

Compatible and Incompatible Abstractions in Bayesian Networks

Barbaros Yet* and William Marsh

School of Electronic Engineering and Computer Science, Queen Mary, University of London, London, UK

*Corresponding Author: Barbaros Yet

Risk and Information Management (RIM) Research Group, Room CS332, School of Electronic Engineering and Computer Science, West Square, Queen Mary, University of London, E1 4NS

E-mail: b.yet@qmul.ac.uk

Tel: +44 20 7882 8027

Abstract: The graphical structure of a Bayesian network (BN) makes it a technology well-suited for developing decision support models from a combination of domain knowledge and data. The domain knowledge of experts is used to determine the graphical structure of the BN, corresponding to the relationships and between variables, and data is used for learning the strength of these relationships. However, the available data seldom match the variables in the structure that is elicited from experts, whose models may be quite detailed; consequently, the structure needs to be abstracted to match the data. Up to now, this abstraction has been informal, loosening the link between the final model and the experts' knowledge. In this paper, we propose a method for abstracting the BN structure by using four 'abstraction' operations: node removal, node merging, state-space collapsing and edge removal. Some of these steps introduce approximations, which can be identified from changes in the set of conditional independence (CI) assertions of a network.

Keywords: Bayesian Networks, Knowledge Engineering, Abstraction

1 Introduction

A knowledge-based Bayesian network (BN) aims to model the data-generating process of a problem domain by encoding knowledge about influences and independences between the important variables of the domain. The first step in building the BN is for a knowledge engineer to elicit the structure of the BN from domain experts. When the structure is finalised, any available data can be used to learn the parameters of the BN or, if no data are available, the parameters can also be elicited. This paper is about the knowledge engineering techniques used in the first stage of this process: the development of the BN structure.

Knowledge-engineered BNs are often developed through multiple stages as the knowledge engineers and the domain experts refine the model iteratively (Laskey and Mahoney, 2000). The initial knowledge model is often large and detailed, and some elements of the model may need to be simplified or *abstracted* as data is lacking or the parameters are too difficult to elicit. However, even simple abstraction operations, such as removing a node, can result in numerous and complicated alternative BNs which are difficult for the knowledge engineers to evaluate without a structured method. The effects of these abstractions must be carefully examined by domain experts to prevent any unwanted changes in the modelled knowledge of the data generating process. Moreover, the way that the final BN has been derived needs to be presented thoroughly so that the knowledge base of the model and its derivation is understandable.

Our aim is to present a method of abstracting a BN structure. The method is developed for knowledge engineers developing a BN structure with domain experts. The method provides a set of abstraction operations which together:

1. Allow a BN to be simplified by removing and merging nodes, removing edges and reducing the number of states
2. Distinguish abstractions that add to the knowledge base from those compatible with the knowledge elicited so far, so that the added knowledge can be confirmed by domain experts
3. Provide a way to show the link between the initial and abstracted models, in the form of a derivation that captures the complete sequence of abstraction operations

The method can be used to help knowledge engineers to select the most suitable model refinements by evaluating alternative abstractions, in consultation with domain experts. The selection may also be guided by considering the availability of data or compatibility with causal relationships.

Our knowledge engineering method is based on well-known techniques mainly used for learning and inference problems (Shachter, 1986; Wellman and Liu, 1994; van Engelen, 1998; Choi and Darwiche, 2006). Our main contribution is to explore knowledge engineering aspect of these operations.

The remainder of this paper is organised as follows: Section 2 reviews the related work. Section 3 gives an overview of the relation between knowledge and conditional independences (CI) in BNs. Section 4 introduces abstraction as a knowledge engineering

method and Section 5 describes the abstraction operations of this method and examines their compatibility properties. These operations are illustrated by a medical case-study in Section 6. Section 7 shows the graphical representation of the abstraction operations, and Section 8 presents the conclusions.

2 Related Work

In this section, we review the previous studies about abstraction and knowledge engineering of causal models and BNs. The abstraction methodology proposed in this paper was initially motivated by ABEL (Patil, 1981). Section 2.1 discusses this motivation by illustrating the similarities and differences between ABEL and BNs. Section 2.2 reviews the existing knowledge engineering techniques for eliciting and abstracting a BN structure. Since this paper focuses on the BN structure, techniques for eliciting parameters are not discussed.

