec{ma}$. Then one obtains a three-dimensional assessment $P(+cm|+\vec{ma})$ for each response plan, i.e., the probability of operational impact onto the company if the proposed response plan is executed in a short-, mid-, and long-term time period.

Granadillo, Motzek, Garcia-Alfaro, and Debar (2016) demonstrate an approach to unify these three-dimensional assessments with additional multi-

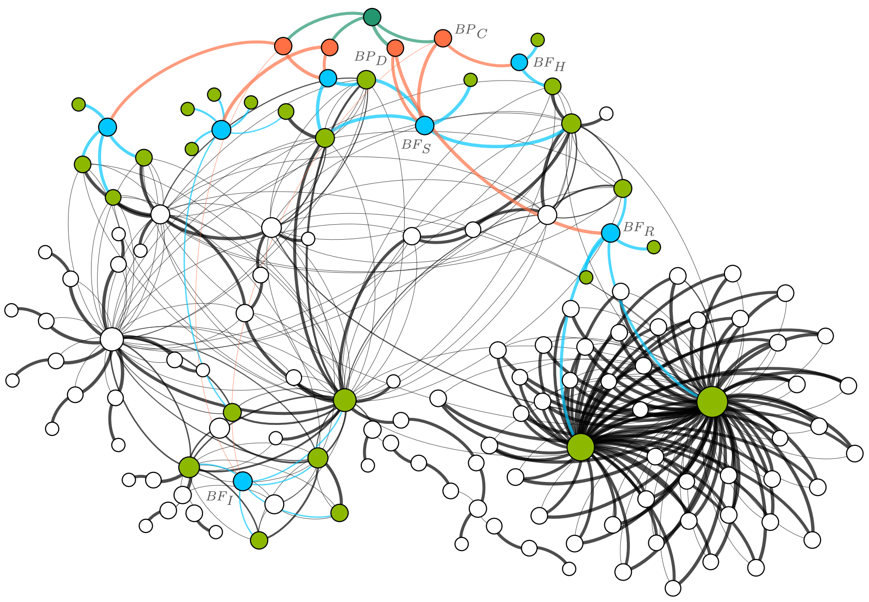

Figure 6.6. *Resource dependency model extracted from roughly one month of traffic captures in ARETi (represented in dark green), where related critical devices are highlighted in green, business functions in blue, and business processes in orange. This model was validated and verified to be reasonable by the company's IT experts.* $n_N = 344$, $n_E = 754$. *Visualized using Gephi (Bastian et al., 2009). Entities correspond to Figure 6.5 and are marked accordingly.*

dimensional assessments of response plans and propose a selection of optimal response plans based on an unweighted best compromise in all dimensions. Granadillo et al. (2016) evaluate the suitability of the presented mission impact assessment for operational impact assessment to obtain adequate responses to cyber-attacks in the here-discussed ARETi environment. In our expertise, we obtained that operational impact assessments of individual response plans bear reasonable assumptions, i.e., venturesome response plans were assessed to bear a high operational impact, and that a combination of impact assessment from financial and operational impact aspects delivers appropriate and non-trivial response plans. Clearly, both use cases are motivated from opposite perspectives, i.e., one from an adversarial perspective assessing probabilities of adversarial impact, and one from an operational perspective assessing probabilities of self-inflicted operational impact. Motzek and Möller (2016) discuss and evaluate the benefits of considering operational- as well as adversarial-impacts towards a well-defined probabilistic mission defense and assurance approach. Their approach is based on multi-dimensional optimizations and graph-theoretical problems in the probabilistic graphical model formed by Examples 6.3 and 6.4, an automatically learned resource dependency model, and an acquired mission dependency model.

Intermediate conclusion In summary, the presented independent-timeslice probabilistic mission impact assessment allows for causal, context-free, and validatable assessments of parameters and delivers context- and bias-free understandable assessments. Experiments on real data in two real world use cases show that the approach is well accepted, delivers satisfying results, and that all required models are directly obtainable. Moreover, Motzek and Möller (2017) experimentally and theoretically evaluate that the approach scales linearly even in large domains by the use of a verified approximation procedure.

Still, the introduced temporal aspects motivate that assessments require a consideration of *time*. To a limited degree, temporal aspects implement a consideration of time in the presented impact assessments, but we intend to consider the following situation as well: A node is threatened due to a set of vulnerabilities and is potentially impacted through them and dependent nodes become impacted transitively as well. For example, a hypothetical attacker is probably able to extract valuable information passively, able to manipulate processed data, or is able to gain further access through unforeseen events, to and from any dependent resource. If one now assures that the initial nodes are not impacted, e.g., by reinstalling all directly affected system, affections of other nodes do not immediately vanish as well. For example, all previously caused harm by the hypothetical attacker does not vanish magically, but does persist, i.e., any probability of inferred potential impact must persist to a certain degree. In the same sense as persistence is "added" to Bayesian networks by dynamic Bayesian networks, we extend probabilistic mission impact assessment to dynamic mission impact assessment in the following section. In fact, we show that an application of ADBNs is evident immediately and does closely represent the taintedness domain.

6.3 Extensions to Dynamic Mission Impact Assessment

Temporal aspects introduced in Definition 6.5 introduce a need for mission impact assessments over *time*. In rapidly changing environments, where dependencies of resources rapidly change over time, one requires a finely-granular time-sensitive evaluation of a mission impact. An extension of Bayesian networks (compare Section 6.1.1, a mission dependency model is a Bayesian network) towards dynamic domains considering evolutions of states over time is commonly known as a dynamic Bayesian network (DBN). In DBNs values of random variables depend not only on current influences, but also on their respective history. An extension towards a dynamic probabilistic graphical model for dynamic mission impact assessment (DMIA) proposed in this section allows for time-dependent impacts, e.g., decaying impacts, evolving mission impact analyses, and retrospective considerations of potential sources of impacts *inside* a network allowing for forensic and predictive analyses.

So far in this chapter, we outlined how mission dependency models, resource dependency models and impact models are obtainable, validatable and combineable towards one independent-timeslice probabilistic graphical model. Let R be a resource dependency model, then it is a straight-forward extension to introduce a dimension of time t into a time-dependent model representing a resource dependency model R^t for each timeslice t. In every R^t, each resource node $RN_i^t \in R^t$ is dependent on its predeccessor $RN_i^{t-1} \in R^{t-1}$ forming a dynamic probabilistic graphical model (DPGM) as shown in Figure 6.7. Respectively, at every timestep t a mission dependency model M^t is formed, whose business critical functions are dependent on some business critical resources in R^t. Likewise, at every timeslice some business resources are threatened directly by some observed or unobserved shock events in SE^t. However, as performed in Section 6.2, a well-defined semantics is required for such a DPGM, which, due to the (potentially) cyclic nature of a resource dependency model, are not immediately given by classic DBN semantics.

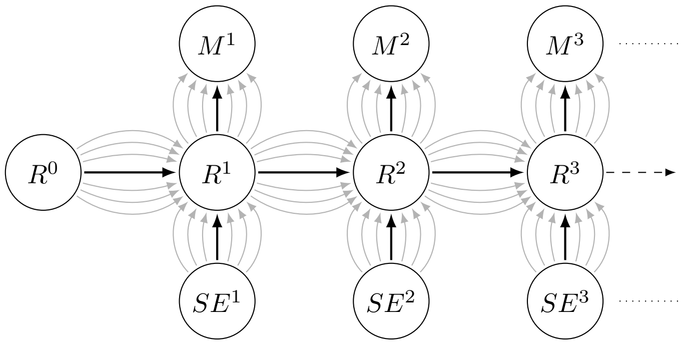

Figure 6.7. *An extension of the presented probabilistic graphical model for mission impact assessment towards a dynamic assessment, considering history states of (potentially impacted) nodes. This is beneficial for applications where interactions of nodes are rapidly changing over* time *and a live tracking of impacts is required. Resource nodes RN_i^t of a resource dependency model R^t are dependent on their predecessor $RN_i^{t-1} \in R^t$. R^0 represents an initial assumption about the potential impact state of nodes.*

A resource dependency model is derived from (automatic) analyses of communications between resource nodes. Considering such a dependency model over time directly allows modeling each communication at time t as a dependency, i.e., an influence at time t. This means, with every communication there exists a probability of impact. For example, every time a significant amount of data is transferred from an, say, integrity impacted node to another, there exists a probability that the other node becomes impacted as well, as it is the case for message transfers M_{XY}^t in the taintedness domain. The latter probability can, e.g., either be derived as described in Section 6.1.2 or be derived by experts as described for the mission dependency model.

Still, a high rate of communications between of nodes as, e.g., present in IT infrastructures, is computationally too expensive. Considering an IT infrastructure as an information processing chain, it is reasonable to aggregate communications over a reasonable timeframe suited to an use case. An aggregation significantly can reduce computational costs for obtaining inference results in DPGMs, but requires a careful consideration of indirect influences as we discussed in previous chapters. Under the assumption of an information processing chain it is reasonable to assume that during one timeframe no cyclic communication occurs, i.e., a feedback loop in information processing is not finished in one timeframe. Then, in fact, the presented dynamic mission impact assessment model represents an

ADBN as described by the previous chapters, closely resembling the taintedness domain, and one obtains well-defined semantics. Based on an ADBN, dynamic mission impact assessments can be reduced to filtering and smoothing problems in ADBNs.

A reduction of DMIA to problems in ADBNs has significant advantages: One obtains the possibility to include *evidence* into a model, i.e., one is able to include information about actual observations of impacts on nodes. Then a *filtering* problem is formed by the problem of assessing the impact of any node at a time t given all obtained evidences so far, i.e., represents an evolution of impact "live" at the current time. Observations, e.g., can origin from IDS alerts, antivirus scans, battle reports or plain-sight observations. A significant advantage of ADBNs is that evidence is not only processed "forwardly," but also "backwardly." For example, if X is influenced by Y, and given an observed impact on node $X = {+x}$, an ADBN anticipates implications on Y by the observation of $X = {+x}$. Similarly, a *smoothing* problem is formed by an impact assessment of any node at time k, given evidence obtained until time t. A solution to a smoothing problem delivers valuable information for forensic analyses about intermediate states of mission impacts in retrospect, by including implications of future observations to preceding timeslices.

Moreover, considering a Markov-1 property in ADBNs allows for *persisting* impacts. This is beneficial for situations, where a potential compromise on a node X already has led to an impact on other nodes $\vec{Y}$, whose inferred impact will persists if even, e.g., a cleaning operation is performed on the origin node X at some time t. A Markov-1 ADBN is able to raise awareness for a potential impact on nodes $\vec{Y}$ and, as well as, on a higher goal, such as a mission, even though an original source has been eliminated, but residues remain of them.

In order to deploy a dynamic mission impact assessment based on ADBN, one has to carefully consider the role of persistence, roles of external shock events and the roles of observations. In the following three paragraphs we discuss these roles, discuss different modeling approaches, and outline suited application fields for each model.

