Variable Elimination for Forgetting in Probabilistic Graphical Models - A Work in Progress

Tanya Braun[1,*], Marcel Gehrke[2], Kai Sauerwald[3] and Jan Speller[1]

[1]University of Münster, Münster, Germany
[2]University of Hamburg, Hamburg, Germany
[3]University of Hagen, Hagen, Germany

Abstract

In this article, we take the operators that make up variable elimination in probabilistic graphical models and reinterpret them as forgetting operators in the vein of marginalisation for forgetting in logic. As such, we define the inference task of forgetting in probabilistic graphical models and provide operators to solve the task, discussing effects on complexity versus variable elimination for probabilistic inference. In addition, we consider different properties for forgetting in probabilistic graphical models, arguing whether variable elimination as forgetting fulfils them. We also discuss the limits of forgetting using variable elimination as well as other dimensions and approaches to forgetting in probabilistic graphical models.

Keywords

Probabilistic inference, variable elimination, marginalisation, forgetting, factor graphs

1. Introduction

Forgetting has been investigated in various areas of artificial intelligence [1], whereby forgetting is considered as a technical term referring to the removal of a meta symbol in a representation, or forgetting refers to the everyday understanding one observes in human [2, 3]. One of the ultimate aims of investigating forgetting is to reduce computational costs and hence, research on forgetting often focuses on techniques primarily to modify representations, e.g., [4, 5], so that tasks such as inference become easier. Another approach is to capture the notion of forgetting by principles [6, 7, 8]. The most prominent operation of forgetting in logic-based artificial intelligence is the elimination of a variable, which goes back to Boole [9, 10]. In propositional logic, for a formula φ, one defines syntactically elimination of a set of variables Γ as follows:

$$\begin{aligned} \text{propForget}(\varphi, \emptyset) &= \varphi \\ \text{propForget}(\varphi, \Gamma \cup \{a\}) &= \text{propForget}(\varphi, \Gamma)[a/\top] \vee \text{propForget}(\varphi, \Gamma)[a/\bot] \end{aligned}$$

whereby $\psi[a/\top]$ means replacing all occurrences of a by a tautology in ψ, respectively, $\psi[a/\bot]$ means replacing all occurrences of a by a contradictory formula in ψ.

A popular and powerful approach to capturing and understanding uncertainty in our world is to use a probability distribution over a set of random variables. Probabilistic models represent such probability distributions but have complexity that is exponential in the number of random variables. Probabilistic graphical models (PGMs) use (conditional) independences among the random variables to factorise

Joint proceedings of the Workshop on Theory and Methods for Abstraction (THEMA 2026) and the Workshop on Modularity and Splitting Techniques for Knowledge Representation and Reasoning (MoST 2026) at the 9th Federated Logic Conference (FLoC 2026), July 13–29, Lisbon, Portugal

*Corresponding author.
tanya.braun@uni-muenster.de (T. Braun); marcel.gehrke@uni-hamburg.de (M. Gehrke); kai.sauerwald@fernuni-hagen.de (K. Sauerwald); jan.speller@uni-muenster.de (J. Speller)
https://www.uni-muenster.de/Informatik.AGBraun/team/braun.shtml (T. Braun); https://www.uni-muenster.de/Informatik.AGBraun/team/braun.shtml (M. Gehrke); https://www.kai-sauerwald.de (K. Sauerwald); https://www.uni-muenster.de/Informatik.AGBraun/team/speller.shtml (J. Speller)
0000-0003-0282-4284 (T. Braun); 0000-0001-9056-7673 (M. Gehrke); 0000-0002-1551-7016 (K. Sauerwald); 0000-0003-1106-8177 (J. Speller)

© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073
published 2026-11-15

CEUR-WS.org/Vol-4331/paper3.pdf

a full joint into a set of factors – reducing the cost of representation in practice. There are several factorised models, such as Bayesian networks (BNs) [11], Markov networks [12], and factor graphs [13]. The inference problem of query answering then asks for the probability (distribution) of a set of random variables, possibly given a set of observations [14]. To solve the problem, marginalisation can be used to sum out those variables that do not occur in the query. For factorised models, this method has been formalised as variable elimination (VE) [15]. However, to the best of our knowledge, there is no established notion of forgetting in PGMs. Hence, in probabilistic inference, the notion of forgetting is not yet formally investigated, with the exception of a few works on probabilistic forms of belief change [16, 17] and close to none on forgetting.

An inference method shared between logic and probabilistic inference is that of marginalisation, which can be used as a method of forgetting in logic. In propositional logic, marginalisation can be characterized as the operation that eliminates all variables except of Γ [18]. Hence, one can implement the logical form of marginalisation to a set of variables Γ of a formula φ with variables $\mathrm{sig}(\varphi)$ via forgetting as follows:

$$\mathrm{propMarg}(\varphi, \Gamma) = \mathrm{propForget}(\varphi, \mathrm{sig}(\varphi) \setminus \Gamma)$$

In this paper, we employ the connection of marginalisation and forgetting in logic, to analogously define and investigate forgetting in PGMs. We are treating forgetting through VE, which has the upsides of looking at a well-understood method with known complexity results, enabling for transferring results to the new setting of forgetting in PGMs. The contributions of this paper are as follows:

- We define a set of properties to describe a forgetting method, namely, semantics preservation, order-independence, idempotence, and reversibility.
- We define the inference problem of forgetting in factorised models.
- We define two forgetting operators, one for forgetting a factor and one for forgetting a random variable based on VE operators, to solve the forgetting problem, showing their correctness and discussing their runtime complexity, their effect on the runtime complexity of the query answering problem, and the properties defined before.

We also discuss how other aspects of PGMs relate to VE-based forgetting. To the best of our knowledge, we are the first to formally treat forgetting in PGMs using VE.

The remainder of this article is structured as follows: We start with notations, defining factorised models and the query answering problem, before recapping VE as a solution method including the two main VE operators. Then, we look at use cases, list our assumptions, and define four properties for forgetting in PGMs. We follow up with our main contribution, forgetting in PGMs based on VE. We end with a discussion of other aspects to forgetting in PGMs.

2. Preliminaries

The section provides basic notations and recaps VE as the probabilistic inference algorithm, whose operators we use for forgetting.

2.1. Factor Graphs

In this subsection, we introduce factorised models, which encode a factorisation of a full joint probability distribution over a set of random variables based on (conditional) independences between random variables. We use an epidemic scenario as a running example, in which one may be *sick*, *travel*, and get *treated*, while there may be an *epidemic*. We use $\boldsymbol{Y}$ to denote sets and $\mathcal{Y}$ to denote sequences. In abuse of notation, we use set operations on sequences, which treat sequences like sets but preserve the order of the elements in them.

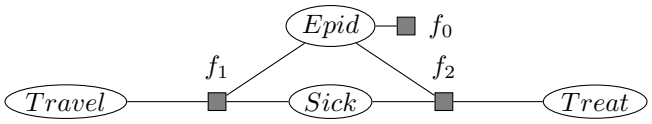


Figure 1: Factor graph for the example model F_{ex} of Example 1.

Definition 1 (Random variable, range, event)**.** *Let* $\mathbf{R}$ *be a finite set of random variable names. A* random variable *with name* $R \in \mathbf{R}$ *can be mapped to elements of a set of possible values* $\mathrm{ran}(R)$*, called the* range*. An* event $R = r$ *denotes the occurrence of random variable* R *with range value* $r \in \mathrm{ran}(R)$*. If* R *is clear from context, we write* r *for short.*

To relate random variables, we use factors. A factor consists of a sequence of random variables and a function, mapping argument values of the range of the random variables to real values, called *potentials*, of which at least one is non-zero.