2.1 ABEL

ABEL (Patil, 1981; Patil et al. 1981) was a pioneering clinical expert system that was developed for diagnosing acid-base disturbances of patients. Given laboratory data about a particular patient, ABEL generates the relevant causal diagrams from its knowledge-base, which is called a patient specific model (PSM). It reasons by abstracting and elaborating these causal models to make diagnostic inferences, which is considered to be similar to how clinicians express their decisions. The causal models at the higher levels are directed acyclic graphs like BNs. Although widely referenced, PSMs have not become a commonly used approach for developing clinical decision support models.

The causal diagrams of ABEL have several differences from the BN formalism. First, each node in a PSM represents a single state of a variable, whereas each node in a BN represents a variable that can have multiple states. Second, the lower abstraction levels of a PSM can have feedback loops which are always eliminated at higher levels; but BNs are acyclic graphs. Third, a PSM does not reason probabilistically and its reasoning mechanism does not take the prior probabilities of diseases into account; BNs have superior probabilistic reasoning algorithms that are able to calculate complex learning and inference problems. Fourth, BNs are lacking techniques for abstracting their knowledge-base for different levels of detail as used in ABEL. Abstraction is clearly necessary for developing knowledge-based BNs for complex domains, and for explaining these models to external users. Our work in this paper was initially motivated by ABEL, notably its hierarchical structure and abstraction operations.

2.2 Knowledge Engineering Approaches in BN

Wu and Poh (2000) propose a set of operations that change the abstraction level of knowledge-based influence diagrams. The ‘extend’ and ‘retract’ operations respectively add and remove the parents of a variable. The ‘abstract’ operation merges a set of variables that share a single parent and child. The ‘refine’ operation is the opposite of ‘abstract’, dividing a variable with a single parent and child into multiple variables. These operations can be applied to limited and simple modelling tasks. For example, Wu and Poh (2000) do not discuss merging variables that do not share parents.

Srinivas (1994) proposes a hierarchical BN approach for the fault diagnosis problem in engineering systems. In this approach, functional schematics can be defined in multiple levels of abstraction between the inputs and outputs of the system. Shachter's node removal operation (Shachter, 1986) is used to reach to higher level schematics. The different abstraction levels of schematics must have the same inputs and outputs.

Neil et al. (2000) use specific BN fragments called idioms for representing common types of uncertain reasoning. Knowledge engineers and domain experts select the most appropriate idioms for their modelling problems and use these idioms as building blocks for their BN structure. Idioms are reused for similar modelling tasks in order to develop BNs efficiently and consistently. Koller and Pfeffer (1997) describe object-oriented Bayesian Networks (OOBN), representing BNs with inter-related objects. OOBN are particularly useful for complex models that contain repeated fragments, where objects can be reused to decrease the modelling effort. Laskey and Mahoney (1997) also use object-oriented concepts to construct a BN by using semantically meaningful fragments as the basic building blocks. Laskey and Mahoney (2000) propose a system engineering approach that uses a spiral lifecycle model for the development of BNs. Their approach starts by defining objectives and building initial prototypes with simple features. These prototypes are evaluated and rebuilt. As this process proceeds the knowledge engineer understands the domain and the domain experts understand the principles of BN. The systems engineering approach uses network fragments (Laskey and Mahoney, 1997) as basic elements of model building.

Heckerman (1990) describes similarity networks that can be used for diagnosing a single hypothesis that has mutually exclusive and exhaustive states. In this approach, each pair of similar hypotheses is connected in a similarity network. A separate BN network structure is elicited for each pair of these similar hypotheses. Then, the separate BN structures are merged to form the final BN structure. This approach divides the task of network building into pieces that are easier to manage. However, it can only be applied when the hypotheses are mutually exclusive and exhaustive and the hypothesis variable has no parents. Parent divorcing approach can reduce the parameter space by adding a variable to the BN (Nielsen and Jensen, 2007). As the parameter space of a variable increases exponentially with the number of its parents, adding an intermediate variable between the variable and its parents can make parameter space smaller.