Persistence is the degree an impact shall persist over time, i.e., is a conditional probability fragment $p({+x}^t|{+x}^{t-1})$. In essence, two modeling approaches can be taken: $p({+x}^t|{+x}^{t-1}) = 1$ and $p({+x}^t|{+x}^{t-1}) < 1$. For the first option, one obtains a model in which any potentially caused impact will persist forever, and will "aggregate" over time. Note that such an aggregation is significantly different from an aggregation obtained through a score-based propagation approach, as one has to consider prior random probability distributions correctly. For example, if there solely exists a prior random variable Q with $P({+q}) = 0.1$ as the only source of any impact in a network, every impact assessment will remain below 0.1. Moreover, irregardless how often any node came in contact with the potential impact source, all probabilities of impact will vanish, once one observes $\neg q$. We believe that it is highly complicated to include these mechanisms into any score-based propagation approach. A total persistence approach is suited to impact assessments for ICT networks, as resources usually are not able to "heal" themselves from any adversarially caused impacts. The second option, $p({+x}^t|{+x}^{t-1}) < 1$, resembles a domain in which impacts do not completely persists, and "cool down" over time. This is an approach suited to, e.g., infectious disease monitoring, where, in fact, entities modeled in a resource dependency model, i.e., people, actually heal themselves over time. It is arguable that a decaying impact is as well applicable to an ICT related use case, as a compromise will unlikely persist forever, as compromised data (in compromised memory locations and compromised data on a file system) will consecutively be exchanged.

Observations of impacts and non-impacts of random variables must be carefully differentiated in a dynamic MIA based on an ADBN. One has to distinguish between induced observations and true observations; a differentiation related to Pearl's (2002) introduction on the do-calculus and is best explained with an example: Considering an infectious-disease monitoring system, one has to differentiate between a person being healthy, because he *has been* healed and between a person being tested to be healthy. The latter implies that there exists the probability that the person has never been infected, i.e., did never have a possibility to infect other persons he came in contact with. The first is an induced action and delivers solely information from the current state on. In an ICT security related use case this means that if one actively cleans a system, the system is not impacted *from now on*, but previously caused impacts will persist. If one deeply and passively inspects a system and comes to the conclusion that the system has never been compromised, one truly observes $\neg x$, which has an effect on all previously caused transitive impacts. In order to model these differences, one needs to reconsider the role of external shock events.

External shock events in an ADBN used for DMIA are used to include soft evidence and external sources of observations. In order to include external sources of observations, one is able to model an external shock event that inevitably will cause the desired observation. For example, to observe a non-impacted node $\neg x^t$, but X has actively been cleaned at time t, one includes a random variable SE^t with $P(+se^t) = 1$ and $p(\neg x^t|+se^t) = 1$ s.t. $\forall \vec{z} : P(\neg x^t|+se^t, \vec{z}) = 1$. Due to a context-specific independence of X^t on all its possible dependencies $\vec{Z}$, given $+se^t$, X^t is decoupled from all other potential sources of (non-)impact, and the observation $\neg x^t$ is completely accredited to SE^t. Using external shock events to include soft evidence is especially useful to include information obtained from present vulnerabilities on nodes in an ICT security use case, which we discuss in the following example.

Example 6.5 (Roles of vulnerabilities and initial believes in DMIA)**.** *In an ICT/ICS security related use case of a dynamic mission impact assessment, one must include information of potential compromises, i.e., information obtained through IDS alerts and from vulnerability advisories. An IDS alert, say "compromise on node X" represents a classic observation $+x^t$ and will affect other, transitively dependent devices even from previous timesteps, as the source of this observed compromise will lie in the network. Still, such sources must be modeled in an ADBN. We envision two potential use cases of DMIA, depending on two perspectives what should be achieved: (1) Viewing DMIA as a simulation of evolution of compromise, all sources of impacts are modeled as initial believes about compromise in B_0 in prior random distributions $P(\vec{X}^0)$. From this perspective, all future observations are credited to any circumstance that is modeled, i.e., all impacts must origin from inside a network. In order to integrate a degree of* *flexibility, one is able to include a "leakyness" in CPDs, i.e., $P(+x^t|\neg\vec{z}^t) > 0$ ($\vec{z}^t$ representing all dependencies of X^t). Still, by modeling leaky CPDs, one loses deeper explanations for potential impacts, as impacts can occur "from nowhere." (2) Viewing DMIA as a monitoring system, one is able to explicitly model all potential sources of impacts, but might over-accredit external- instead of internal-sources of impact. From a monitoring perspective, at every timestep for every vulnerability V an external shock event SE_V exists, as discussed in Example 6.4, and respective CPDs of affected nodes X accredit a (new) potential compromise due to this vulnerability at every timestep, i.e., $p(+x|+se_V) > 0$ and $P(+x|\neg\vec{z}) = 0$. By following this approach, one is further able to represent that it is more likely that a node becomes compromised, given it is prone to a vulnerability and actively retrieves information from a dependent, impacted node.*

This example outlines and discusses how to include information about (potential) compromises and threats in an ICT security DMIA based on an ADBN and outlines two possible perspectives that can be taken when using a DMIA.

We refrain from a deeper analysis and evaluation of the outlined approach as DMIA represents a too large novel research field and is beyond the scope of this thesis. Still, without the preceding chapters on ADBNs, an associated DPGM to a DMIA problem is not well-defined and would exactly suffer from problems evident in the taintedness domain. Without an ADBN, dependencies in a resource dependency network must be bent to consecutive timeslices, either enforcing an infinitesimal resolution of time, or ignoring crucial indirect causes and influences.

6.4 Related Work

Mission modeling and mission impact assessment is an emerging field of research; and, naturally in new, viral research areas, employ ad-hoc solutions using algorithms involving fudge factors. While delivering early results and acclaimed solutions for mission impact assessment, a formal definition of an underlying problem is yet missing. Employed fudge factors in newly established algorithms lead to untraceable and possibly spurious results demanding data-driven validations. Unfortunately, large, standardized datasets for validation are yet missing for mission impact assessment, and no evidence of any data-driven validation is given in the following presented works. Ad-hoc solutions can deliver acceptable results, if an expert is familiar with an employed solution and understands the nature of setup requirements, but frequently involves unvalidated and unverifiable "black magic" (non-context-free).

De Barros Barreto et al. (2013) introduce a well-understood modeling technique and use BPMN models to acquire knowledge. An impact assessment is based on various indexes and numerical scores, such as exploit index, impact factor, infrastructure capacity index, and graph distances. Various numerical factors are arbitrarily combined, without a mathematical foundation and cannot provide a transparent, understandable and verifiable assessment to an expert. Further, an assessment is solely based on direct impacts, leaving aside transitive impacts and/or defining a manual description of all dependencies between individual devices inside one organization, which is, in most of the cases an unfeasible process

Albanese et al. (2013) a well-modeled formalism for complex inter-dependencies of missions as a set of tasks. Using numerical scores and tolerances in a holistic approach Albanese et al. focus on cost minimization. Their approach can solely be validated holistically, as involved parameters do not bear local semantics and do not provide bias-free and context-free understandable results. Buckshaw et al. (2005) propose a quantitative risk management by involving various experts and present a score-based assessment based on individual values and a standardization using a weighted sum. Unfortunately, a mathematical foundation is missing and obtained results are only interpretable after deep training of experts in the characteristics of this approach. Buckshaw et al. themselves note that a validation of the proposed model requires large amounts of actual data and ground truth, which both are not available.

Jakobson (2011) presents a well-understood conceptual framework using interdependencies based on operational capacity at different abstraction layers. In this dependency model, impacts are propagated and reduce the operational capacity, which has a similar intention to our approach. However, Jakobson (2011) uses self-defined metrics for propagating impacts through Boolean gates, which cannot provide context- and bias-free understandable results or parametrization. Moreover, an explicit representation of "intra-asset" dependencies is required, i.e., all individual critical, and non-critical resources must be identified. (Musman et al., 2010) proposes the use of BPMN models and describes a process for evaluating impacts of cyber-attacks. However, (Musman et al., 2010) fails to get across any mathematical approaches or formal definitions for impact assessment.

Further works focused solely on modeling. For example, Goodall, D'Amico, and Kopylec (2009) focus on modeling and available data integration using ontologies but do not address an impact assessment. Another ontology-based approach is presented by D'Amico, Buchanan, Goodall, and Walczak (2010) and identifies multiple experts while noting that, e.g., system administrators are not capable of understanding an organization's missions.

In terms of (probabilistic) approaches towards assessments of impacts caused by vulnerabilities and attacks, probabilistic models have been researched by Wang et al. (2008), Liu and Man (2005), and Xie et al. (2010). However, they base their work on attack graphs and do not consider imperfect knowledge, e.g., unknown extents of damage causable by vulnerabilities, uncertainty of specific events and potentially disagreeing information sources as we do. Moreover, Xie et al., as well as, Liu and Man are significantly limited by the lack of supporting cyclic dependencies and do not consider any mission impact relations. Chung et al. (2013) consider a probabilistic approach as well to determine the likelihoods of explicit attack paths. However, presented probability theory by Chung et al. (2013) is not sound and voids fundamental principles of probabilistic inference in multiply connected graphs. Other impact propagation approaches, e.g., by Kheir et al. (2009) or Jahnke et al. (2007), claiming to handle details such as disagreeing information sources and cycles, are not probabilistic based and degrade to a handcrafted propagation algorithm with arbitrary scores, where parameters are only assessable by deeply trained experts and obtained results can only be used in a holistic way, as they provide no directly interpretable meaning.

6.5 Conclusion

By using filtering and smoothing problems for mission impact assessments, we incorporate valuable reasoning mechanisms into an assessment predestined for detailed predictive and forensic analyses. Although these assessments are based on computational expensive probabilistic inference problems, approximate solutions based on sampling procedures deliver detailed solutions and scale linearly with domain size and inference timepoint, as shown in Section 2.4.

In contrast to previous approaches, we refrain from introducing new score-based propagation algorithms, whose results and parameters are only interpretable to a limited extend. We rely on the expertise of different experts and merged all views without losing information or forcing an expert into a knowledge field he cannot understand. By exploiting local semantics of DPGMs, all defined parameters are validatable and understandable locally, i.e., do not require a global view towards a complete system and algorithm. By reducing the novel problem of impact assessments onto well-understood problems of probabilistic inference in DPGMs, obtained results are understandable without indepth knowledge about the assessment's calculation approach. Moreover, obtained results do not need to be validated against ground truth, as soon as an underlying model and parameters have been validated.

We highly believe that the presented dynamic dependent-timeslice, as well as, independent-timeslice approach is not limited towards ICT related use cases, and is able to deliver valuable advantages for other civil- and military situational awareness and mission impact assessment and is, as well, applicable towards infection- and disease spreading, monitoring and prevention.

Chapter 7

Discussion and Conclusion

Throughout this thesis we have introduced a novel form of PGMs and DPGMs, called ADBNs, as a solution for eliminating limitations of classical DBN formalisms while maintaining desired local and global semantics of (D)BNs. We have shown that classic dynamic Bayesian networks are significantly limited in anticipating indirect influences and cause conflicts when attempting to represent causal dependencies from local points of views. These conflicts arise from using a modeled dimension (time) for assuring syntactic requirements of acyclicity in classical DBN formalisms. By identifying activator criteria of random variables in (dynamic) Bayesian networks, we introduce ADBNs, which are able to move acyclicity constraints from a design phase to a later operation phase, and which, by adequate modeling approaches, are able to resolve their acyclicity constraints intrinsically. Without the need of external reasoning frameworks, degrading a Bayesian network to a reasoning process, we provide a solid mathematical basis similar to Bayesian networks with a causally correct anticipation of indirect causes while remaining in a classical familiar calculus.