Definition 2 (Factor, size, model)**.** *Let* Φ *be a finite set of factor names. We denote a* factor f *by* $\phi(\mathcal{R})$ *with* $\mathcal{R} = (R_1, \ldots, R_m)$ *a sequence of random variables and* $\phi : \times_{i=1}^{m} \mathrm{ran}(R_i) \to \mathbb{R}^+$ *a function with name* $\phi \in \Phi$*. The term* $\mathrm{rv}(Y)$ *refers to the random variables in some element* Y*, a factor or sets of factors. The* size *of a factor is given by* $|\mathrm{ran}(\mathcal{R})|$*. A set of factors forms a* model $F := \{f_i\}_{i=1}^{n}$*.*

Factor graphs are graphical representations of models of Def. 2 that explicitly represent random variables and factors as nodes.

Definition 3 (Factor graph)**.** *Given a model* F*, a* factor graph *is a graphical representation, which is a bipartite graph* (V, E) *with nodes* $V = \mathrm{rv}(F) \cup F$ *for random variables and factors, respectively, and edges* $E = \{\, \{R, f\} \mid R \in \mathrm{rv}(f), f \in F \,\}$*, i.e., the edges are between a factor and a variable whenever the variable occurs in the factor.*

The following shows an example model for the running example of an epidemic scenario.

Example 1. *Let* $\mathbf{R}$ *be* $\{Epid, Travel, Sick, Treat\}$*, which form Boolean random variables. Then, we build an example model* $F_{ex} = \{f_i\}_{i=0}^{2}$ *with*

$$f_0 = \phi_0(Epid),\ f_1 = \phi_1(Epid, Sick, Travel),\ f_2 = \phi_2(Epid, Sick, Treat).$$

We omit input-output mappings for brevity reasons. Figure 1 depicts F_{ex} *as a factor graph with four variable nodes and three factor nodes.*

The graph structure also encodes a set of (conditional) independences, where random variables are (conditionally) independent of each other given a set of random variables that blocks every path between two variables. The equivalent representation of a Markov network only uses nodes for random variables with edges between those random variables that occur in factors together, which makes the random variables form clusters in the graph. As such, if only given a Markov network without the factors, it is not clear if the underlying model has factors over maximal clusters, minimal clusters, or something in between. The semantics of a model is given by its full joint distribution:

Definition 4 (Semantics)**.** *With* $Z = \sum_{\boldsymbol{r} \in \mathrm{ran}(\mathrm{rv}(F))} \prod_{f \in F} \phi_f(\pi_{\mathrm{rv}(f)}(\boldsymbol{r}))$ *as the normalisation constant, a model* F *represents the full joint distribution* $P_F : \mathrm{ran}(\mathrm{rv}(F)) \to [0, 1]$ *with*

$$P_F(\boldsymbol{r}) = \frac{1}{Z} \prod_{f \in F} \phi_f(\pi_{\mathrm{rv}(f)}(\boldsymbol{r})).$$

We omit the F *in* P_F *if* F *is clear from the context.*

Given a model, the query answering problem poses the problem of computing the answer to a query for a probability distribution over a set of random variables on a given model.

Algorithm 1 Multiply Operator

function MULTIPLY(set of factors F', model F)
  Precondition $F' \subseteq F$
  while $|F'| > 1$ **do**
    Pick two factors $\phi'(\boldsymbol{R}'), \phi''(\boldsymbol{R}'') \in F'$
    for each $\boldsymbol{r} \in \mathrm{ran}(\boldsymbol{R}' \cup \boldsymbol{R}'')$ **do**
      $\phi(\boldsymbol{r}) \leftarrow \phi'(\pi_{\boldsymbol{R}'}(\boldsymbol{r})) \cdot \phi''(\pi_{\boldsymbol{R}''}(\boldsymbol{r}))$
    $F' \leftarrow F' \setminus \{\phi'(\boldsymbol{R}'), \phi''(\boldsymbol{R}'')\} \cup \{\phi(\boldsymbol{R})\}$
  return $f \mid f \in F'$
  Postcondition $P_F = P_{F \setminus F' \cup \{\phi(\boldsymbol{R})\}}$

Algorithm 2 Sum-out Operator

function SUM-OUT(Random variable R, factor f, model F)
  Precondition $f \in F$, $R \in \mathrm{rv}(f)$, $R \notin \mathrm{rv}(F \setminus \{f\})$
  for each $\boldsymbol{r} \in \mathrm{ran}(\mathrm{rv}(f) \setminus \{R\})$ **do**
    $\phi(\boldsymbol{r}) \leftarrow \sum_{r \in \mathrm{ran}(R)} \phi(r_1, \ldots, r_{i-1}, r, r_{i+1}, \ldots, r_n)$
  return $\phi(\boldsymbol{R})$
  Postcondition $P_{F \setminus \{f\} \cup \phi(\mathcal{R})} = \sum_{r \in \mathrm{ran}(R)} P_F(\boldsymbol{r}, r)$ with $\boldsymbol{r} \in \mathrm{ran}(\mathrm{rv}(F) \setminus \{R\})$

Algorithm 3 A pseudocode description of VE

function VE(Model F, query variables $\boldsymbol{Q}$)
  while $\mathrm{rv}(F) \setminus \boldsymbol{Q} \neq \emptyset$ **do**
    Pick next random variable $R \in \mathrm{rv}(F) \setminus \boldsymbol{Q}$ to eliminate
    $F' \leftarrow \{f \mid R \in \mathrm{rv}(f), f \in F\}$
    $f \leftarrow$ MULTIPLY(F', F)
    $f \leftarrow$ SUM-OUT(R, f, F)
    $F \leftarrow F \setminus F' \cup \{f\}$
  if $|F| > 1$ **then**
    $f \leftarrow$ MULTIPLY(F, F)
  Normalise f ▷ Divide each potential in f by the sum of all potentials in f
  return f

Definition 5 (Query, query answering problem)**.** *Given a model F, a set of events $\boldsymbol{e}$ called evidence, and a set of random variables $\boldsymbol{Q}$, the syntactic construct $P_F(\boldsymbol{Q} \mid \boldsymbol{e})$ denotes a* query *for the probability distribution of Q given $\boldsymbol{e}$ on P_F. The inference problem of* query answering *then refers to the problem of computing an answer to $P_F(\boldsymbol{Q} \mid \boldsymbol{e})$.*

Example 2. *For the example model F_{ex}, $P(Epid)$ would form a query for a marginal probability distribution. $P(Epid \mid Sick = true) = P(Epid \mid sick)$ forms a query for a conditional probability distribution, given evidence $sick \triangleq (Sick = true)$.*

2.2. Variable Elimination

VE is an inference algorithm for answering queries in PGMs based on marginalisation [15]. The idea is to eliminate all non-query variables from a model by summing over their range values. To do so in an efficient manner, the basic algebraic laws of commutativity and distributivity allow for organising the elimination order such that the interim results remain as small as possible, which can be formalised as the following two theorems [19]:

Theorem 1. *Given a factor $\phi(\mathcal{R}) = \phi(\mathcal{R}_1, \mathcal{R}_2)$ with $\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset$, then*

$$\sum_{\boldsymbol{r} \in \text{ran}(\mathcal{R})} \phi(\mathcal{R}) = \sum_{\boldsymbol{r}_1 \in \text{ran}(\mathcal{R}_1)} \sum_{\boldsymbol{r}_2 \in \text{ran}(\mathcal{R}_2)} \phi(\boldsymbol{r}_1, \boldsymbol{r}_2) = \sum_{\boldsymbol{r}_2 \in \text{ran}(\mathcal{R}_2)} \sum_{\boldsymbol{r}_1 \in \text{ran}(\mathcal{R}_1)} \phi(\boldsymbol{r}_1, \boldsymbol{r}_2)$$

Theorem 2. *Given two factors $\phi(\mathcal{R}), \phi(\mathcal{S}, \mathcal{T})$ with $\mathcal{R} \cap \mathcal{S} = \emptyset$, then*

$$\sum_{\boldsymbol{s} \in \text{ran}(\mathcal{S})} \phi(\mathcal{R})\phi(\boldsymbol{s}, \mathcal{T}) = \phi(\mathcal{R}) \sum_{\boldsymbol{s} \in \text{ran}(\mathcal{S})} \phi(\boldsymbol{s}, \mathcal{T})$$

Finding the optimal elimination order is an NP-complete problem [14]. Usually, one has to rely on heuristics to find a good enough approximation.