3 Conditional Independences in BNs

Bayesian Networks

A BN represents a joint probability distribution compactly in a factorised way. The structure of a BN is a directed acyclic graph that consists of nodes representing variables and directed edges encoding a set of CI conditions about these variables. Every node in a BN is independent of its non-descendants given that the state of its parents is known. Therefore, each node has a conditional probability distribution that defines its probabilistic relation with its parents. A probability distribution P_X factorises over a BN structure G_X if P_X can be

decomposed into the product of factors $P_X = P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | PA_{X_i}^{G_X})$ where X_1, \dots, X_n are a set of variables, $PA_{X_i}^{G_X}$ is the set of parents of X_i in G_X .

We say that G_X asserts the set of CIs $I(G_X)$. P_X can factorise on G_X if $I(G_X)$ is a subset of $I(P_X)$ where $I(P_X)$ is the set of CIs in P_X .

Domain Knowledge and Conditional Independences

The aim of a knowledge-based BN is to model the data-generating process for the domain by encoding knowledge about influences and independences in the BN structure. A satisfactory modelling of this knowledge is when a BN G_X is able to represent the joint probability distribution P_X of the data-generating process (see Fig. 1).

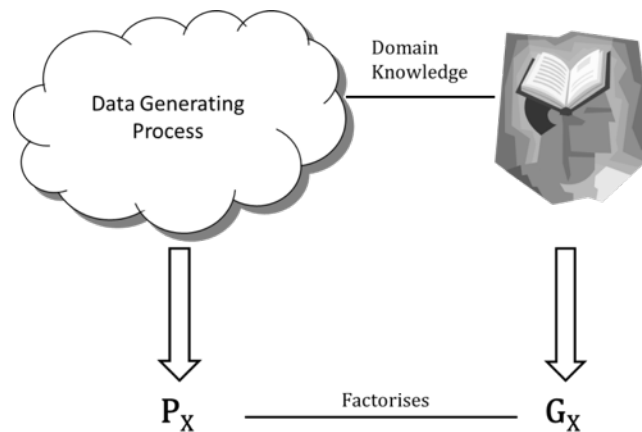


Fig. 1 Knowledge-based Bayesian Network

A BN structure encodes a set of CIs; therefore its ability to represent a data-generating process depends on the CIs the BN asserts. Consequently, compatibility of the BN abstractions can be evaluated by their effect on the CI assertions of the BN.

CI assertions of a BN: D-separation

The CI assertions of a BN can be determined by d-separation (Pearl, 1988).

d-separation: A trail $X_1 \rightleftharpoons \dots \rightleftharpoons X_n$ is a consecutive sequence of edges that can be in any direction. Let G be a BN structure, A, B and V be three disjoint sets of nodes in G . A and B are d-separated by V , $dsep_G(A; B|V)$, if and only if there is no *active trail* between A and B given that V is observed. An active trail requires the following conditions:

- 1) For every converging relation $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$ in the trail, the node X_i or one of its descendants is a member of V .
- 2) The other nodes in the trail are not members of V .

If A and B are d-separated given V in the BN structure G , then A and B are conditionally independent given V in any probability distribution that factorises over the BN. A and B are called d-connected if they are not d-separated. It follows from the definition of d-separation that adding an edge to a BN increases the number of trails and therefore does not increase the number of CI conditions.

Any CI that holds on the BN structure G must also hold on the probability distribution P , if P factorises over G . On the other hand, P can have additional CI conditions that are not reflected in G . Therefore, a probability distribution can factorise over various BN structures.

An example of this situation can be seen by the BNs in Fig. 2. Some probability distributions can factorise on both of these BNs even though their graphical structure is different. In the BN in Fig. 2a, as well as in the probability distribution P , A and B are conditionally independent given that the state of C is not known. This CI is not represented in the graphical structure of the BN in Fig. 2b. However, the CI condition can still be present in the probability distribution that factorises over this BN structure. In other words, the CI between A and B can be encoded in the parameters of this BN rather than its structure. The BN on the left is preferable since an edge between A and B is unnecessary for this probability distribution, and additional edges increase the computational burden of a BN. The obvious conclusion is to choose a BN structure that encodes all of the CIs of the probability distribution in its graphical structure. Unfortunately, this is not possible in general. Symmetric variable-level CIs or some regularities in the parameters do not have a BN structure that represents all of the CIs (Pearl, 1988).