We have discussed and derived solutions to commonly known query answering problems and, further, that ADBNs do not introduce any computational overhead, compared to its closest relative in classic DBN formalisms. To reduce computational complexity of finding solutions to common query answering problems in ADBNs, we derived approximate inference techniques for ADBNs and have shown that familiar approaches remain applicable. Moreover, we have derived a learning approach for ADBNs. In fact, we are able to show that classical "virtual data counts" approaches remain applicable, and, once again, that cyclic ADBNs do not introduce any modeling or computational overhead. We demonstrate that, even if parts of structure relevant information are missing and, thus, no structure is knowable, ADBNs are still able to learn model parameters which reciprocally decide an effective structure. We have shown that not even the most general form of DBNs is able to learn the taintedness-domain of the running example, and that cyclic ADBNs are causally required. As ADBNs represent superclasses of Markov-1 DBNs, parameter and structural learning of ADBNs are fused into one atomic phase of constraint parameter learning, while being able to handle hidden variables and hidden structural information.

Furthermore, we investigated implications of activator random variables in CPDs, and are now able to formalize an innocuousness property of random variables, which is often associated with Boolean combination functions for general CPDs. Based on a formalization with random variables taking the role of activators, we are able relax restrictions on ADBNs and give a quantitative evaluation of restrictions posed on such networks. By a further investigation of properties of CPDs in specific modeling approaches, we extend the theory of ADBNs towards extended ADBNs. We show that structural information need neither be evident nor be supplied by observations for eADBNs. This is highly beneficial for applications where one is not able to proactively assure that observations, and further, *regular* observations will be available. In effect, and to the best of our knowledge, an eADBN is the first probabilistic graphical model that is able to represent multiple joint probability distributions in one model. Further, to the best of our knowledge, an eADBN is the first probabilistic graphical model that is able to intrinsically represent a distribution over its own network structure and is able to base inference on all structural candidates at the same time.

We show that eADBNs are able to intrinsically handle not necessarily regular observations (according to Theorem 2.1 and 4.1) where *insufficient* structural information is evident, and multiple regular instantiations, i.e., structural candidates, remain possible. If one directly observes a non-regular instantiation of random variables in (e)ADBNs, one is prone to a challenge for future work on regularity in (e)ADBNs, previously outlined in Section 5.3: The model declines the possibility of this observation with a probability of 0. This is a similar problem evident from diagonal networks, where observations including effective indirect influences are deemed inexplicable as shown in Example 2.7. In such a particular situation, an action is required to assure regularity of (e)ADBNs, which, mostly likely, must be an invasive modification of an observation. Such a modification would be, for example, to keep all instantiation assignments, but to move small subsets of observations to a neighboring timestep. Another option is to modify some instantiation assignments in an observation by finding the most likely, but minimally invasive alternative observation that conforms to Theorem 2.1, 4.1, or 5.1. As discussed in Section 2.5 we refrain from deriving such an action in this thesis, as such an action involves a witfull modification of the world, which stands in significant conflict with our views on DBNs that shall be a direct representation of the world instead of a reasoning tool. We, therefore, advise to rethink the underlying model or problem once such a situation arises.

All of above has been achieved by specific parameter settings of CPDs. At no time, an ADBN requires an external reasoning framework that explicitly analyzes an instantiation or observation and explicitly creates a (new) dynamic Bayesian network. Moreover, the JPD and CPD definitions remain in a classical familiar calculus without requiring any new form of operators such as those novel operators introduced by Milch et al. (2005) or by Bilmes (2000). In fact, neither exact inference nor (exact) learning, even from incomplete datasets, requires a topological ordering to become explicit at any time. This is highly beneficial, as derived algorithms for solving commonly known problems are not tailored to one specific regularity constraint and remain universally applicable even if novel regularity predicates are discovered in future work. Furthermore, we have shown that approximate inference techniques renders finding answers to inference problems in ADBNs manageable, even in largely scaled domains.

The discussed sequential importance sampling approach requires that at every timestep an effective topological ordering is made explicitly, i.e., a topological ordering must be obtained from made observations. Performing a topological sorting does not represent a bottleneck and the requirement does not come as a surprise, as discussed earlier. Moreover, the presented SIS and SIR algorithm remains globally applicable to universal regularity predicates as well; it is not tailored to, e.g., an acyclicity nor an innocuousness predicate.

Throughout this thesis we exploited various forms of independencies in Bayesian networks and focused on an increased expressiveness, but focused less on their implications on more efficient exact inference. In fact, various forms of independencies deliver promising points of optimization on exact inference as, e.g., discussed by Boutilier et al. (1996), Poole and Zhang (1996, 2003), and Heckerman and Breese (1996). Moreover, the combination of exact and approximate inference in ADBNs represent a highly interesting future research area: If substructures are formed for which exact inference remains tractable, exact inference can be combined with approximate inference for the remainder of the network. As shown by Doucet et al. (2000) in the form of Rao-Blackwellised Particle Filtering (RBPF) such approaches deliver more accurate inference results than, e.g., a classical SIR procedure. While in classical DBNs a structure remains constant and a "tractable substructure" is known in advance, adapting RBPF to ADBNs motivates a new form of RBPF: substructural-adaptive RBPF.

In essence, this work is motivated by being able to include cyclic structures in a PGM, and one could switch from directed models towards chain graphs as described by Drton (2009) or to general Markov networks, which, both, support cyclic dependency structures. However, by switching to (partially) undirected models one loses the direct and intuitive interpretation of locally specified conditional probability distributions by introducing the burden of computing non-local normalization quotients. In cyclic (e)ADBNs, all desired local semantics are preserved without introducing any computational or modeling overhead. On top of that, an undirected model inherently models a different domain as undirected edges imply symmetric influences, whereas in an ADBN one can specify asymmetric influence strengths. Further, an undirected model implies a steady state influence domain, where influences let random variables converge to a stable state. The latter is an application of a diagonal DBN, where a cyclic directed graphical model is unfolded into a diagonal DBN and simulated over a long timeperiod to obtain said stable state. Note that, the misuse of time to simulate this cyclic behavior is exactly the problem that motivates this thesis and leads to the problem of being unable to anticipate indirect influences in one timestep of a DBN, if time is supposed to represent an actual wall-clock time, as envisioned in ADBNs.

Furthermore, we demonstrate and punctuate the need of locally interpretable parameters and well-defined semantics of probabilistic graphical models at a real world use case, where experts' assessments have to be trusted and have to be integrated directly. We show that cyclic ADBNs arise naturally and are highly beneficial for real world applications.

Considering all of above, we feel confident to conclude that (dynamic) Bayesian networks can contain cyclic dependencies, can be based on cyclic graphs, and are required to be based on cyclic graphs representing distributions of multiple full joint probability distributions in one first-class model.

Appendix A

Derivation of a Bayesian Network's JPD

It is stated by Theorem 1.1 that the global semantics of a PGM based on a DAG are given as the product of all locally defined CPDs. A proof is straightforward by Bayes' theorem, but will later be required in a deeper, complex setting and is therefore given here for reference.

Proof of Theorem 1.1 (Bayesian network). A Bayesian network is defined by a set of CPDs, in the form $P(X|\vec{par}_X)$, where $\vec{par}_X$ represents the set of random variables that influence X. Bayes' theorem is given as

$$P(X|Y) \cdot P(Y) = P(X,Y) = P(Y|X) \cdot P(X) \ . \tag{A.1}$$

Let $B = \langle V, E \rangle$ be a Bayesian network. Let $X_i \in \vec{X}$ represent the random variable associated with the i-th vertex in V according to a topological sorting. Then, X_1 is not influenced by any other random variables, and X_2 is influenced (at most) by X_1, i.e., $\vec{par}_{X_2} = \langle X_1 \rangle$ or $\vec{par}_{X_2} = \emptyset$. According to Theorem 1.1 the JPD is given as

$$\begin{aligned} P(\vec{X}) &= \prod_{X \in \vec{X}} P(X|\vec{par}_X) = \prod_{i}^{n} P(X_i|\vec{par}_{X_i}) \\ &= P(X_n|\vec{par}_{X_n}) \cdot P(X_{n-1}|\vec{par}_{X_{n-1}}) \cdot \ldots \cdot P(X_2|\vec{par}_{X_2}) \cdot P(X_1|\vec{par}_{X_1}) \end{aligned}$$

If X_2 is not directly influenced by X_1, then $P(X_2|X_1) = P(X_2)$ holds $\forall X_1 = x_1 \in \mathrm{dom}(X_1)$ in the definition of X_2's local CPD, i.e., X_1 is irrelevant for a decision on which parameter to use, as the parameter only depends on the value of X_2. In reverse, any "irrelevant," i.e., independent, random variable can be added to a conditional probability distribution without any modification. Therefore, $P(X_2|\vec{par}_{X_2}) = P(X_2|X_1)$ always holds. Then, under Bayes' theorem, one obtains

$$\begin{aligned} P(\vec{X}) &= P(X_n|\vec{par}_{X_n}) \cdot P(X_{n-1}|\vec{par}_{X_{n-1}}) \cdot \ldots \cdot P(X_2|X_1) \cdot P(X_1) \\ &= P(X_n|\vec{par}_{X_n}) \cdot P(X_{n-1}|\vec{par}_{X_{n-1}}) \cdot \ldots \cdot P(X_2, X_1) \ . \end{aligned}$$

Likewise, $\vec{par}_{X_3}$ can at most be $\langle X_2, X_1 \rangle$, or less. And, following the same procedure, one obtains

$$\begin{aligned} P(\vec{X}) &= P(X_n|\vec{par}_{X_n}) \cdot P(X_{n-1}|\vec{par}_{X_{n-1}}) \cdot \ldots \cdot P(X_3|X_2, X_1) \cdot P(X_2, X_1) \\ &= P(X_n|\vec{par}_{X_n}) \cdot P(X_{n-1}|\vec{par}_{X_{n-1}}) \cdot \ldots \cdot P(X_3, X_2, X_1) \\ &\vdots \\ &= P(X_n|X_{n-1}, X_{n-2}, \ldots, X_1) \cdot P(X_{n-1}, X_{n-2}, \ldots, X_1) \\ &= P(X_n, X_{n-1}, \ldots, X_3, X_2, X_1) \ . \end{aligned}$$

Therefore, the product of all locally defined CPDs represents, in fact, the full joint probability distribution over all random variables. ■

The proof further shows why $\langle V, E \rangle$ must be acyclic, as otherwise a topological sorting is undefined. In this thesis we show that cyclic PGMs exist for which the joint probability distribution is still defined as the product of all locally defined CPDs, that all desired properties of Bayesian networks remain, and that solutions to commonly known problems in DBNs remain applicable without introducing any overhead.

Appendix B

Proof of ADBN Well-Definedness

We prove Theorem 2.1 of the well-definedness of a (cyclic) ADBN by showing that Proposition 2.4, i.e., the joint probability over all random variables of a dense intra-timeslice ADBN, corresponds to a joint probability of a well-defined DBN (according to Proposition 2.1) under Theorem 2.1. This is a proof based on one special ADBN, but Proposition 2.3 states that a dense intra-timeslice ADBN includes all possible intra-timeslice dependencies and thus the following proof is a proof for Theorem 2.1. An ADBN can include inter-timeslice dependencies, but these are subject to the well-definedness Proposition 2.1 (see Theorem 2.1) and the following proof is equivalent.