Note 1. *When defining algebraic operators to apply to a model F with full joint P_F, we require those operators to adhere to Theorems 1 and 2 to preserve P_F through their application, implying correctness. We call such operators* semantics-preserving.

To formalise VE, we introduce two operations, MULTIPLY and SUM-OUT, shown in Algs. 1 and 2. For the sum-out operation to be correct and as such, semantics-preserving, all factors that contain R need to be multiplied as multiplication comes before addition. VE then proceeds as follows (see Alg. 3 as well): While there are non-query variables, the next variable to eliminate is picked given a heuristics (or given a predetermined elimination order). Then, all factors containing the variable are multiplied into one factor, in which the variable to eliminate is then summed out. At the end, all remaining factors are multiplied into one factor, which is then normalised. The given description does not include evidence to keep it short and simple. Please refer to [19] for details regarding evidence absorption.[1]

Example 3. *Consider the query $P(Epid)$ from Example 2 in the example model F_{ex} from Example 1. VE needs to eliminate the variables $Treat$, $Travel$, and $Sick$. Pick $Treat$ as the first variable to eliminate. Since $Treat$ only occurs in f_2, VE does not need to multiply any factors and can proceed to sum out $Treat$ from f_2 by summing over its range values, resulting in a factor $\phi_2'(Epid, Sick)$. The same holds for $Travel$ in f_1, resulting in a factor $\phi_1'(Epid, Sick)$. To eliminate $Sick$, VE needs to multiply ϕ_1' and ϕ_2', before summing out $Sick$, yielding a factor $\phi(Epid)$. Together with f_0, there are two factors left, defined over $Epid$. As such, they need to be multiplied and then the result's entries normalised to form the probability distribution that is the answer to $P(Epid)$. Mathematically, the following calculations happened:*

$$P(Epid) \propto \phi(Epid) \sum_{s \in \text{ran}(Sick)} \sum_{tl \in \text{ran}(Travel)} \phi_1(Epid, s, tl) \sum_{tt \in \text{ran}(Treat)} \phi_2(Epid, s, tt)$$

To characterise the complexity of answering a query using VE, the largest intermediate result under an optimal elimination order is used, that is the smallest possible largest factor $\phi(R_1, \ldots, R_w)$ with a size of $\times_{i=1}^{w} |\text{ran}(R_i)|$ that occurs during sum-out operations, with w called *tree width*. The tree width is bounded from below by the maximum number of input arguments in a model but can become much larger if it is necessary to multiply many factors to sum out a next variable. The complexity is then given by $O(kr^w)$ with $k = |\text{rv}(F)|$, $r = \max_{R \in \text{rv}(F)} |\text{ran}(R)|$, and w the tree width [19].

Example 4. *Consider the answer computation in Example 3. The largest (interim) factor during the calculation has three inputs. Since $Treat$ and $Travel$ are independent from each other given $Sick$, their elimination can be handled independently from each other and could be done in any order without any effect on the largest interim result. However, choosing to eliminate $Sick$ first would require multiplying f_1 and f_2, leading to an interim factor with four inputs, illustrating the need for good elimination orders and its influence on runtime.*

Next, we look at certain use cases for forgetting in PGMs, the assumptions we thus make with the given setting of forgetting in PGMs as well as some properties that may characterise forgetting in PGMs before defining forgetting using VE operations.

[1] In essence, one could think of evidence as factors mapping the observed range value to 1 and the remaining values to 0, having them added to the model to be handled like the rest.

3. Use Cases, Assumptions & Properties

This section looks at certain use cases to illustrate the purpose of forgetting in PGMs, the assumptions that underlie our work, and some properties to characterise a solution method for forgetting in PGMs.

3.1. Use Cases

Forgetting can have applications in various scenarios to realise different requirements or tasks. We highlight a string of applications, on different levels of abstractions.

Changing Privacy Demands When learning a model from given data, the data is distilled into a more compact representation that (hopefully) encodes the essence of the data. Formally speaking, the idea is that there is a true distribution encoded in a model, of which the training data are samples, and learning encompasses to find a distribution that is close to the true distribution by finding one that makes the data most probable. Please note that the result might be a local optimum.

With changing demands regarding privacy, it might happen that certain features in the data should no longer be represented explicitly, while keeping the original relations between the remaining features intact. There is the possibility of relearning a model on the data set without the offending feature(s). Doing so has drawbacks: On the one hand, it costs time and requires to still have the training data around. On the other hand, the learned model might end up in a different local optimum and thus the relations might look a bit differently. As such, queries might have different answers than before. Thus, it would be more suitable to forget (remove) the features that should no longer be represented in the model while keeping the rest of the model as is.

Changing Monitoring Setups in Predictive Maintenance Predictive maintenance regards the task of predicting when a component might require maintenance, which usually involves a set of sensors for monitoring a component over time and a probabilistic model encoding normal behaviour over time. When sensors over time consistently show values that do not align well with normal behaviour, the probability for failures increases. There are then two possibilities: One, maintenance might be necessary for the component. Two, a sensor is faulty. While a faulty sensor might simply come back to life or be simply replaced, sensors might also be switched off. Keeping the sensor around in the model is inefficient as it has to be dealt with during query answering as part of the model. The more efficient way would be to forget (remove) the sensor in the model, while keeping the remaining relations as is.

Manuscript Analysis Manuscripts tell a story over time, and the focus changes with the story progressing. When distilling a PGM as a probabilistic representation of a text, the representation might need to change over the sequence of the text since components are no longer relevant, while the remainder of the model still holds true. To more closely represent such processes over the course of a manuscript, one could apply forgetting operations to transform the model accordingly to provide a sequence of snapshots.

Human-aware decision making In recent years, human-aware planning and decision making has become a hot topic, extending classical planning and decision making by adapting plans to human expectations of some form (see, e.g., [20] and the papers cited therein) or by providing explanations through so-called model reconciliation (see, e.g., [21] and the papers cited therein). In logic-based model reconciliation, forgetting can be employed to generate explanations based on abstraction [22]. For decision-theoretic PGMs, i.e., PGMs extended by decision nodes and utility functions [23], forgetting also provides the potential to move closer towards explanations for decision-theoretic PGMs.

3.2. Assumptions

We discuss the assumptions we make regarding the general setting of PGMs as well as forgetting in them, next to which components to forget specifically based on the chosen setting.