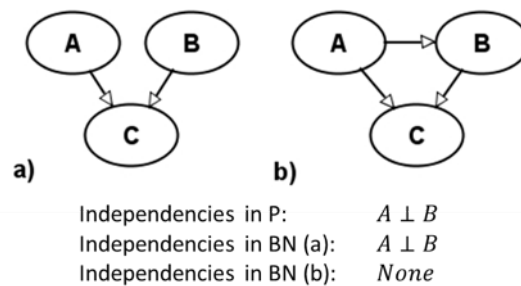


Fig. 2 Same probability distribution factorised over two different BN structures

4 Abstraction as a Knowledge Engineering Method

In this section, we present an overview of our abstraction method for knowledge-engineered BNs. Our approach is to construct the BN structure based on expert knowledge before using available data, statistics from relevant studies, and numbers elicited from domain experts to parameterise the BN. The first step of our method is to elicit the BN structure for the domain. The initial structure should be considered as a knowledge-model of the domain so it should include all relevant variables and relations without being limited by issues such as availability of data or model complexity. Knowledge engineering techniques for eliciting the BN structure are discussed in Section 2.2.

When the initial structure of the BN is complete, we compare it with the resources for parameterising the BN. The data and the statistics from relevant studies may not be enough to learn the parameters for all of the variables in the BN. For example, some of the variables may not be directly observed or observations may not be recorded in practice. Moreover, the conditional probability tables of some variables may be too large to elicit or learn from

available resources. Consequently, the initial BN structure may need to be abstracted in order to have a parameterised and working BN.

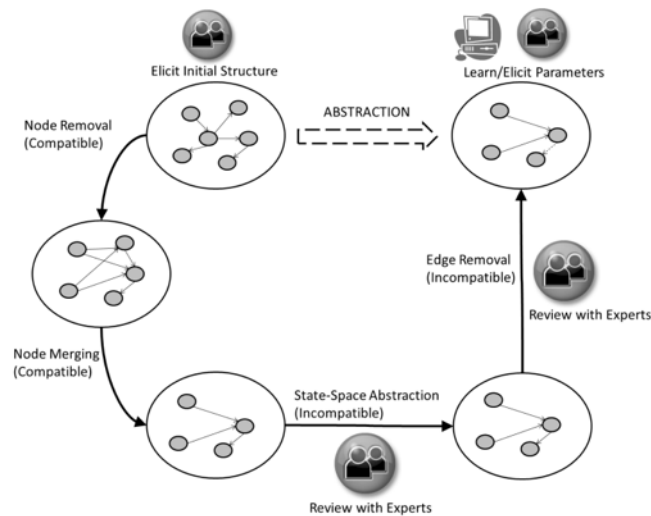


Fig. 3 Overview of abstraction as a method of model development

We propose 4 abstraction operations that are classified as compatible or incompatible as shown in Table 1. Node removal and node merging are ‘compatible’ abstractions meaning that these operations can always be applied without adding new CI conditions to the variables shared between the initial and the abstracted BN. State-space abstraction and edge removal are ‘incompatible’ abstractions that bring additional CI conditions to the BN structure. A probability distribution for the data-generating process that factorises on the initial BN structure may no longer factorise when incompatible abstractions are applied. Therefore, the domain experts should review the effects of incompatible abstractions on the knowledge-base of the BN. The compatibility properties of the abstraction operations are defined in Section 5.1. Fig. 3 illustrates the overview of our method. In summary:

- We use compatible abstractions first to reduce the number of variables in the model
- We then make approximations with incompatible abstractions to improve learning and elicitation often as a complement to compatible abstractions

However, the order of the abstraction operations is not restricted to this pattern; any operation can be applied to any network.

Table 1 Abstraction Operations

Operation	Compatibility
Node removal	Compatible
Node merging	Compatible
State-space abstraction	Incompatible
Edge removal	Incompatible

The purpose of the BN must be considered when removing variables: some variables may have primary importance, and they should not be removed even when no data is available; whereas other variables can be removed without affecting the reasoning mechanism of the model significantly. The node removal operation is described in Section 5.2.