Notation B.1 (Vector probability operands)**.** *For brevity, we define a probability of a vector containing random variables as a shorthand notation for a product of probabilities. Let $\vec{X}$, $\vec{Z}$, $|\vec{X}| = |\vec{Z}|$ be column vectors of random variables. Let $\mathbf{P}_\Gamma(\vec{X}|Y,\vec{Z})$ denote the product of probabilities $P(X_i|Y,Z_i)$ where X_i and Z_i are taken row-wise from $\vec{X}$ and $\vec{Z}$, except rows identified in the exclusion-set Γ. Scalars Y are repeated in every row. Formally,*

$$\mathbf{P}_\Gamma(\vec{X}|Y,\vec{Z}) = \prod_i P(X_i|Y,Z_i) \qquad i \in \{1 \leq i \leq |\vec{X}|\}\backslash\Gamma$$

Respectively, we apply this notation to (conditional) probabilities with n-ary dependencies and without dependencies, i.e., prior random variables.

Notation B.2 (Lexicographic order)**.** *Let $\prec$ be a lexicographic term order, such that $X_*^{t-1} \prec X_*^t$, $X_i^t \prec X_{i+1}^t$, and $A_{**}^{t-1} \prec A_{**}^t$, $A_{i*}^t \prec A_{(i+1)*}^t$, $A_{ij}^t \prec A_{i(j+1)}^t$, and $A_{**}^t \prec X_*^t$, $X_*^{t-1} \prec A_{**}^t$.*

We rewrite Proposition 2.4 using Notation B.1 as:

Proposition B.1 (Shorthand joint probability distribution notation)**.** *From Proposition 2.4, a dense intra-timeslice ADBN's semantics is*

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = \prod_{X_k^0 \in \vec{X}^0} P(X_k^0) \cdot \prod_{i=1}^{t} \prod_{X_k^i \in \vec{X}^{\,i}} P(X_k^i | \vec{X}^{i^\intercal} \backslash X_k^i, \vec{A}_k^{i^\intercal}, X_k^{i-1}) \cdot \prod_{A_{cv}^i \in \vec{\mathcal{A}}^i} P(A_{cv}^i) \ ,$$

written for brevity using Notation B.1 as

$$P(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}}) = \mathbf{P}(\vec{X}^0) \cdot \prod_{i=1}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i^{\intercal}} \backslash \vec{X}^i, A^{i^{\intercal}}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \qquad \blacktriangle$$

The shorthand allows for shorter notation of products where some indexes of a product need to be excluded.

Proof of Theorem 2.1 (ADBN well-definedness). We show that the joint probability distribution stated in Proposition 2.4 is indeed well-defined for every instantiation of $(\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t})$, if for all t the instantiation $\vec{a}^t$ of $\vec{\mathcal{A}}^t$ follows Eq. 2.3 by Theorem 2.1. We show this by reversing local conditional independency assumptions in the semantic joint probability and that one obtains a topological ordering of a syntactical graph structure, i.e., a syntactically cyclic graph structure in $B_{\rightarrow}$ is semantically a DAG for which Bayesian network semantics defines the same joint probability distribution as defined under restrictions in Theorem 2.1. B_0 can be written as

$$P(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}}) = P(X_1^0) \cdot \ldots \cdot P(X_n^0) \cdot \gamma = P(X_1^0, \ldots, X_n^0) \cdot \gamma = P(\vec{X}^{0^{\intercal}}) \cdot \gamma \ ,$$

with

$$\gamma = \prod_{i=1}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i^{\intercal}} \backslash \vec{X}^i, A^{i^{\intercal}}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ .$$

Consecutively, one is able to roll up the joint distribution according to Bayes' chain rule. Considering an extreme case of a set of activators corresponding to Eq. 2.3, it is straightforward that in any instantiation following Eq. 2.3 it must always hold that $\exists X_{E1}^1 : \forall i A_{i(E1)}^1 = \mathit{false}$, such that due to Eq. 2.1, the set of activators and previous states uniquely identify the CPD entry of $P(X_{E1}^1 | \ldots)$ and X_{E1}^1 becomes locally independent of all other $\vec{X}^1$, such that the joint probability can be written as

$$\begin{aligned} &P(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}}) = \\ &\quad P(\vec{X}^{0^{\intercal}}) \cdot P(X_{E1}^1 | *, \vec{A}_{E1}^{1^{\intercal}}, X_{E1}^0) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1^{\intercal}} \backslash \vec{X}^1, A^{1^{\intercal}}, \vec{X}^0) \cdot \mathbf{P}(\vec{\mathcal{A}}^1) \\ &\qquad \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i^{\intercal}} \backslash \vec{X}^i, A^{i^{\intercal}}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \end{aligned} \tag{B.1}$$

By reversing X_{E1}^1's local conditional independence assumptions, i.e., one reverses "if X is not directly influenced by Z then $P(X|Y,Z) = P(X|Y)$ holds in X's local CPD" and one enriches X's local CPD $P(X|Y)$ with locally independent random variable(s) Z to $P(X|Y,Z)$, one obtains

$$\begin{aligned} &P(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}}) = \\ &\quad P(\vec{X}^{0^{\intercal}}) \cdot P(X_{E1}^1 | *, \vec{\mathcal{A}}^{1^{\intercal}}, \vec{X}^{0^{\intercal}}) \cdot \mathbf{P}(\vec{\mathcal{A}}^1) \cdot \mathbf{P}_{\{E1\}}(\vec{X}^1 | \vec{X}^{1^{\intercal}} \backslash \vec{X}^1, A^{1^{\intercal}}, \vec{X}^0) \\ &\qquad \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i | \vec{X}^{i^{\intercal}} \backslash \vec{X}^i, A^{i^{\intercal}}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \end{aligned}$$

Hence, with

$$\begin{aligned} \mathbf{P}(\vec{\mathcal{A}}^t) &= P(A_{12}^t) \cdot \ldots \cdot P(A_{1n}^t) \cdot \ldots \cdot P(A_{n1}^t) \cdot \ldots \cdot P(A_{n(n-1)}^t) \\ &= P(A_{12}^t, \ldots, A_{1n}^t, \ldots, A_{n1}^t, \ldots, A_{n(n-1)}^t) = P(\vec{\mathcal{A}}^{t^{\intercal}}) \ , \end{aligned}$$

one can combine $P(\vec{X}^{0^\intercal})$ with $P(\vec{\mathcal{A}}^{1^\intercal})$ to $P(\vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal})$ s.t. the probability distribution of the first eliminated state variable X^1_{E1} can be combined as $P(X^1_{E1}|*, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \cdot P(\vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) = P(X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal})$, and one obtains

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ &\cdot \mathbf{P}_{\{E1\}}(\vec{X}^1|\vec{X}^{1^\intercal} \backslash \vec{X}^1, A^{1^\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \end{aligned}$$

Consecutively, there $\exists X^1_{E2} : \forall \{i \backslash E1\}\, A^1_{i(E2)} = \textit{false}$ in every regular instantiation s.t.

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \cdot P(X^1_{E2}|*, X^1_{E1}, *, \vec{A}^{1^\intercal}_{E2}, X^0_{E2}) \\ &\cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1|\vec{X}^{1^\intercal} \backslash \vec{X}^1, A^{1^\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ , \end{aligned}$$

for which one reverses the local conditional independency again and one obtains

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \cdot P(X^1_{E2}|*, X^1_{E1}, *, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ &\cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1|\vec{X}^{1^\intercal} \backslash \vec{X}^1, A^{1^\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ , \end{aligned}$$

which, according to Bayes' chain rule, can be written as

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X^1_{E2}, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ &\cdot \mathbf{P}_{\{E1,E2\}}(\vec{X}^1|\vec{X}^{1^\intercal} \backslash \vec{X}^1, A^{1^\intercal}, \vec{X}^0) \cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \end{aligned}$$

Consecutively repeating this process for every remaining X_{Ei} where the i^{th} elimination variable is maximally directly dependent on the previous $(i-1)$ eliminated variables, one, henceforth, approaches the elimination of X^1_{En}, which is dependent on up to every other $\vec{X}^1$, which are all eliminated variables up to now, i.e.,

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ \cdot P(X^1_{En}|X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{A}^{1^\intercal}_{En}, X^0_{En}) &\cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ , \end{aligned}$$

for which one reverses the local conditional independency again and combines the joint probability finally to

$$\begin{aligned} P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) &= P(X^1_{En}, X^1_{E(n-1)}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ &\cdot \prod_{i=2}^{t} \mathbf{P}(\vec{X}^i|\vec{X}^{i^\intercal} \backslash \vec{X}^i, A^{i^\intercal}, \vec{X}^{i-1}) \cdot \mathbf{P}(\vec{\mathcal{A}}^i) \ . \end{aligned}$$

Indeed, one already obtains a topological ordering $>$ for the first two timeslices of $X^1_{En} > X^1_{E(n-1)} > \ldots > X^1_{E1} > \vec{\mathcal{A}}^{1^\intercal} > \vec{X}^{0^\intercal}$.

Following this procedure for the remaining $t > 1$, one finally obtains

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = P(X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ \cdot P(X^t_{En} | X^t_{E(n-1)}, \ldots, X^1_{E1}, A^{1^\intercal}_{En}, X^{t-1}_{En}) \ .$$

With a final reverse local conditional independence assumption,

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = P(X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \\ \cdot P(X^t_{En} | X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \ ,$$

one obtains a complete topological ordering and a full joint probability over all random variables of

$$P(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}) = \\ P(X^t_{En}, X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal}) \ . \quad \text{(B.2)}$$

One, thus, obtains a complete topological ordering that syntactically defines an equivalent Bayesian network (unrolled dynamic Bayesian network) $(B_0, B'_\rightarrow)$ with the same random variables $\vec{X}^{0:t}, \vec{\mathcal{A}}^{1:t}$, i.e., defines the same full joint probability distribution $P(X^t_{En}, X^t_{E(n-1)}, \ldots, X^t_{E1}, \vec{\mathcal{A}}^{t^\intercal}, \ldots, X^1_{En}, \ldots, X^1_{E1}, \vec{\mathcal{A}}^{1^\intercal}, \vec{X}^{0^\intercal})$. Therefore, the product of all locally defined CPDs, in fact, is the semantics of a dense intra-timeslice ADBN, despite being based on a cyclic graph and does not require any global normalization factors. ■

Appendix C

Appendix on Learning ADBNs

C.1 Extended Smoothing Problem

Theorem C.1 (Exact solution to the extended smoothing problem)**.** *Given a complete smoothing problem* $\texttt{ExtdSP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\beta}^{1:t}, t)$*, finding an exact solution is linear in t. Finding an exact solution is exponential in the maximal dimension of unobserved variables* $\vec{\zeta}^*$*,* $\vec{\beta}^*$ *in a timestep* $0 < * \leq t$*, and in the largest domain* $\mathrm{dom}(\zeta_+)$*,* $\mathrm{dom}(\beta_+)$ *of all random variables* $\zeta_+ \in \vec{\zeta}^{0:t}$*,* $\beta_+ \in \vec{\beta}^{1:t}$*. Finding an exact solution is exponential in the dimension of number of random variables* $|\vec{X}^{t-1:t}|$*,* $|\vec{\mathcal{A}}^{t-1:t}|$ *and a respective maximal domain size* $\mathrm{dom}(X_+)$*,* $\mathrm{dom}(A_{++})$ *of all random variables* $X_+ \in \vec{X}^{t-1:t}$*,* $A_{++} \in \vec{\mathcal{A}}^{t-1:t}$*.* ▲

Theorem C.1 is proven by showing that an algorithm exists that finds an exact solution to $\texttt{ExtdSP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\beta}^{1:t}, t)$ in time-complexity $\mathcal{O}(t \cdot |\,\mathrm{dom}(X_+)|^{|\vec{X}^t|} \cdot |\,\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^t|} \cdot |\,\mathrm{dom}(\zeta_+)|^{|\vec{\zeta}^*|} \cdot |\,\mathrm{dom}(\beta_+)|^{|\vec{\beta}^*|})$ and with $\mathcal{O}(|\,\mathrm{dom}(X_+)|^{|\vec{X}^{t-1:t}|} \cdot |\,\mathrm{dom}(A_{++})|^{|\vec{\mathcal{A}}^{t-1:t}|})$ space-complexity for storing one extended smoothing distributions.