Probabilistic Setting By choosing to look at forgetting in PGMs, we make certain assumption about the general setting: We assume that the environment or system we are looking at is describable by a probability distribution over a set of discrete random variables. These random variables exhibit (conditional) independences among each other, yielding a factorisation of said distribution. We also assume that the model is stationary, meaning that the distribution and dependences themselves do not change over time through same forces inherent to the environment and not necessarily encoded by the model, as well as static, meaning that the state of the system does not depend on time. In Section 5, we briefly consider temporal PGMs, where the system state changes from one time step to the next, describable by a (factorised) probability distribution.[2]

Forgetting in General We treat forgetting as the technical notion of deleting a syntactical component from a model while keeping the semantics of the model unaffected, which enables to formally define a forgetting operation, which can then fulfil certain properties on a technical level. Whether these properties are desirable may depend on a specific application scenario, and whether they are something with an equivalent notion in human forgetting may be up for discussion. With forgetting defined through respective operations, forgetting does not inherently evolve within the model but rather occurs through the application of the operation, which is externally triggered by a user. As we assume the model to be otherwise stationary, this is consistent with our assumptions to have a change in the model to be triggered from an outside source and not be part of the modelling itself. Of course considering the model before and after a forgetting operation, one could argue that the model is no longer static. However, for true non-static modelling, we would need to model the process of forgetting using a probabilistic model, which opens up a whole new world of research and lies therefore outside the scope of this article.

Components to Forget By choosing the above framing, we can look at different syntactical components to delete from a PGM, which includes a random variable or factor if focussing on the factor representation. If considering a factor graph, then edges also become relevant. Since we specifically look at VE to implement forgetting due to its connection to forgetting in logic via marginalisation, we restrict ourselves to forgetting factors and random variables as they can be dealt with naturally using the existing VE operators. Section 5 discusses further components like edges, but also (in)dependences among random variables.

3.3. Properties

We consider the following properties to characterise methods of forgetting: (i) semantics preservation, i.e., having equivalent semantics while forgetting syntactic representation, (ii) order-independence, i.e., interchanging the order of forgetting multiple things not changing the outcome, (iii) idempotence, i.e., forgetting something more than once not changing the outcome, and (iv) reversibility, i.e., being able to restore the same state as before forgetting. These properties are not necessarily desirable in all cases but allow for characterising different solution methods and comparing them against each other.

The first property is the most important one regarding forgetting, as it makes up the main characteristic of forgetting. It focuses on the semantics of a model, which must be preserved through any forgetting operation. The remaining properties focus on syntax: The next two properties are of technical nature. Whereas order-independence might not be something prevalent to human forgetting due to its hierarchical nature, it might be a property that a forgetting operations exhibits. In contrast, not being

[2] If the probability distribution describing the state change remains the same over time, a temporal model is considered to be stationary.

able to forget something after it has already been forgotten is very much part of human forgetting, although it might be realised through not having an effect on the technical side. The last property may not be something that one is interested in, given a specific application where the goal might be to not be able to reverse engineer the forgotten syntactic representation. However, when thinking about forgetting, thinking about remembering comes naturally, which leads to a notion of reversibility, which is why we include it in our deliberations here.

Next, we formalise the properties for models F of Def. 2 with semantics P_F of Def. 4 using $\textsc{forget}(\xi, F)$ to denote a forgetting operation, forgetting ξ in F with ξ being a component of F, which changes F into a model F', in which ξ is no longer syntactically present.

Semantics Preservation The main property characterising forgetting concerns the central idea that forgetting does not change the semantics per se, only the syntactical representation, meaning that any forgetting operation must be semantics-preserving. That is, the full joint resulting from forgetting something in a model F must be equivalent to the full joint P_F, which may be transformed using semantics-preserving operations. We formalise the property as follows using $\equiv$ to denote equivalence:

$$P_F \equiv P_{\textsc{forget}(\xi,F)} \tag{1}$$

A consequence of semantics preservation is that all possible queries that do not mention ξ have the same answer before and after forgetting, i.e.,

$$\forall \boldsymbol{Q}, \boldsymbol{E} \subseteq \textsc{rv}(\textsc{forget}(\xi, F)), \boldsymbol{Q} \cap \boldsymbol{E} = \emptyset : P_F(\boldsymbol{Q}|\boldsymbol{E}) = P_{\textsc{forget}(\xi,F)}(\boldsymbol{Q}|\boldsymbol{E}) \tag{2}$$

Equation (1) only regards the relationship between the semantics of the two models, before and after forgetting, with Eq. (2) making explicit that the answer to any query remains identical as the semantics in itself does not change. The property does not constrain the syntactical representation, though.

Order-independence The order of forgetting two things should not matter, leading to the same outcome in either order. Formally, order independence between forgetting two components ξ_1, ξ_2 can be specified as follows:

$$\textsc{forget}(\xi_2, \textsc{forget}(\xi_1, F)) = \textsc{forget}(\xi_1, \textsc{forget}(\xi_2, F)) \tag{3}$$

This property regards the syntactical representation, i.e., F. In combination with semantics preservation, the semantics are preserved under the two forgetting operations, as each single operation is semantics preserving and can therefore be applied in any order. As mentioned above, this property might not be a property necessarily found in human forgetting but one that can be technically verified rather easily and therefore, characterises a forgetting method and also possibly contrasts it towards other approaches that focus on order-dependence.

Idempotence Forgetting something more than once should not be possible as the result of the first forgetting operation should make the application of the same forgetting operation not possible or, more specifically, without any effect. Formally, idempotence can be specified as follows:

$$\textsc{forget}(\xi, \textsc{forget}(\xi, F)) = \textsc{forget}(\xi, F) \tag{4}$$

Like order-independence, this property is defined on a syntactical level, with the equivalent semantics following the semantics preservation property.

Reversibility For humans, forgetting something is closely related to remembering, which we formalise as reversibility using $^{-1}$ to denote a remembering operation:

$$F = \textsc{forget}^{-1}(\xi, \textsc{forget}(\xi, F)) \tag{5}$$

This property is specified on a syntactical level, with semantics again following from semantics preservation. However, Eq. (5) also implies that one needs to be able to define a remembering function $\textsc{forget}^{-1}$ that is able to restore the original semantics of F. As mentioned before, whether reversibility is desirable in a given situation may depend on the specific application, but considering the (data) complexity needed to reverse a forgetting operation can also reveal something about the forgetting method, and as such, we consider reversibility a property of interest.

Finally, we would like to remark that also for forgetting in logics, especially for ASP [8] and propositional logic [24], several principles have been discussed, which have similar motivations to those principles above. Notably, Delgrade [6] provided a general approach to forgetting in Tarskian logics, in which order-independence and idempotence are also mentioned, which subsume several logical approaches to forgetting (but not all [3]).

In the following section, we look at VE as a means of forgetting components of a model F and define two forgetting operations, which we discuss regarding the above-defined properties as well as complexity.

4. Variable Elimination as Forgetting

Inspired by marginalisation as forgetting in logic, we look at VE to realise forgetting in a PGM. We use models of Def. 2 as the representation formalism, with factor graphs as the graphical representation, but since VE works on other PGMs like BNs as well, one could easily adopt forgetting for other formalisms.

Given a model F, the two natural components to forget are factors and random variables. Before we look at how to forget these components using VE, we define the inference problem of forgetting, which already encodes the central property of semantics preservation by ensuring equivalent semantics before and after forgetting.

Definition 6 (Forgetting problem)**.** *Given a model F, the syntactical construct of $\textsc{forget}(\xi, F)$ refers to the inference problem of* forgetting *a component $\xi \in F$ by computing a model F', in which $\xi \notin F'$ but $P_F \equiv P_{F'}$.*

Note that when setting $\xi = R$ with $R \in \mathrm{rv}(F)$, then $\xi \in F$ and $\xi \notin F'$ need to become $\xi \in \mathrm{rv}(F)$ and $\xi \notin \mathrm{rv}(F')$. Next, we define how to solve the problem of forgetting factors and random variables using VE operations, discussing the solution w.r.t. the above properties as well as complexity considerations. Other components than factors and random variables like edges, we discuss in Section 5.