In the next step, merging is used on the variables that require simplifications but should not be removed. The suitable candidates for merging are the variables that together represent an abstract concept. These variables can be described as the parts of the definition of the abstract concept. The node merging operations is demonstrated in Section 5.3.

After determining variables of the BN, the states of the variables can be abstracted in order to make it feasible to learn or elicit their parameters with available resources. The domain experts should review the CI conditions added as a result of state-space abstraction. The state-space abstraction operation is described in Section 5.4.

The final abstraction step in our method is the edge removal. This operation can be used to further simplify the BN when its parameter space is too large to be learned or elicited. The edge removal is also an incompatible abstraction; therefore, its effects on the CIs should be reviewed by experts. The data, if available, can be used to assess the effects of the edge removals by using statistical independence tests or model selection scores. The edge removal operation and the approaches for reviewing its effects are described in Section 5.5.

5 Abstraction Operations

5.1 Compatibility of Abstractions

A BN structure has fewer variables, states or relations than the initial knowledge model after the abstraction operations are applied. A crucial factor to consider is whether the abstracted BN G_A is a compatible abstraction that is able to represent the probability distribution of the remaining variables. In this section, we present the definition of compatible abstractions in the case where some variables are removed from the model. In the following sections we will expand this definition for the node merging operation, and discuss the compatibility of the state-space abstraction and edge removal operations.

A compatible node removal operation is able to represent the probability distribution of the remaining variables in the BN. Let G_X be a BN structure where the probability distribution P_X of the set of variables X factorises. When a set of nodes is removed from this BN, the abstracted BN structure G_A must be able to represent the probability distribution of the remaining variables A . Since $A \subseteq X$, the probability distribution of P_A of the set of variables A is simply a marginalisation of P_X , and the compatible abstraction G_A is able to factorise $P_A = \sum_{X \setminus A} P_X$. This is possible if the abstraction operation does not introduce additional independence assertions to the BN structure so that G_A asserts a subset of the d-separations in G_X .

Compatible Abstraction: G_X is an compatible abstraction of G_A if $dsep_{G_A}(D; E|F) \subseteq dsep_{G_X}(D; E|F)$ where D, E, F are three sets of variables in A .

When abstraction operations results in an incompatible structure by adding a CI assertion, the knowledge engineers must carefully evaluate the differences between the additional CI and the initial knowledge-based model (See Fig. 4).

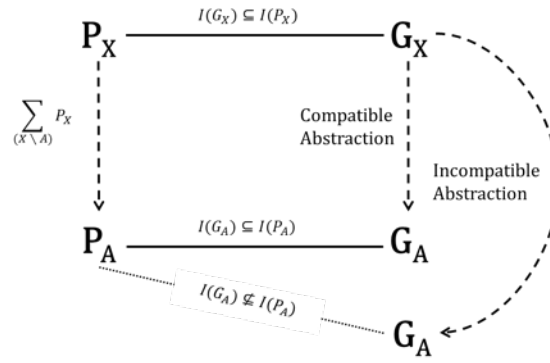


Fig. 4 Compatible and Incompatible Abstraction

In the remainder of this section, we present each of the abstraction operations and discuss their compatibility properties.

5.2 Node Removal

Any node in a BN can be removed without adding independence assertions using Shachter's topological operations (Shachter, 1986; 1989). The node removal operation is based on the concepts of barren nodes, covered edges and edge reversals:

1. **Barren Nodes:** Nodes that do not have any descendants in the BN are called barren nodes. Removing barren nodes does not add CI assertions to the rest of the BN: $dsep_{G_A}(D; E|F) = dsep_{G_X}(D; E|F)$ where $(X \setminus A)$ are barren nodes and D, E, F are three sets of variables in A .
2. **Covered Edge and Edge Reversal:** An edge $Y \rightarrow Z$ in a graph G is *covered* if $PA_Y^G = PA_Z^G \setminus \{Y\}$ that is if the set of parents of Y and set of parents of Z excluding Y are equivalent. Covered edges can be reversed without adding any independence assertions to the BN structure. This is useful since the outgoing edges from a node can be reversed to make the node barren and remove the node without adding CIs when these edges are covered.
3. **Adding Edges:** Adding edges to a BN structure does not add CI assertions to G_X , but it removes CI assertions, since all the previously active trails are still present and the added edges may add further active trails: $I(G_E) \subseteq I(G_X)$ where G_E is derived from adding some edges to G_X . As a result, any edge in a BN can be covered without introducing CIs by adding edges between the variables that the edge connects and their parents.