Proof of Theorem C.1 (Extended smoothing problem). An algorithm obtaining solution for an extended smoothing problem $\texttt{ExtdSP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\beta}^{1:t}, t)$ is similar to an algorithm derived in Proof of Theorems 2.5 and 2.4. For the case of a dense intra-timeslice ADBN an exact solution to an extended smoothing problem is given by straight marginalization from a JPD as

$$P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}}, \vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}} | \vec{z}^{0:t^{\intercal}}, \vec{\beta}^{1:t^{\intercal}})$$
$$= \alpha \cdot \sum_{\vec{\zeta}^{0:k-2}} \sum_{\vec{\beta}^{1:k-2}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}})$$

using the JPD of a dense intra-timeslice ADBN, one obtains

$$\begin{aligned}
&P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}}, \vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}} | \vec{z}^{0:t^{\intercal}}, \vec{\mathscr{b}}^{1:t^{\intercal}}) \\
&\qquad = \alpha \cdot \sum_{\vec{\zeta}^{0:k-2}} \sum_{\vec{\beta}^{1:k-2}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:t-1^{\intercal}}, \vec{\mathcal{A}}^{1:t-1^{\intercal}}) \\
&\qquad\qquad \cdot \prod_{X_i^t \in \vec{X}^t} P(X_i^t | \vec{X}^{t^{\intercal}} \backslash X_i^t, A_i^{t^{\intercal}}, X_i^{t-1}) \cdot \prod_{A_{ij}^t \in \vec{\mathcal{A}}^t} P(A_{ij}^t) \ . \qquad \text{(C.1)}
\end{aligned}$$

Using an intermediate joint probability distribution definition, one obtains

$$\begin{aligned}
&P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}}, \vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}} | \vec{z}^{0:t^{\intercal}}, \vec{\mathscr{b}}^{1:t^{\intercal}}) \\
&= \alpha \cdot \sum_{\vec{\zeta}^{0:k-2}} \sum_{\vec{\beta}^{1:k-2}} \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{0:k^{\intercal}}, \vec{\mathcal{A}}^{1:k^{\intercal}}) \cdot P(\vec{X}^{k+1:t^{\intercal}}, \vec{\mathcal{A}}^{k+1:t^{\intercal}}) \\
&= \alpha \cdot \sum_{\vec{\zeta}^{0:k-2}} \sum_{\vec{\beta}^{1:k-2}} P(\vec{X}^{0:k^{\intercal}}, \vec{\mathcal{A}}^{1:k^{\intercal}}) \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t^{\intercal}}, \vec{\mathcal{A}}^{k+1:t^{\intercal}}) \\
&= \alpha \cdot \sum_{\vec{\zeta}^{0:k-2}} \sum_{\vec{\beta}^{1:k-2}} P(\vec{X}^{0:k-1^{\intercal}}, \vec{\mathcal{A}}^{1:k-1^{\intercal}}) \cdot P(\vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}}) \\
&\qquad\qquad \cdot \sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t^{\intercal}}, \vec{\mathcal{A}}^{k+1:t^{\intercal}}) \\
&= \alpha \cdot \left(\sum_{\vec{\zeta}^{0:k-2}} \sum_{\vec{\beta}^{1:k-2}} P(\vec{X}^{0:k-1^{\intercal}}, \vec{\mathcal{A}}^{1:k-1^{\intercal}}) \right) \cdot \left(P(\vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}}) \right) \\
&\qquad\qquad \cdot \left(\sum_{\vec{\zeta}^{k+1:t}} \sum_{\vec{\beta}^{k+1:t}} P(\vec{X}^{k+1:t^{\intercal}}, \vec{\mathcal{A}}^{k+1:t^{\intercal}}) \right) \\
&= \alpha \cdot P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}} | \vec{z}^{0:k^{\intercal}}, \vec{\mathscr{b}}^{1:k^{\intercal}}) \cdot P(\vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}}) \cdot P(\vec{z}^{k+1:t^{\intercal}}, \vec{\mathscr{b}}^{k+1:t^{\intercal}} | \vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}}) \ . \\
&\qquad\qquad \text{(C.2)}
\end{aligned}$$

Evaluating the extended smoothing Equation C.2 for all instantiations of $\vec{X}^{k-1}, \vec{\mathcal{A}}^{k-1}, \vec{X}^{k}, \vec{\mathcal{A}}^{k}$ decrementally for descending $k = t \ldots 0$ is an algorithm that gives an exact solution to the extended smoothing problem $\texttt{ExtdSP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$. The decremental evaluation allows for storing the backward message: an intermediate result $P(\vec{z}^{k+1:t^{\intercal}}, \vec{\mathscr{b}}^{k+1:t^{\intercal}} | \vec{X}^{k+1^{\intercal}}, \vec{\mathcal{A}}^{k+1^{\intercal}})$ from an evaluation of $P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}}, \vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}} | \vec{z}^{0:t^{\intercal}}, \vec{\mathscr{b}}^{1:t^{\intercal}})$ is needed in an upcoming evaluation of $P(\vec{X}^{k-2^{\intercal}}, \vec{\mathcal{A}}^{k-2^{\intercal}}, \vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}} | \vec{z}^{0:t^{\intercal}}, \vec{\mathscr{b}}^{1:t^{\intercal}})$. Thus, obtaining the last term of Equation C.2 is constant in t for every evaluation. Obtaining the middle term $P(\vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}})$ is linear in the number of random variables of timeslice k.

The first term $P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}} | \vec{z}^{0:k-1^{\intercal}}, \vec{\mathscr{b}}^{1:k-1^{\intercal}})$ of the extended smoothing equation poses a filtering problem, for which a solution is found in $\mathcal{O}(1)$ in a storage from a solution to an offline filtering problem $\texttt{OffFP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$, which is found in $\mathcal{O}(t)$ by the previously derived algorithm in Proof of Theorems 2.2 and 2.3.

For a fixed $B_0, B_{\rightarrow}$ and a fixed number and domain size of unobserved variables per timeslice in $\vec{z}^{1:t}, \vec{\mathscr{b}}^{1:t}$ the algorithm has linear time-complexity $\mathcal{O}(t)$ and linear space-complexity $\mathcal{O}(t)$. It requires storage for one distribution $P(\vec{X}^{k-1^{\intercal}}, \vec{\mathcal{A}}^{k-1^{\intercal}}, \vec{X}^{k^{\intercal}}, \vec{\mathcal{A}}^{k^{\intercal}} | \vec{z}^{0:t^{\intercal}}, \vec{\mathscr{b}}^{1:t^{\intercal}})$ and storage for a solution of $\texttt{OffFP}(B_0, B_{\rightarrow}, \vec{z}^{0:t}, \vec{\mathscr{b}}^{1:t}, t)$. ■

C.2 Derivation of EM Procedure

We prove Theorem 3.1 by showing that the proposed procedure in fact maximizes the likelihood of the dataset, i.e., the probability of observing the dataset under the optimized parameters. To do so, we analytically solve Eq. 3.2. We derive an ADBN EM procedure for the most general, i.e., dense, intra-timeslice ADBN, which encapsulates all possible intra-timeslice DBN structures and onto which all other intra-timeslice (A)DBNs are reducable.

Proof of Theorem 3.1 (EM procedure). Optimized parameters $\vec{\vartheta}^*$ are obtained by

$$\vec{\vartheta}^* = \arg\max_{\vec{\Theta}} P_{\vec{\Theta}}(\vec{d}) = \arg\max_{\vec{\Theta}} \log\left(P_{\vec{\Theta}}(\vec{d})\right) , \tag{C.3}$$

i.e., by definition maximize the likelihood of observing a dataset under a given parameter set. In the following, we explicitly derive one parameter.

One obtains

$$\log\left(P_{\vec{\Theta}}(\vec{d})\right) = \log\left(\sum_{\vec{\zeta}^{0:t}}\sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|\vec{d}^{0:t}) \cdot P_{\vec{\Theta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})\right) ,$$

where Jensen's inequality for the concav function log, i.e.,

$$\log\left(\sum_i p_i x_i\right) \geq \sum_i p_i \log(x_i)$$

is applicable. Under Jensen's inequality one obtains

$$\log\left(P_{\vec{\Theta}}(\vec{d})\right) = \sum_{\vec{\zeta}^{0:t}}\sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|\vec{d}^{0:t}) \cdot \log\left(P_{\vec{\Theta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal})\right) , \tag{C.4}$$

as $\sum_{\vec{\zeta}^{0:t}}\sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|\vec{d}^{0:t}) = 1$, if the dataset yields a well-defined ADBN, i.e., if the dataset only contains regular instantiations. Using the definition of the JPD of a dense intra-timeslice ADBN (Eq. 2.4) yields

$$\log\left(P_{\vec{\Theta}}(\vec{d})\right) = \sum_{\vec{\zeta}^{0:t}}\sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|\vec{d}^{0:t})$$

$$\cdot\Big(\sum_{X_k^0\in\vec{X}^0} \mathfrak{P}_\Theta(X_k^0) + \sum_{i=1}^{t}\big(\sum_{X_k^i\in\vec{X}^i} \mathfrak{P}_\Theta(X_k^i|\vec{X}^{i^\intercal}\backslash X_k^i, \vec{A}_k^{i^\intercal}, X_k^{i-1}) + \sum_{A_{cv}^i\in\vec{\mathcal{A}}^i} \mathfrak{P}_\Theta(A_{cv}^i)\big)\Big) ,$$

with $\mathfrak{P}(\cdot) = \log P(\cdot)$ for brevity. $P_{\vec{\vartheta}}(\vec{X}^{0:t^\intercal}, \vec{\mathcal{A}}^{1:t^\intercal}|\vec{d}^{0:t})$ is a numerical value and values of random variables are uniquely identified by $\vec{d}^{0:t}$ or by summation over the unobserved variables in the dataset.