4.1. Forgetting Factors

The first component we focus on are factors, specifying a solution method using the VE operation of MULTIPLY. Forgetting a factor is achieved by multiplying factors. Essentially, one forgets (conditional) independences among random variables. Doing so keeps the (conditional) independence alive in the full joint probability distribution but removes its explicit representation in the graph structure.

Solution Formally, forgetting a factor f can be achieved simply by applying the MULTIPLY operation from VE to f and another factor f', called target, shown in Alg. 4. Note that forgetting a factor f with target f' leads to the same result as forgetting f' with target f. One could even argue that forgetting a factor leads to forgetting both the factor itself as well as its target and replacing it with its merged version f''.

While any factor could be chosen as the target, it would be reasonable to choose a factor such that the tree width only increases as much as necessary. Choosing the target factor could follow any heuristics developed for choosing the next operation during a VE run such as choosing the factor which leads to the smallest result after multiplication. The effect on the graph structure can be described as follows: Forgetting a factor has the consequence of two factor nodes becoming one in the factor graph with its neighbours being the union of the neighbours of the previous two factors, i.e., $V \leftarrow V \setminus \{f, f'\} \cup \{f''\}$

Algorithm 4 Forgetting a Factor

1: **procedure** FORGETFACTOR(Factor f, model F)
2: Pick factor $f' \in F$ as target
3: $f'' \leftarrow \text{MULTIPLY}(\{f, f'\}, F)$
4: $F \leftarrow F \setminus \{f, f'\} \cup \{f''\}$

and $E \setminus \{\{R, f\} \mid R \in \text{rv}(f)\} \setminus \{\{R, f'\} \mid R \in \text{rv}(f')\} \cup \{\{R, f''\} \mid R \in \text{rv}(f'')\}$. Before we briefly argue for correctness, we illustrate forgetting a factor in our running example.

Example 5. *Consider forgetting f_0 in F_{ex}, i.e., FORGET(f_0, F_{ex}). We choose as target f_1, meaning that FORGETFACTOR(f_0, F_{ex}) multiplies f_0 and f_1, yielding a factor $f'' = \phi(Epid, Sick, Travel)$, with F_{ex} becoming $\{f'', f_2\}$.*

Theorem 3. *Given a model F and a factor $f \in F$, Alg. 4 correctly solves the problem FORGET(f, F).*

Proof. The forgetting problem defined in Def. 6 requires that after forgetting a component, the component is no longer part of the model, while the semantics are equivalent. Since Alg. 4 removes f from F in line 4, $f \notin F$ afterwards. Additionally, as forgetting is achieved by calling the VE operation of MULTIPLY, which has an equivalent algebraic operation of multiplication, yielding equivalent semantics, encoded in the postcondition in Alg. 1, correctness follows. □

Complexity Implications The complexity of the forgetting operation follows that of the multiply operation: It depends on the size of the resulting factor f'', i.e., $O(r^k)$, with k the number of distinct random variables $\boldsymbol{R}'' = \text{rv}(f) \cup \text{rv}(f')$ between the two factors f, f' involved and $r = \max_{R \in \text{rv}(f,f')} |\text{ran}(R)|$ the largest range size, as each potential for each mapping from $\text{ran}(\boldsymbol{R})$ needs to be calculated.

Since forgetting a factor is done by multiplication, the largest factor size may grow, which may increase the lower bound on the tree width characterising the complexity of probabilistic inference. Doing a multiplication because of forgetting intervenes with the elimination order, which may in turn lead to a larger tree width. If the factors multiplied together have argument sets that are subsets of each other, then the tree width does not change. If the forgetting operation combines random variables that were previously not in a factor together, intermediate results at least regarding this part of the VE run will be larger. If these intermediate results are actually those that determine the tree width, then the tree width will increase.

Example 6. *Consider forgetting f_1 in F_{ex}. Choosing f_0 as target, the result would be the same as in Example 5, f'', meaning that there is no difference between forgetting two factors if they are chosen as each other's target. Choosing f_2 as the target yields a factor $\phi(Epid, Sick, Travel, Treat)$, increasing the largest factor size from 2^3 to 2^4, requiring more computations during the multiplication itself as well as yielding a larger runtime complexity for any query answering problem in the resulting model.*

Properties *Semantics preservation* follows from Theorem 3 about the correctness of forgetting a factor w.r.t. the inference problem, which states that the semantics needs to be preserved. *Order-independence* also holds by commutativity of multiplication, given fixed targets. *Idempotence* holds since a forgotten factor f is no longer part of the model F, which violates the precondition of multiplication.

Reversibility does not hold without additional effort. Multiplying two factors to forget one essentially leads to forgetting the explicit representation of (conditional) independences holding between the random variables of the two factors being multiplied. However, due to semantics preservation, the (conditional) independences are still part of the full joint and could therefore be recovered by checking for (conditional) independences, for which the following has to hold:

$$P(\boldsymbol{R}_1, \boldsymbol{R}_2 \mid \boldsymbol{R}_3) = P(\boldsymbol{R}_1 \mid \boldsymbol{R}_3) \cdot P(\boldsymbol{R}_2 \mid \boldsymbol{R}_3) \tag{6}$$

Algorithm 5 Forgetting a Random Variable

1: **procedure** FORGETRANDVAR(Random variable R, model F)
2: $F' \leftarrow \{f \mid f \in F, R \in \mathrm{rv}(f)\}$
3: $f \leftarrow$ MULTIPLY(F', F)
4: $f \leftarrow$ SUM-OUT(R,f, F)
5: $F \leftarrow F \setminus F' \cup \{f\}$

Checking for (conditional) independences without knowing the original factorisation of the two factors can be costly, since any combination of variables or sets thereof need to be checked. If storing the argument lists of the two factors involved in forgetting a factor, which requires memory in the number of variables in the worst case, i.e., $O(N)$ with $N = |\mathrm{rv}(F)|$, then the graph structure can be easily recovered and the potentials retrieved up to a scaling factor by solving Eq. (6) for the argument lists, with their Markov blanket as $\boldsymbol{R}_3$, with the Markov blanket being those variables that block all paths from the argument lists to any other variable in the model.

With forgetting a factor in place and the main properties fulfilled, we turn to forgetting a random variable.

4.2. Forgetting Random Variables

The next natural component to forget in a factor graph is a random variable. Similarly to forgetting factors, forgetting random variables can be achieved using a VE operation, SUM-OUT in this case.

Solution Formally, forgetting a random variable R can be achieved by applying the SUM-OUT operation from VE to the random variable, shown in Alg. 5. The graph structure changes along the lines of the operations performed during forgetting a random variable R: The factor nodes for the factors in F that contain R, i.e., F', are replaced by a single factor node for f, which is the result of forgetting R in the product of F', and the variable node of R is deleted, i.e., $V \leftarrow ((V \setminus F') \cup \{f\}) \setminus \{R\}$. The edges between F' and its arguments are replaced by edges to f, i.e., $E \leftarrow E \setminus \{\{R', f'\} \mid f' \in F'\} \cup \{\{R', f\} \mid R' \in \mathrm{rv}(f)\}$. While using the sum-out operation, which multiplies all factors that contain R first, can change the syntactic representation quite drastically, if R appears in many factors, multiplying those factors is necessary to preserve the semantics as correctly summing out a variable requires all factors to be multiplied first. Summing out R only locally, i.e., in a single factor ignoring all other appearances, can drastically change the semantics.

Again, before we briefly argue for correctness, we illustrate forgetting a random variable in our running example.