In summary, we can cover any edge in a BN by adding more edges to the BN structure, and this will not add more CI assertions. We can reverse any edge and make any node barren since we can cover any edge in the BN structure. Therefore, we can remove any node as a compatible abstraction without adding CI assertions to the BN (Shachter, 1989).

Removing a node can increase the parameter-space and computational complexity of a BN especially if the removed node has many children. As the number of children of a node increases, more edges are required to cover the outgoing edges from the node. For example, if

we remove the node R from the BN structure in Fig. 5a, we need to add 4 edges to keep the structure compatible, and the number of parents of C, D and E is tripled (Fig. 5c).

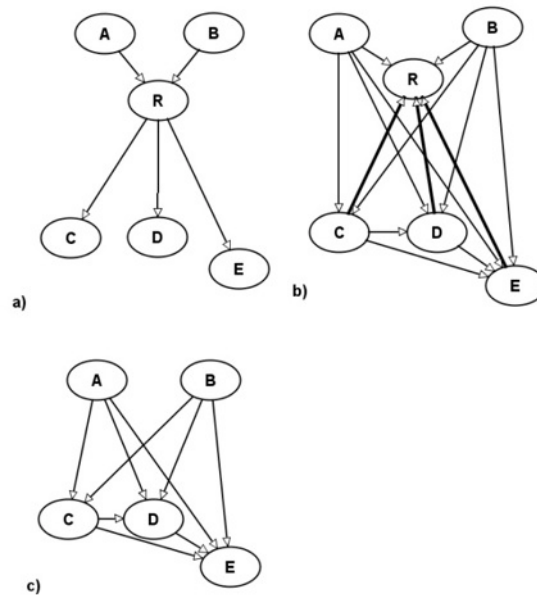


Fig. 5 (a) R with multiple children (b) Making R a barren node (c) R removed

Suppose that we need to reverse the edge $A \rightarrow B$ so that A becomes a barren node. If another directed path exists from $A \rightarrow \dots \rightarrow B$ then reversing the edge $A \rightarrow B$ would introduce a cycle. However, in this case node B has another parent ' X ' on this path; since X is a descendant of A and the BN is acyclic we can be sure that there is no directed path from X to B via A . In general, since the BN is acyclic, the set PA_B must contain at least one node that does not have an additional directed path leading to B . This node is not greater than the other parents of B in the partial order on nodes defined by the BN's graph. It is therefore always possible to select a node from PA_B so that its edge to B can be reversed without introducing a cycle and is therefore possible to make any node in a BN barren without introducing a cycle.

Since the BN's graph defines only a partial, not total order, on nodes, we may have to choose the order in which the edges leading from a node R are reversed as we transform the BN to make R a barren node. Changing the order of edge reversal can change the structure of the final BN. For example, we have to reverse $R \rightarrow D$ and $R \rightarrow C$ in order to remove R from the BN shown in Fig. 6a. If we start by reversing $R \rightarrow D$ the resulting structure has a total of 6 edges as shown in Fig. 6b. However, if we start by reversing $R \rightarrow C$, the resulting structure has 7 edges as shown in Fig. 6c.

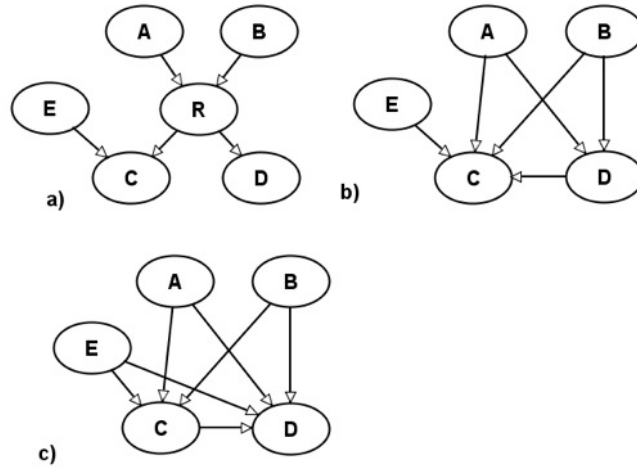


Fig. 6 (a) Initial BN; Equivalent abstractions when edge reversals start from (b) $R \rightarrow D$ and (c) $R \rightarrow C$