Under a stationary process, i.e., $P_\Theta(\cdot^i) = P_\Theta(\cdot^j) = P_\Theta(\cdot)$, an optimized parameter set $\vec{\vartheta}^*$ is obtainable in a closed form. Let $X_\lambda^t \in \vec{X}^t$ be some random variable and let $A_{\mu\nu}^t \in \vec{\mathcal{A}}^t$ be some activator. In the following, we extract one parameter $P_\Theta(x_\lambda^t|\vec{x}^{t^\intercal}\backslash x_\lambda^t, \vec{a}_\lambda^{t^\intercal}, x_\lambda^{t-1})$ from all sum-products to allow for partial derivation. First, one is able to represent the likelihood of data as

$$
\begin{aligned}
\log\left(P_{\vec{\Theta}}(\vec{d})\right) &= \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \sum_{X_k^0 \in \vec{X}^0} \mathfrak{P}_\Theta(X_k^0) \\
&+ \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \sum_{i=1}^{t} \sum_{X_k^i \in \vec{X}^{\,i}} \mathfrak{P}_\Theta(X_k^i | \vec{X}^{i^{\mathsf{T}}} \backslash X_k^i, \vec{A}_k^{i^{\mathsf{T}}}, X_k^{i-1}) \\
&+ \sum_{\vec{\zeta}^{0:t^{\mathsf{T}}}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \sum_{i=1}^{t} \sum_{A_{cv}^i \in \vec{\mathcal{A}}^i} \mathfrak{P}_\Theta(A_{cv}^i) \ ,
\end{aligned}
$$

factoring out further yields

$$
\begin{aligned}
\log\left(P_{\vec{\Theta}}(\vec{d})\right) &= \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \cdot \sum_{X_k^0 \in \vec{X}^0} \mathfrak{P}_\Theta(X_k^0) \\
&+ \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \cdot \sum_{X_k^i \in \vec{X}^{\,i}} \mathfrak{P}_\Theta(X_k^i | \vec{X}^{i^{\mathsf{T}}} \backslash X_k^i, \vec{A}_k^{i^{\mathsf{T}}}, X_k^{i-1}) \\
&+ \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \cdot \sum_{A_{cv}^i \in \vec{\mathcal{A}}^i} \mathfrak{P}_\Theta(A_{cv}^i) \ ,
\end{aligned}
$$

where one is able to explicitly represent the summation over all possible instantiations of X_λ^i by

$$
\begin{aligned}
\log\left(P_{\vec{\Theta}}(\vec{d})\right) &= \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \cdot \sum_{X_k^0 \in \vec{X}^0} \mathfrak{P}_\Theta(X_k^0) \\
&+ \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t} \backslash X_\lambda^i} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, {+x}_\lambda^i, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \\
&\cdot \left(\sum_{X_k^i \in \vec{X}^{\,i} \backslash X_\lambda^i} \mathfrak{P}_\Theta(X_k^i | \vec{X}^{i^{\mathsf{T}}} \backslash X_k^i, \vec{A}_k^{i^{\mathsf{T}}}, X_k^{i-1}) + \mathfrak{P}_\Theta({+x}_\lambda^i | \vec{X}^{i^{\mathsf{T}}} \backslash X_\lambda^i, \vec{A}_\lambda^{i^{\mathsf{T}}}, X_\lambda^{i-1}) \right) \\
&+ \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t} \backslash X_\lambda^i} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \neg x_\lambda^i, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \\
&\cdot \left(\sum_{X_k^i \in \vec{X}^{\,i} \backslash X_\lambda^i} \mathfrak{P}_\Theta(X_k^i | \vec{X}^{i^{\mathsf{T}}} \backslash X_k^i, \vec{A}_k^{i^{\mathsf{T}}}, X_k^{i-1}) + \mathfrak{P}_\Theta(\neg x_\lambda^i | \vec{X}^{i^{\mathsf{T}}} \backslash X_\lambda^i, \vec{A}_\lambda^{i^{\mathsf{T}}}, X_\lambda^{i-1}) \right) \\
&+ \sum_{\vec{\zeta}^{0:t^{\mathsf{T}}}} \sum_{\vec{\beta}^{1:t}} \sum_{i=1}^{t} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t}) \cdot \sum_{A_{cv}^i \in \vec{\mathcal{A}}^i} \mathfrak{P}_\Theta(A_{cv}^i) \ ,
\end{aligned}
$$

Note that an instantiation of X_λ^i might be part of $\vec{d}^{0:t}$ and therefore was not present in the previous summation, i.e., was not included in $\vec{\zeta}^{0:t^{\mathsf{T}}}$. Nevertheless, the extraction remains sound as a respective $P_{\vec{\vartheta}}(\vec{X}^{0:t^{\mathsf{T}}}, x_\lambda^i, \vec{\mathcal{A}}^{1:t^{\mathsf{T}}} | \vec{d}^{0:t})$ is then 0.

With $\gamma(X_\lambda = x_\lambda)$ as a shorthand for

$$\gamma(X_\lambda = x_\lambda) = \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t} \setminus X_\lambda^i} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\intercal}}, x_\lambda^i, \vec{\mathcal{A}}^{1:t^{\intercal}} | \vec{d}^{0:t})$$
$$\cdot \Big(\sum_{X_k^i \in \vec{X}^{\,i} \setminus X_\lambda^i} \mathfrak{P}_\Theta(X_k^i | \vec{X}^{i^{\intercal}} \setminus X_k^i, \vec{A}_k^{i^{\intercal}}, X_k^{i-1}) + \mathfrak{P}_\Theta(x_\lambda^i | \vec{X}^{i^{\intercal}} \setminus X_\lambda^i, \vec{A}_\lambda^{i^{\intercal}}, X_\lambda^{i-1}) \Big) \ .$$

one obtains

$$\log\Big(P_{\vec{\Theta}}(\vec{d})\Big) = \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}} | \vec{d}^{0:t}) \cdot \sum_{X_k^0 \in \vec{X}^0} \mathfrak{P}_\Theta(X_k^0)$$
$$+\gamma(X_\lambda = {}^{+}x_\lambda) + \gamma(X_\lambda = \neg x_\lambda) + \sum_{\vec{\zeta}^{0:t}} \sum_{\vec{\beta}^{1:t}} \sum_{i=1}^{t} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\intercal}}, \vec{\mathcal{A}}^{1:t^{\intercal}} | \vec{d}^{0:t}) \cdot \sum_{A_{cv}^i \in \vec{\mathcal{A}}^i} \mathfrak{P}_\Theta(A_{cv}^i) \ .$$

Classically, to learn a specific parameter, i.e., to find an optimized parameter $P_\Theta(x_\lambda^i | \vec{x}^{i^{\intercal}} \setminus x_\lambda^i, \vec{a}_\lambda^{i^{\intercal}}, x_\lambda^{i-1}) \in \vec{\Theta}^*$ it must be explicitly extracted from all (nested) summations s.t. one is able to solve Eq. C.3 by a partial derivation. By extracting this parameter from all summations one obtains,

$$\begin{aligned}
&\gamma(X_\lambda = x_\lambda) \\
&= \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t} \setminus X_\lambda^i} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\intercal}}, x_\lambda^i, \vec{\mathcal{A}}^{1:t^{\intercal}} | \vec{d}^{0:t}) \sum_{X_k^i \in \vec{X}^{\,i} \setminus X_\lambda^i} \mathfrak{P}_\Theta(X_k^i | \vec{X}^{i^{\intercal}} \setminus X_k^i, \vec{A}_k^{i^{\intercal}}, X_k^{i-1}) \\
&\quad + \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t} \vec{\beta}^{1:t} \setminus \vec{x}^{i^{\intercal}}, \vec{a}_\lambda^i, x_\lambda^{i-1}} P_{\vec{\vartheta}}(\vec{X}^{0:t^{\intercal}}, x_\lambda^i, \vec{\mathcal{A}}^{1:t^{\intercal}} | \vec{d}^{0:t}) \mathfrak{P}_\Theta(x_\lambda^i | \vec{X}^{i^{\intercal}} \setminus X_\lambda^i, \vec{A}_\lambda^{i^{\intercal}}, X_\lambda^{i-1}) \\
&\qquad + \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:i-2}} \sum_{\vec{\beta}^{1:i-2}} \sum_{\vec{\zeta}^{i+1:t}} \sum_{\vec{\beta}^{i+1:t}} \sum_{\vec{\zeta}^{i-1} \setminus X_\lambda^{i-1}} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^{i} \setminus \vec{A}_\lambda^i} \\
&P_{\vec{\vartheta}}(\vec{X}^{0:i-2^{\intercal}}, \vec{X}^{i-1^{\intercal}} \setminus X_\lambda^{i-1}, x_\lambda^{i-1}, \vec{\mathcal{A}}^{1:i-1^{\intercal}}, \vec{x}^i, \vec{a}_\lambda^i, \vec{\mathcal{A}}^{i^{\intercal}} \setminus \vec{A}_\lambda^i, \vec{X}^{i+1:t^{\intercal}}, \vec{\mathcal{A}}^{i+1:t^{\intercal}} | \vec{d}^{0:t}) \\
&\qquad\qquad \cdot \mathfrak{P}_\Theta(x_\lambda^i | \vec{x}^{i^{\intercal}} \setminus x_\lambda^i, \vec{a}_\lambda^{i^{\intercal}}, x_\lambda^{i-1}) \ , \qquad \text{(C.5)}
\end{aligned}$$

where $\sum_{\vec{\zeta}^{0:t} \vec{\beta}^{1:t} \setminus \vec{x}^i, \vec{a}_\lambda^i, x_\lambda^{i-1}}$ represents the summation over all possible instantiations of all unobserved variables excluding a specific instantiation $\vec{x}^i, \vec{a}_\lambda^i, x_\lambda^{i-1}$. However, Equation C.5 extracts a too general parameter $P_\Theta(x_\lambda^i | \vec{x}^{i^{\intercal}} \setminus x_\lambda^i, \vec{a}_\lambda^i, x_\lambda^{i-1})$ and does not consider activator criteria, which are needed in order to assure well-definedness in the transformation towards Equation C.4.

To learn an activation criteria aware parameter $P_\Theta(x^i_\lambda|\vec{x}^{i^\mathsf{T}}\backslash x^i_\lambda, \vec{a}^{i^\mathsf{T}}_\lambda, x^{i-1}_\lambda)$, we use Definition 3.2 and extract the parameter from all summations as

$$
\begin{aligned}
&\gamma(X_\lambda = x_\lambda)\\
&= \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t}\backslash X^i_\lambda} \sum_{\vec{\beta}^{1:t}} P_{\vec{\vartheta}}(\vec{X}^{0:t^\mathsf{T}}, x^i_\lambda, \vec{\mathcal{A}}^{1:t^\mathsf{T}}|\vec{d}^{0:t}) \sum_{X^i_k \in \vec{X}^{\,i}\backslash X^i_\lambda} \mathfrak{P}_\Theta(X^i_k|\vec{X}^{i^\mathsf{T}}\backslash X^i_k, \vec{A}^{i^\mathsf{T}}_k, X^{i-1}_k)\\
&\quad + \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:t}\vec{\beta}^{1:t}\backslash x^i_\lambda, \vec{x}^i_{\blacksquare\lambda}, \vec{a}^i_\lambda, x^{i-1}_\lambda} P_{\vec{\vartheta}}(\vec{X}^{0:t^\mathsf{T}}, x^i_\lambda, \vec{\mathcal{A}}^{1:t^\mathsf{T}}|\vec{d}^{0:t}) \mathfrak{P}_\Theta(x^i_\lambda|\vec{X}^{i^\mathsf{T}}\backslash X^i_\lambda, \vec{A}^{i^\mathsf{T}}_\lambda, X^{i-1}_\lambda)\\
&\qquad + \sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:i-2}} \sum_{\vec{\beta}^{1:i-2}} \sum_{\vec{\zeta}^{i+1:t}} \sum_{\vec{\beta}^{i+1:t}} \sum_{\vec{\zeta}^{i-1}\backslash X^{i-1}_\lambda} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^{i}\backslash \vec{A}^i_\lambda} \sum_{\vec{X}^i_{\square\lambda}}\\
&\qquad P_{\vec{\vartheta}}(\vec{X}^{0:i-2^\mathsf{T}}, \vec{X}^{i-1^\mathsf{T}}\backslash X^{i-1}_\lambda, x^{i-1}_\lambda, \vec{\mathcal{A}}^{1:i-1^\mathsf{T}}, \vec{X}^{i^\mathsf{T}}_{\square\lambda}, \vec{x}^{i^\mathsf{T}}_{\blacksquare\lambda},\\
&\qquad\quad \vec{a}^i_\lambda, \vec{\mathcal{A}}^{i^\mathsf{T}}\backslash \vec{A}^i_\lambda, \vec{X}^{i+1:t^\mathsf{T}}, \vec{\mathcal{A}}^{i+1:t^\mathsf{T}}|\vec{d}^{0:t}) \mathfrak{P}_\Theta(x^i_\lambda|\vec{x}^{i^\mathsf{T}}\backslash x^i_\lambda, \vec{a}^{i^\mathsf{T}}_\lambda, x^{i-1}_\lambda)\ .
\end{aligned}
$$