Example 7. *Consider forgetting $Travel$ in F_{ex}, i.e., FORGET($Travel, F_{ex}$). Since $Travel$ occurs only in f_1, FORGETRANDVAR($Travel, F_{ex}$) does not need to multiply any factors, and can directly use the sum-out operator on $Travel$ and f_1, yielding a factor $f_1' = \phi(Epid, Sick)$, with F_{ex} becoming $\{f_0, f_1', f_2\}$. The equivalent algebraic operation is the following:*

$$\phi_1'(Epid, Sick) = \sum_{tl \in \mathrm{ran}(Travel)} \phi_1(Epid, Sick, tl)$$

Theorem 4. *Given a model F and a random variable $R \in \mathrm{rv}(F)$, Alg. 5 correctly solves the problem FORGET(R, F).*

Proof. After forgetting a random variable R in a model F, R must no longer appear in F, while the semantics remain equivalent as per Def. 6. By the precondition of the sum-out operator in Alg. 2, R may only appear in one factor, which after the main loop no longer contains R as R is then summed out. The resulting factor replaces the original factor, meaning that $R \notin \mathrm{rv}(F)$ after Alg. 5. Additionally, by the postcondition of the sum-out operator, the semantics remain equivalent, with the sum over

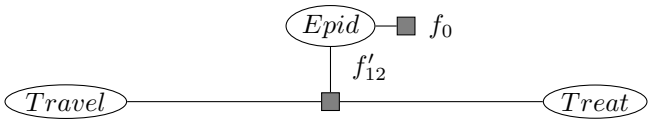


Figure 2: The factor graph for the example model F_{ex} of Example 1 after forgetting $Sick$ (Example 8)

the range of the random variable in the full joint the equivalent algebraic operation that preserves semantics. Thus, correctness follows. □

Complexity Implications The complexity of forgetting a random variable R again follows the complexity of the VE operations, with the multiply operation on a set of factors being in $O(r^k)$, with k the number of distinct random variables $\boldsymbol{R} = \bigcup_{f \in F'} \mathrm{rv}(f)$ in the factors F' containing R and $r = \max_{R \in \boldsymbol{R}} |\mathrm{ran}(R)|$ the largest range size. The sum-out operation has the same complexity as the sum runs through all entries and adds them up, leading to an overall complexity of $O(r^k)$.

Since forgetting a random variable possibly requires multiplying a set of factors, the largest factor size may grow as well, which again may lead to an increase in the lower bound on the tree width and the tree width itself: Since forgetting a random variable explicitly sums out a random variable detached from any elimination order, which requires multiplying all factors that contain said variable, the worst case is that all factors are multiplied, leading to a factor the size of the full joint distribution. The best case is that the random variable to forget appears in only one factor or, less strict, could be eliminated without enlarging the tree width.

Example 8. *Consider Example 7, in which we forgot $Travel$, which did not require multiplying any factors, keeping the complexity of the query answering problem unchanged. The same could be said for forgetting $Treat$. However, when forgetting $Sick$ or $Epid$, Alg. 5 needs to multiply factors that then have larger input sets as an interim result, with the former requiring f_1 and f_2 to be multiplied and the latter requiring all three factors. The result afterwards will again only contain three variables, yielding the same complexity of the query answering problem as before. However, one can also see in this case that by forgetting, e.g., $Sick$, we have obscured the explicit representation of the conditional independence between $Travel$ and $Treat$ (see Fig. 2 for the graphical representation of the result).*

Properties Just like forgetting a factor, forgetting a random variable using Alg. 5 fulfils the property of *semantics preservation* based on its correctness shown in Theorem 4, as semantics are preserved. *Order-independence* holds by commutativity of addition. However, the size of any interim result may depend on the order, meaning that the order can have an effect on the runtime, even though the result is the same. Regarding *idempotence*, as argued in the proof of Theorem 4, Alg. 5 removes a random variable from its model. By the precondition of the sum-out operator in Alg. 2, the random variable to sum out has to be part of the model, which is no longer the case after forgetting it with Alg. 5, meaning that we cannot forget it twice. Thus, idempotence holds.

To achieve *reversibility*, more effort is necessary compared to forgetting a factor. Setting aside the problem of possibly having multiplied several factors to ensure the precondition of summing out, which would require its own recovering action based on the reversibility of forgetting factors, the problem with forgetting a variable R from a factor f is that apart from storing, which factor contained R to begin with, it would also require that we store all original potentials, since we could not recover the original summands from the sum of potentials in the resulting factor. The memory necessary to do so would be exponential in the size of the original factor f used as a basis for summing out, i.e., $O(r^n)$, with $r = \max_{R \in \mathrm{rv}(f)} |ran(R)|$ and $n = |\mathrm{rv}(f)|$.

5. Discussion

In this section, we discuss other components to forget in a PGM as well as extensions of PGMs and other interpretations of forgetting.

5.1. Other Components of a PGM

Considering that a PGM is always also a graph, one can think of forgetting edges. Additionally, while the semantics of PGM is given by its full joint in this article, one can also consider the (conditional) independences used for the factorisation and encoded in the graph structure for defining semantics. We discuss these cases next.

Edges One could argue that forgetting edges is not feasible in the sense of forgetting as we have set it up here: A basic assumption in probabilistic modelling is that random variables depend on each other. Only if identifying a (conditional) independence, a factorisation is permitted,[3] which implies that knowing a (conditional) independence is knowledge that can be forgotten, which happens when we multiply factors for forgetting them. Deleting an edge would mean assuming an independence between random variables, which adds information and does not remove it.

Apart from this more philosophical argument, the operation could not be implemented with VE operations. It would require changing a factor locally by somehow removing an argument from it, which may change the full joint behind the factorisation: A straightforward approach would be to sum out the argument in the factor, without multiplying all other factors containing that variable. That however changes the full joint as we violate the order of sum and product operations if the variable occurs in more than one factor. Other possibilities would be to not add up the entries for the different range values but to calculate an average or select the minimum or maximum potential, all of which change the full joint. Addressing this particular research question is part of future work.

Conditional Independences Forgetting specific conditional independences might not be feasible depending on the graph structure as random variables can interact in various ways and forgetting a specific conditional independence might require multiplying two factors that lead to further independences being obfuscated. One could consider handling forgetting a specific conditional independence between two random variables in a light weight manner by adding a direct connection between the two variables through a constant factor, that is a factor mapping all inputs to a single constant, which does not change the full joint at all since all rows are multiplied by that constant, which vanishes through normalisation. While the full joint would remain identical and the set of (conditional) independences hidden in the full joint would remain unchanged, the graph structure would contain more edges, possibly leading to fewer (conditional) independences in the graph.

Direct Dependences A complementary idea to forgetting a conditional independence is to forget a direct dependence, which would require splitting factors in some way. There actually is a division operation [25] as an inverse of the multiplication operation that might be of interest here. Given that a factor always specifies dependence between a set of random variables, it remains to be seen how one can isolate a single dependence and forget it, which might then have close connections to forgetting edges.

5.2. Other Dimensions of Forgetting in PGMs

We briefly consider more expressive models that also allow for encoding time or first-order constructs, for which one could consider setting up a forgetting operation.

[3] To handle learning a PGM efficiently, of course one does not want to start with a fully connected graph and start deleting edges but with a sparse graph adding edges, but that is notwithstanding the probabilistic foundations.