Then, an optimized parameter $P^*_{\vec{\vartheta}}(x^i_\lambda|\vec{x}^{i^\mathsf{T}}\backslash x^i_\lambda, \vec{a}^{i^\mathsf{T}}_\lambda, x^{i-1}_\lambda) \in \vec{\vartheta}^*$ according to Eq. 3.2 is explicitly obtained by partial derivation as

$$
\frac{\delta P_{\vec{\Theta}}(\vec{d})}{\delta P_\Theta(x^i_\lambda|\vec{x}^{i^\mathsf{T}}\backslash x^i_\lambda, \vec{a}^{i^\mathsf{T}}_\lambda, x^{i-1}_\lambda)} \Big(P^*_{\vec{\vartheta}}(x^i_\lambda|\vec{x}^{i^\mathsf{T}}\backslash x^i_\lambda, \vec{a}^{i^\mathsf{T}}_\lambda, x^{i-1}_\lambda)\Big) = 0\ ,
$$

which, under a stationary process reduces to

$$
P^*_{\vec{\vartheta}}(x^i_\lambda|\vec{x}^{i^\mathsf{T}}\backslash x^i_\lambda, \vec{a}^{i^\mathsf{T}}_\lambda, x^{i-1}_\lambda) = \frac{\gamma'(X_\lambda = x_\lambda)}{\gamma'(X_\lambda = {}^{+}x_\lambda) + \gamma'(X_\lambda = \neg x_\lambda)}\ ,
$$

with

$$
\begin{aligned}
\gamma'(X_\lambda = x_\lambda) = &\sum_{i=1}^{t} \sum_{\vec{\zeta}^{0:i-2}} \sum_{\vec{\beta}^{1:i-2}} \sum_{\vec{\zeta}^{i+1:t}} \sum_{\vec{\beta}^{i+1:t}} \sum_{\vec{\zeta}^{i-1}\backslash X^{i-1}_\lambda} \sum_{\vec{\beta}^{i-1}} \sum_{\vec{\beta}^{i}\backslash \vec{A}^i_\lambda} \sum_{\vec{X}^i_{\square\lambda}}\\
&P_{\vec{\vartheta}}(\vec{X}^{0:i-2^\mathsf{T}}, \vec{X}^{i-1^\mathsf{T}}\backslash X^{i-1}_\lambda, x^{i-1}_\lambda, \vec{\mathcal{A}}^{1:i-1^\mathsf{T}}, \vec{X}^{i^\mathsf{T}}_{\square\lambda}, \vec{x}^{i^\mathsf{T}}_{\blacksquare\lambda},\\
&\qquad \vec{a}^i_\lambda, \vec{\mathcal{A}}^{i^\mathsf{T}}\backslash \vec{A}^i_\lambda, \vec{X}^{i+1:t^\mathsf{T}}, \vec{\mathcal{A}}^{i+1:t^\mathsf{T}}|\vec{d}^{0:t})\ .
\end{aligned}
$$

in which an extended smoothing problem $\texttt{ExtdSP}(B_0, B_\rightarrow, \vec{z}^{0:t}, \vec{\mathfrak{b}}^{1:t}, t)$ is evident and can be written as in Theorem 3.1. Thus, the proposed procedure generates parameter instantiations $\vec{\vartheta}^*$ of $\vec{\Theta}$ that maximize the likelihood of the dataset. ■

The proof for activator parameters is equivalent.

References

Acea SpA. (2016). *The Acea Group.* Retrieved 06.05.2016, from `http://www.acea.it/section.aspx/en/acea_spa`

Acid, S., & de Campos, L. M. (2003). Searching for Bayesian Network Structures in the Space of Restricted Acyclic Partially Directed Graphs. *Journal of Artificial Intelligence Research*, *18*, 445–490.

Albanese, M., Jajodia, S., Jhawar, R., & Piuri, V. (2013). Reliable Mission Deployment in Vulnerable Distributed Systems. In *DSN 2013: 43rd IEEE/IFIP International Conference on Dependable Systems and Networks Workshop, Budapest, Hungary, June 24-27, 2013* (pp. 1–8).

Antonucci, A. (2011). The Imprecise Noisy-OR Gate. In *FUSION 2011: 14th International Conference on Information Fusion, Chicago, Illinois, USA, July 5-8, 2011* (pp. 1–7).

Arulampalam, M. S., Maskell, S., Gordon, N. J., & Clapp, T. (2002). A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, *50*(2), 174–188.

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An Open Source Software for Exploring and Manipulating Networks. In *ICWSM 2009: 3rd International Conference on Weblogs and Social Media, San Jose, California, USA, May 17-20, 2009.*

Beal, M. J., & Ghahramani, Z. (2003). The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures. *Bayesian Statistics*, *7*, 453–464.

Bilmes, J. A. (2000). Dynamic Bayesian Multinets. In *UAI 2000: 16th Conference on Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000* (pp. 38–45).

Boutilier, C., Friedman, N., Goldszmidt, M., & Koller, D. (1996). Context-Specific Independence in Bayesian Networks. In *UAI 1996: 12th Conference on Uncertainty in Artificial Intelligence, Reed College, Portland, Oregon, USA, August 1-4, 1996* (pp. 115–123).

Buckshaw, D. L., Parnell, G. S., Unkenholz, W. L., Parks, D. L., Wallner, J. M., & Saydjari, O. S. (2005). Mission Oriented Risk and Design Analysis of Critical Information Systems. *Military Operations Research*, *10*(2), 19–38.

Chung, C., Khatkar, P., Xing, T., Lee, J., & Huang, D. (2013). NICE: Network Intrusion Detection and Countermeasure Selection in Virtual Network Systems. *IEEE Trans. Dependable Sec. Comput.*, *10*(4), 198–211.

Combs, G., & The Wireshark Foundation. (2016). *Wireshark.* Retrieved 11.10.2016, from `https://www.wireshark.org/`

Cooper, G. F. (1990). The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks. *Artificial Intelligence*, *42*(2-3), 393–405.

Cozman, F. G. (2004). Axiomatizing Noisy-OR. In *ECAI 2004: 16th Eureopean Conference on Artificial Intelligence, including PAIS'2004: Prestigious Applicants [sic] of Intelligent Systems, Valencia, Spain, August 22-27, 2004* (pp. 979–980).

D'Amico, A., Buchanan, L., Goodall, J., & Walczak, P. (2010). Mission Impact of Cyber Events: Scenarios and Ontology to Express the Relationships between Cyber Assets, Missions, and Users. In *ICIW 2010: 5th International Conference on Information Warfare and Security, Wright-Patterson Air Force Base, Ohio, USA, April 8-9, 2010* (pp. 8–9).

de Barros Barreto, A., da Costa, P. C. G., & Yano, E. T. (2013). Using a Semantic Approach to Cyber Impact Assessment. In *STIDS 2013: 8th Conference on Semantic Technologies for Intelligence, Defense, and Security, Fairfax, Virginia, USA, November 12-15, 2013* (pp. 101–108).

de Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *IJCAI 2007: 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007* (pp. 2462–2467).

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., … Zhang, W. (2014). Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *KDD 2014: 20th ACM International Conference on Knowledge Discovery and Data Mining, New York, New York, USA, August 24-27, 2014* (pp. 601–610).

Doucet, A., de Freitas, N., Murphy, K. P., & Russell, S. J. (2000). Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *UAI 2000: 16th Conference on Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000* (pp. 176–183).

Doucet, A., & Johansen, A. M. (2009). A Tutorial on Particle Filtering and Smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, *12*, 656–704.

Drton, M. (2009). Discrete Chain Graph Models. *Bernoulli*, *15*(3), 736–753.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., … Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, *31*(3), 59–79.

Fierens, D., den Broeck, G. V., Renkens, J., Shterionov, D. S., Gutmann, B., Thon, I., … de Raedt, L. (2015). Inference and Learning in Probabilistic Logic Programs Using Weighted Boolean Formulas. *Theory and Practice of Logic Programming*, *15*(3), 358–401.

Friedman, N. (1997). Learning Belief Networks in the Presence of Missing Values and Hidden Variables. In *ICML 1997: 14th International Conference on Machine Learning, Nashville, Tennessee, USA, July 8-12, 1997* (pp. 125–133).

Friedman, N. (1998). The Bayesian Structural EM Algorithm. In *UAI 1998: 14th Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998* (pp. 129–138).

Friedman, N., & Koller, D. (2003). Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks. *Machine Learning*, *50*(1-2), 95–125.

Friedman, N., Murphy, K. P., & Russell, S. J. (1998). Learning the Structure of Dynamic Probabilistic Networks. In *UAI 1998: 14th Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998* (pp. 139–147).

Geiger, D., & Heckerman, D. (1996). Knowledge Representation and Inference in Similarity Networks and Bayesian Multinets. *Artificial Intelligence*, *82*(1-2), 45–74.

Ghahramani, Z. (1997). Learning Dynamic Bayesian Networks. In *Adaptive Processing of Sequences and Data Structures, International Summer School on Neural Networks, E.R. Caianiello, Vietri sul Mare, Salerno, Italy, September 6-13, 1997, Tutorial Lectures* (pp. 168–197).

Ghahramani, Z. (2001). An Introduction to Hidden Markov Models and Bayesian Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, *15*(1), 9–42.

Gibbs, A. L., & Su, F. E. (2002). On Choosing and Bounding Probability Metrics. *International Statistical Review*, *70*(3), 419–435.

Glesner, S., & Koller, D. (1995). Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases. In *ECSQARU 1995: Symbolic and Quantitative Approaches to Reasoning and Uncertainty, European Conference, Fribourg, Switzerland, July 3-5, 1995* (pp. 217–226).

Gonzalez Granadillo, G., Alvarez, E., Motzek, A., Merialdo, M., García-Alfaro, J., & Debar, H. (2016). Towards an Automated and Dynamic Risk Management Response System. In *NORDSEC 2016: 21st Nordic Conference on Secure IT Systems, Oulu, Finland, November 2-4, 2016* (pp. 37–53).

Goodall, J. R., D'Amico, A., & Kopylec, J. K. (2009). Camus: Automatically Mapping Cyber Assets to Missions and Users. In *MILCOM 2009: IEEE Military Communications Conference, Boston, Massachusetts, USA, October 18-21, 2009* (pp. 1–7).

Granadillo, G. G., Motzek, A., Garcia-Alfaro, J., & Debar, H. (2016). Selection of Mitigation Actions Based on Financial and Operational Impact Assessments. In *ARES 2016: 11th International Conference on Availability, Reliability and Security, Salzburg, Austria, August 31 - September 2, 2016* (pp. 137–146).

Haddawy, P., Helwig, J., Ngo, L., & Krieger, R. (1995). Clinical Simulation using Context-Sensitive Temporal Probability Models. In *Symposium on Computer Applications in Medical Care* (Vol. 1, pp. 203–207).

Heckerman, D., & Breese, J. S. (1996). Causal Independence for Probability Assessment and Inference Using Bayesian Networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, *26*(6), 826–831.

Henrion, M. (1988). Practical Issues in Constructing a Bayes Belief Network. *International Journal of Approximate Reasoning*, *2*(3), 337.

Hol, J. D., Schön, T. B., & Gustafsson, F. (2006). On Resampling Algorithms for Particle Filters. In *NSSPW 2006: IEEE Nonlinear Statistical Signal Processing Workshop, Cambridge, UK, September 13-15, 2006* (pp. 79–82).

Jaeger, M. (2001). Complex Probabilistic Modeling with Recursive Relational Bayesian Networks. *Annals of Mathematics and Artificial Intelligence*, *32*(1-4), 179–220.

Jahnke, M., Thul, C., & Martini, P. (2007). Graph based Metrics for Intrusion Response Measures in Computer Networks. In *LCN 2007: 32nd Annual IEEE Conference on Local Computer Networks, Clontarf Castle, Dublin, Ireland, October 15-18, 2007* (pp. 1035–1042).

Jakobson, G. (2011). Mission Cyber Security Situation Assessment using Impact Dependency Graphs. In *FUSION 2011: 14th International Conference on Information Fusion, Chicago, Illinois, USA, July 5-8, 2011* (pp. 1–8).

Jha, S., Sheyner, O., & Wing, J. (2002). Two Formal Analyses of Attack Graphs. In *CSFW 2002: 15th IEEE Workshop on Computer Security Foundations, Cape Breton, Nova Scotia, Canada, June 24-26, 2002* (pp. 49–63).

Kheir, N., Debar, H., Cuppens-Boulahia, N., Cuppens, F., & Viinikka, J. (2009). Cost Evaluation for Intrusion Response Using Dependency Graphs. In *N2S 2009: International Conference on Network and Service Security, Paris, France, June 24-26, 2009* (pp. 1–6).

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models - Principles and Techniques.* MIT Press.

Langner, R. (2013). *To Kill a Centrifuge. A Technical Analysis of What Stuxnet's Creators Tried to Achieve* (Tech. Rep.).

Levesque, H. J., Davis, E., & Morgenstern, L. (2012). The Winograd Schema Challenge. In *KR 2012: 13th International Conference on Principles of Knowledge Representation and Reasoning, Rome, Italy, June 10-14, 2012.*

Liu, Y., & Man, H. (2005). Network Vulnerability Assessment Using Bayesian Networks. In *SPIE Vol. 5812: Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security, Orlando, Florida, USA, March 28, 2005* (pp. 61–71).

Massey, B. (2008). Fast Perfect Weighted Resampling. In *ICASSP 2008: IEEE International Conference on Acoustics, Speech, and Signal Processing, Caesars Palace, Las Vegas, Nevada, USA, March 30 - April 4, 2008* (pp. 3457–3460).

Milch, B., Marthi, B., Sontag, D., Russell, S. J., Ong, D. L., & Kolobov, A. (2005). Approximate Inference for Infinite Contingent Bayesian Networks. In *AISTATS 2005: 10th International Workshop on Artificial Intelligence and Statistics, Bridgetown, Barbados, January 6-8, 2005.*

Motzek, A., Geick, C., & Möller, R. (2016). Semantic Normalization and Merging of Business Dependency Models. In *CBI 2016: 18th IEEE Conference on Business Informatics, Paris, France, August 29 - September 1, 2016* (pp. 7–15).

Motzek, A., & Möller, R. (2015a). Exploiting Innocuousness in Bayesian Networks. In *AI 2015: 28th Australasian Joint Conference on Artificial Intelligence, Canberra, ACT, Australia, November 30 - December 4, 2015* (pp. 411–423).

Motzek, A., & Möller, R. (2015b). Indirect Causes in Dynamic Bayesian Networks Revisited. In *IJCAI 2015: 24th International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, July 25-31, 2015* (pp. 703–709).

Motzek, A., & Möller, R. (2016). Probabilistic Mission Defense and Assurance. In *NATO IST-148: Symposium on Cyber Defence Situation Awareness, STO-MP-IST-148, Sofia, Bulgaria, October 3-4, 2016* (pp. 4-1–4-18). STO. doi: 10.14339/STO-MP-IST-148

Motzek, A., & Möller, R. (2017). Context- and Bias-Free Probabilistic Mission Impact Assessment. *Computers & Security*, *65*, 166–186.

Motzek, A., Möller, R., Lange, M., & Dubus, S. (2015). Probabilistic Mission Impact Assessment based on Widespread Local Events. In *NATO IST-128 Workshop: Assessing Mission Impact of Cyberattacks, NATO IST-128 Workshop, Istanbul, Turkey, June 15-17, 2015* (pp. 16–22).

Murphy, K. P. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning* (Unpublished doctoral dissertation). University of California, Berkeley.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* Cambridge, MA: The MIT Press.

Musman, S., Temin, A., Tanner, M., Fox, D., & Pridemore, B. (2010). Evaluating the Impact of Cyber Attacks on Missions. In *ICIW 2010: 5th International Conference on Information Warfare and Security, Wright-Patterson Air Force Base, Ohio, USA, April 8-9, 2010* (pp. 446–456).

Ngo, L., & Haddawy, P. (1997). Answering Queries from Context-Sensitive Probabilistic Knowledge Bases. *Theoretical Computer Science*, *171*(1-2), 147–177.

Ngo, L., Haddawy, P., & Helwig, J. (1995). A Theoretical Framework for Context-Sensitive Temporal Probability Model Construction with Application to Plan Projection. In *UAI 1995: 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, Canada, August 18-20, 1995* (pp. 419–426).

Nodelman, U., Shelton, C. R., & Koller, D. (2002). Continuous Time Bayesian Networks. In *UAI 2002: 18th Conference on Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002* (pp. 378–387).

Nodelman, U., Shelton, C. R., & Koller, D. (2003). Learning Continuous Time Bayesian Networks. In *UAI 2003: 19th Conference on Uncertainty in Artificial Intelligence, Acapulco, Mexico, August 7-10, 2003* (pp. 451–458).

Ou, X., Govindavajhala, S., & Appel, A. W. (2005). MulVAL: A Logic-based Network Security Analyzer. In *14th USENIX Security Symposium, Baltimore, Maryland, USA, July 31 - August 5, 2005.*

Panoptesec DOW. (2013). Panoptesec Annex I, Description Of Work. In *Project Deliverables of the Panoptesec Collaborative Research Project on Dynamic Risk Approaches for Automated Cyber Defence, Grant Agreement No: 610416, ICT-2013.1.5, Trustworthy ICT, Version 4th September, 2015.*

Pearl, J. (2002). Reasoning with Cause and Effect. *AI Magazine*, *23*(1), 95–112.

Pearl, J. (2009). *Causality: Models, Reasoning and Inference* (2nd ed.). New York, New York, USA: Cambridge University Press.

Pearl, J., & Russell, S. (2003). Bayesian Networks. In M. A. Arbib (Ed.), *Handbook of Brain Theory and Neural Networks* (pp. 157–160). MIT Press.

Pfeffer, A., & Tai, T. (2005). Asynchronous Dynamic Bayesian Networks. In *UAI 2005: 21st Conference on Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005* (pp. 467–476).

Poole, D., & Zhang, N. L. (2003). Exploiting Contextual Independence In Probabilistic Inference. *Journal Of Artificial Intelligence Research*, *18*, 263–313.

Robinson, J. W., & Hartemink, A. J. (2008). Non-Stationary Dynamic Bayesian Networks. In *NIPS 2008: 22nd Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008* (pp. 1369–1376).

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence - A Modern Approach (3. internat. ed.).* Pearson Education.

Saleh, B., & Masseglia, F. (2008). Time Aware Mining of Itemsets. In *TIME 2008: 15th International Symposium on Temporal Representation and Reasoning, Université du Québec à Montréal, Canada, 16-18 June, 2008* (pp. 93–97).

Sanghai, S., Domingos, P., & Weld, D. (2005). Relational Dynamic Bayesian Networks. *Journal of Artificial Intelligence Research*, *24*, 759–797.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing* (No. 3). Cambridge University Press.

Sloane, N. J. A. (2015). *The On-Line Encyclopedia of Integer Sequences.* http://oeis.org/. OEIS Foundation Inc. (Sequences A003024 & A001831.)

Sommestad, T., & Hunstad, A. (2013). Intrusion Detection and the Role of the System Administrator. *Information Management & Computer Security*, *21*(1), 30-40.

Song, L., Kolar, M., & Xing, E. P. (2009). Time-Varying Dynamic Bayesian Networks. In *NIPS 2009: 23rd Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 7-10, 2009* (pp. 1732–1740).

Srinivas, S. (1993). A Generalization of the Noisy-Or Model. In *UAI 1993: 9th Conference on Uncertainty in Artificial Intelligence, The Catholic University of America, Providence, Washington, DC, USA, July 9-11, 1993* (pp. 208–218).

Torres-Toledano, J. G., & Sucar, L. E. (1998). Bayesian Networks for Reliability Analysis of Complex Systems. In *IBERAMIA 98: 6th Ibero-American Conference on AI, Lisbon, Portugal, October 5-9, 1998* (pp. 195–206).

Wang, L., Islam, T., Long, T., Singhal, A., & Jajodia, S. (2008). An Attack Graph-Based Probabilistic Security Metric. In *DBSec 2008: 22nd Annual IFIP WG 11.3 Conference on Data and Applications Security, London, UK, July 13-16, 2008* (pp. 283–296).

Xie, P., Li, J. H., Ou, X., Liu, P., & Levy, R. (2010). Using Bayesian Networks for Cyber Security Analysis. In *DSN 2010: 40th IEEE/IFIP International Conference on Dependable Systems and Networks 2010, Chicago, Illinois, USA, June 28 - July 1, 2010* (pp. 211–220).

Yoshida, R., Imoto, S., & Higuchi, T. (2005). Estimating Time-Dependent Gene Networks from Time Series Microarray Data by Dynamic Linear Models with Markov Switching. In *CSB 2005: 4th International IEEE Computer Society Computational Systems Bioinformatics Conference, Stanford, California, USA, August 8-11, 2005* (pp. 289–298).

Zagorecki, A., & Druzdzel, M. J. (2006). Probabilistic Independence of Causal Influences. In *PGM 2006: 3rd European Workshop on Probabilistic Graphical Models, Prague, Czech Republic, 12-15 September, 2006* (pp. 325–332).

Zhang, N. L., & Poole, D. L. (1996). Exploiting Causal Independence in Bayesian Network Inference. *Journal of Artificial Intelligence Research (JAIR)*, *5*, 301–328.