Temporal Models The models considered so far capture static scenarios in the sense that there is no temporal dimension. Temporal models usually consist of a static model describing the behaviour of the modelled scenario within a time step as well as a set of factors describing how the state of a model changes from one time step to the next, which are used as template models that can be instantiated for different time steps [26]. Most temporal models use the Markov-k assumption, which assumes that the current time step only depends on the k previous ones, to simplify the inference problem, with $k = 1$ as the most prominent one. Forgetting edges has therefore a lot of potential as it may reduce the complexity of temporal inference if reducing a model with a Markov-k assumption to one with $k' < k$.

Decision-theoretic Models As mentioned in Section 3, decision-theoretic PGMs are PGMs extended with decision nodes and utility functions, which are used to compute a plan or policy that leads to an maximum expected utility, which can be solved by using the VE operators as defined above [23, 27, 28, 29]. Forgetting in such models could therefore be considered in a similar manner as given in this paper.

First-order Models Forgetting as considered in this article concerns propositional PGMs, that is, models without any first-order constructs. There exist a set of first-order probabilistic models, with parfactor graphs [30] being a first-order counterpart to the factor graphs defined here and Markov logic networks [31] a counterpart to Markov networks, next to the popular ProbLog system as a probabilistic extension of Prolog [32]. Especially formalisms such as Markov logic networks and ProbLog, which use weighted sentences, might provide a basis for a direct translation of logical forgetting to a weighted version. In first-order settings, with logical variables allowing for specifying factors for groups of random variables that behave indistinguishably, new aspects to forget arise like constants of the domain of logical variables or whole sets of indistinguishable random variables. In addition, lifted VE as a version of VE that works on parfactor graphs [30, 33] has the potential for enabling forgetting in parfactor graphs.

Of course, there also exist combinations of the dimensions mentioned above, such as first-order temporal models [34, 35], first-order decision-theoretic models [28, 36, 29] as well as first-order temporal decision-theoretic models [37], which can all benefit from a discussion of forgetting in its different dimensions as well as in their combination.

Further modelling variants that are interesting to consider are PGMs for continuous settings such as Gaussian Bayesian networks [38] or causal models encoding cause-effect relationships between random variables [39] as well as first-order variants [40, 41, 42] of these, which all could also be treated to a forgetting operation tailored towards their assumptions and inference problems.

5.3. Other Notions of Forgetting

In contrast to the notion of forgetting above and possible extensions to more expressive models, one can consider a different notion of forgetting altogether, which we highlight next.

Probabilities We do not meddle with the potentials in a model themselves. However, one could also consider that forgetting could refer to the certainty about specific potentials. One way could be to look at Credal networks [43] based on imprecise probability theory [44], which are an extension of BNs and allow for specifying intervals instead of specific probabilities.

Abstraction Forgetting here is considered as a form of removing syntactical constructs while keeping the semantics of the model. Forgetting could also be interpreted as a form of abstraction, where different, almost indistinguishable factors are combined into a single (representative) factor, using logical variables to encode that the new factor represents two instances. Advanced colour passing (ACP) [45] and, especially, its approximate version ϵ-ACP [46] allow for combining (almost) equivalent factors, which could be considered a form of abstraction. A hierarchical version of ϵ-ACP [47], which builds a

hierarchy of increasingly more compact models, could possibly emulate the more hierarchical nature of human forgetting. Temporal approximate merging (TAMe) [48] uses a similar idea when merging parfactors over time after evidence has lead to a splitting factors parameterised with logical variables into smaller and smaller groups.

Overall, as this discussion shows, there are a plethora of starting places to further investigate forgetting in PGMs, focussing on other modelling formalisms or other solution methods, adding to a hopefully emerging landscape of forgetting in PGMs.

6. Conclusion

This paper looks at VE as a basis for forgetting in PGMs, transferring the idea of marginalisation as forgetting in logic to the probabilistic setting. Doing so allows for forgetting random variables and factors through forgetting operators based on VE operators, which enables transferring results regarding correctness or complexity from the VE to forgetting. The article also considers the properties semantics preservation, order-independence, idempotence as well as reversibility and discusses the forgetting operators in the light of these properties, with semantics preservation built in through its definition and order-independence as well as idempotence following from commutativity of addition and multiplication. Reversibility remains the property that requires additional effort to achieve. Within this setting, specific edges and conditional independences cannot be natively forgotten and require further analysis. The article also discusses a handful of other dimensions to consider for forgetting like time and first-order constructs, which are also left as future work for proper analysis.

Acknowledgments

The authors wish to thank the participants of the KR&AI Meet-ups over the years in Hagen, who have helped shape this work through their questions and comments.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] T. Eiter, G. Kern-Isberner, A brief survey on forgetting from a knowledge representation and reasoning perspective, KI 33 (2019) 9–33. URL: https://doi.org/10.1007/s13218-018-0564-6. doi:10.1007/s13218-018-0564-6.

[2] C. Beierle, I. J. Timm, Intentional forgetting: An emerging field in AI and beyond, KI – Künstliche Intelligenz 33 (2019) 5–8. URL: https://doi.org/10.1007/s13218-018-00574-x. doi:10.1007/s13218-018-00574-x.

[3] C. Beierle, G. Kern-Isberner, K. Sauerwald, T. Bock, M. Ragni, Towards a general framework for kinds of forgetting in common-sense belief management, Künstliche Intell. 33 (2019) 57–68. URL: https://doi.org/10.1007/s13218-018-0567-3. doi:10.1007/S13218-018-0567-3.

[4] J. Lang, P. Liberatore, P. Marquis, Propositional independence: Formula-variable independence and forgetting, J. Artif. Intell. Res. 18 (2003) 391–443.

[5] A. S. Sha, B. P. Nunes, A. Haller, Selective forgetting in machine learning and beyond: A survey, ACM Comput. Surv. 58 (2026) 241:1–241:41. URL: https://doi.org/10.1145/3796542. doi:10.1145/3796542.

[6] J. P. Delgrande, A knowledge level account of forgetting, J. Artif. Intell. Res. 60 (2017) 1165–1213.

[7] G. Kern-Isberner, T. Bock, C. Beierle, K. Sauerwald, Axiomatic evaluation of epistemic forgetting operators, in: Proceedings of the 32nd International FLAIRS Conference, FLAIRS-32, AAAI Press, 2019, pp. 470–475.
[8] R. Gonçalves, M. Knorr, J. Leite, The ultimate guide to forgetting in answer set programming, in: C. Baral, J. P. Delgrande, F. Wolter (Eds.), Principles of Knowledge Representation and Reasoning: Proceedings of the Fifteenth International Conference, KR 2016, Cape Town, South Africa, April 25-29, 2016., AAAI Press, 2016, pp. 135–144.
[9] G. Boole, An Investigation of the Laws of Thought, on which are founded the mathematical theories of logic and probabilities, Walton and Maberly, London, 1854.
[10] F. Lin, R. Reiter, Forget it!, in: Proceedings of the AAAI Fall Symposium on Relevance, AAAI Press, Menlo Park, CA, 1994, pp. 154–159.
[11] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
[12] V. Isham, An Introduction to Spatial Point Processes and Markov Random Fields, International Statistical Review 49 (1981) 21–43.
[13] F. R. Kschischang, B. J. Frey, H.-A. Loeliger, Factor Graphs and the Sum-product Algorithm, IEEE Transactions on Information Theory 47 (2001) 498–519.
[14] D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques, The MIT Press, 2009.
[15] N. L. Zhang, D. Poole, A Simple Approach to Bayesian Network Computations, in: Proc. of the 10th Canadian Conference on Artificial Intelligence, Springer, 1994, pp. 171–178.
[16] R. Ramachandran, A. Ramer, A. C. Nayak, Probabilistic belief contraction, Minds Mach. 22 (2012) 325–351. URL: https://doi.org/10.1007/s11023-012-9284-0. doi:10.1007/S11023-012-9284-0.
[17] J. P. Delgrande, G. Lakemeyer, M. Pagnucco, J. Sack, Belief revision in a probabilistic setting, in: M. Ortiz, R. Wassermann, T. Schaub (Eds.), Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning, KR 2025, Melbourne, Australia, November 1-17, 2025, 2025. URL: https://doi.org/10.24963/kr.2025/25. doi:10.24963/KR.2025/25.
[18] K. Sauerwald, C. Beierle, G. Kern-Isberner, Propositional variable forgetting and marginalization: Semantically, two sides of the same coin, in: A. Meier, M. Ortiz (Eds.), Foundations of Information and Knowledge Systems - 13th International Symposium, FoIKS 2024, Sheffield, UK, April 8-11, 2024, Proceedings, volume 14589 of *Lecture Notes in Computer Science*, Springer, 2024, pp. 144–162. URL: https://doi.org/10.1007/978-3-031-56940-1_8. doi:10.1007/978-3-031-56940-1_8.
[19] A. Darwiche, Modeling and Reasoning with Bayesian Networks, Cambridge University Press, 2009.
[20] T. Chakraborti, A. Kulkarni, S. Sreedharan, D. E. Smith, S. Kambhampati, Explicability? Legibility? Predictability? Transparency? Privacy? Security? The Emerging Landscape of Interpretable Agent Behavior, in: ICAPS-19 Proc. of the 29th International Conference on Automated Planning and Scheduling, AAAI Press, 2019, pp. 86–96.
[21] T. Chakraborti, S. Sreedharan, S. Kambhampati, The Emerging Landscape of Explainable Automated Planning & Decision Making, in: IJCAI-20 Proc. of the 29th International Joint Conference on Artificial Intelligence, IJCAI Organization, 2020, pp. 4803–4811.
[22] S. L. Vasileiou, W. Yeoh, PLEASE: Generating Personalized Explanations in Human-Aware Planning, in: ECAI-23 Proc. of the 27th European Conference on Artificial Intelligence, IOS Press, 2023, pp. 2411–2418.
[23] R. A. Howard, J. E. Matheson, Influence Diagrams, in: Readings on the Principles and Applications of Decision Analysis, Strategic Decision Group, 1984, pp. 721–762.
[24] A. Becker, G. Kern-Isberner, K. Sauerwald, C. Beierle, Forgetting formulas and signature elements in epistemic states, in: L. Amgoud, R. Booth (Eds.), 19th International Workshop on Non-Monotonic Reasoning (NMR 2021), Hanoi, Vietnam, November 2-5, 2021, Proceedings, 2021, pp. 233–242.
[25] F. V. Jensen, S. L. Lauritzen, K. G. Olesen, Bayesian Updating in Recursive Graphical Models by Local Computations, Computational Statistics Quarterly 4 (1990) 269–282.
[26] K. P. Murphy, Dynamic Bayesian Networks: Representation, Inference and Learning, Ph.D. thesis,

University of California, Berkeley, 2002.
[27] S. Joshi, K. Kersting, R. Khardon, Generalized First Order Decision Diagrams for First Order Markov Decision Processes, in: IJCAI-09 Proc. of the 21st International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc., 2009, pp. 1916–1921.
[28] G. Van den Broeck, I. Thon, M. van Otterlo, L. De Raedt, DTProbLog: A Decision-theoretic Probabilistic Prolog, in: AAAI-10 Proc. of the 24th AAAI Conference on Artificial Intelligence, AAAI Press, 2010, pp. 1217–1222.
[29] M. Gehrke, T. Braun, R. Möller, A. Waschkau, C. Strumann, J. Steinhäuser, Lifted Maximum Expected Utility, in: Artificial Intelligence in Health, Springer, 2019, pp. 131–141.
[30] D. Poole, First-order Probabilistic Inference, in: IJCAI-03 Proc. of the 18th International Joint Conference on Artificial Intelligence, IJCAI Organization, 2003, pp. 985–991.
[31] M. Richardson, P. Domingos, Markov Logic Networks, Machine Learning 62 (2006) 107–136.
[32] L. De Raedt, A. Kimmig, H. Toivonen, ProbLog: A Probabilistic Prolog and its Application in Link Discovery, in: IJCAI-07 Proc. of 20th International Joint Conference on Artificial Intelligence, IJCAI Organization, 2007, pp. 2062–2467.
[33] N. Taghipour, D. Fierens, J. Davis, H. Blockeel, Lifted Variable Elimination: Decoupling the Operators from the Constraint Language, Journal of Artificial Intelligence Research 47 (2013) 393–439.
[34] J. Vlasselaer, W. Meert, G. Van den Broeck, L. De Raedt, Exploiting Local and Repeated Structure in Dynamic Baysian Networks, Artificial Intelligence 232 (2016) 43–53.
[35] M. Gehrke, T. Braun, R. Möller, Relational Forward Backward Algorithm for Multiple Queries, in: FLAIRS-19 Proc. of the 32nd International Florida Artificial Intelligence Research Society Conference, AAAI Press, 2019, pp. 464–469.
[36] A. Nath, P. Domingos, A Language for Relational Decision Theory, in: Proc. of the 6th International Workshop on Statistical Relational Learning, 2009.
[37] M. Gehrke, T. Braun, R. Möller, Lifted Temporal Maximum Expected Utility, in: Proc. of the 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019, Springer, 2019, pp. 380–386.
[38] D. Geiger, D. Heckerman, Learning Gaussian Networks, in: UAI-94 Proc. of the 10th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1994, pp. 235–243.
[39] J. Pearl, Causal Diagrams for Empirical Research, Biometrika 82 (1995) 669–688.
[40] J. Choi, E. Amir, D. J. Hill, Lifted Inference for Relational Continuous Models, in: UAI-10 Proc. of the 26th Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2010, pp. 13–18.
[41] M. Hartwig, R. Möller, T. Braun, An Extended View on Lifting Gaussian Bayesian Networks, Artificial Intelligence 330 (2024) 104082.
[42] M. Luttermann, M. Hartwig, T. Braun, R. Möller, M. Gehrke, Lifted Causal Inference in Relational Domains, in: CLeaR-24 Proceedings of the 3rd Conference on Causal Learning and Reasoning, PMLR, 2024, pp. 827–842.
[43] F. G. Cozman, Credal Networks, Artificial Intelligence 120 (2000) 199–233.
[44] P. Walley, Towards a Unified Theory of Imprecise Probability, International Journal of Approximate Reasoning 24 (2000) 125–148.
[45] M. Luttermann, T. Braun, R. Möller, M. Gehrke, Colour Passing Revisited: Lifted Model Construction with Commutative Factors, in: AAAI-24 Proc. of the 38th AAAI Conference on Artificial Intelligence, AAAI Press, 2024, pp. 20500–20507. Oral presentation.
[46] M. Luttermann, J. Speller, M. Gehrke, T. Braun, R. Möller, M. Hartwig, Approximate Lifted Model Construction, in: IJCAI-25 Proceedings of the 34th International Conference on Artificial Intelligence, IJCAI, 2025, pp. 9077–9085.
[47] J. Speller, M. Luttermann, M. Gehrke, T. Braun, Compression versus Accuracy: A Hierarchy of Lifted Models, in: ECAI-25 Proceedings of the 28th European Conference on Artificial Intelligence, IOS Press, 2025, pp. 5051–5058.
[48] M. Gehrke, R. Möller, T. Braun, Taming Reasoning in Temporal Probabilistic Relational Models, in: ECAI-20 Proc. of the 24th European Conference on Artificial Intelligence, IOS Press, 2020, pp. 2592–2599.