The order of edge reversals can be selected in a way to minimise either the number of edges added or the size of the final state space. Having the least number of additional edges does not guarantee that the state-space is minimised. A BN with fewer edges can have a larger state-space due to the state-space of its individual variables.

The search-space of possible equivalent abstractions gets increasingly large as the removed node has more children. The outgoing edges from a node that has n children can be reversed in $n!$ possible orders. This makes it difficult to evaluate all alternatives to find the minimal state-space size or number of added edges, especially for the nodes with many children. However, an exhaustive search can be unnecessary from a knowledge engineering perspective. A relation in a BN can represent causality or association. Edges added between the children of a removed node are assumed to represent association due to the missing parent therefore they can be modelled in any direction. However, some of the directions can make more sense to the domain experts even though these variables are assumed to be independent when the state of missing parent is known.

If the domain experts have no preference about the directions of edges, following heuristics can be useful to choose the order of removing nodes without making an exhaustive search:

- In order to add fewer edges, start reversing from the edge $R \rightarrow X$ where X has the smallest number of parents. When multiple edges must be reversed to remove R , an edge is added from X and $PA_X^G - \{R\}$ in order to cover the edges directed to other children of R that are reversed later. Therefore, if we start from reversing the node that has the smallest number of parents, fewer edges will be added to cover the edges that are reversed later.
- Similarly, in order to have a smaller increase in the size of the state-space, start reversing from the edge $R \rightarrow X$ where X has the smallest number of parameters.

Node removal is a well-known technique that has been primarily used for inference problems (Shachter, 1986; 1989; 1990). It is a useful operation for knowledge engineering of the BN structure as it explicitly shows the number and possible direction of edges that must be added for abstraction. A node removal operation is straightforward when the removed node has a

single or no child. In this case, no edges are added and the edge directions are maintained. However, removing a node with multiple children can result in multiple equivalent BN structures with many additional edges as shown in Fig. 5. The additional edges are necessary for compatibility but incompatible abstractions may follow node removals for further simplification.

Using Edge Removal with Node Removal

The edge removal operation (see Section 5.5) can complement node removal when a large number of edges are added as a result of removing a node. Rather than making ad-hoc approximations, domain experts should evaluate the CIs modelled by each added edge, and remove the ones that are considered to be trivial. An example of a node removal followed by edge removals is shown by the case-study in Section 6.

5.3 Node Merging

The second of the compatible abstraction operations is merging multiple nodes T_1, \dots, T_n into a single node M . Two nodes T_i and T_j can be merged into a single node M when an edge can be added between T_i and T_j and this edge is reversible by adding extra edges to cover it. The network with the merged variable combines the networks with this edge in both directions. The merging operation is a compatible abstraction since covering and reversing the edge between T_i and T_j does not add CI conditions to the BN structure (see Section 5.2). When we merge T_i and T_j into M , the state-space of M becomes the Cartesian product of the state-spaces of T_i and T_j .

Node merging does not change the probability distribution that factorises on the BN. The main difference is that T_i and T_j cannot be observed separately, they must be observed or unobserved together. Therefore, observing M is equivalent to observing T_i and T_j together while comparing the CI conditions between G_X and G_A .

An example of the merging operation is shown in Fig. 7 where the nodes X and Y are merged into XY . The first step is to add an edge between X and Y as shown in Fig. 7b. Next, this edge is covered and reversed as shown in Fig. 7c. The final BN with the merged variable XY (see Fig. 7e), can be seen as equivalent to the BNs with edges in both directions (see Fig. 7d).

Multiple nodes can be merged into a single node by repeating the merging operation pairwise. For instance, the nodes X, Y and Z in Fig. 7a can be merged pairwise by first merging X and Y to XY (see Fig. 7e), then merging XY and Z to XYZ as shown in (see Fig.8).

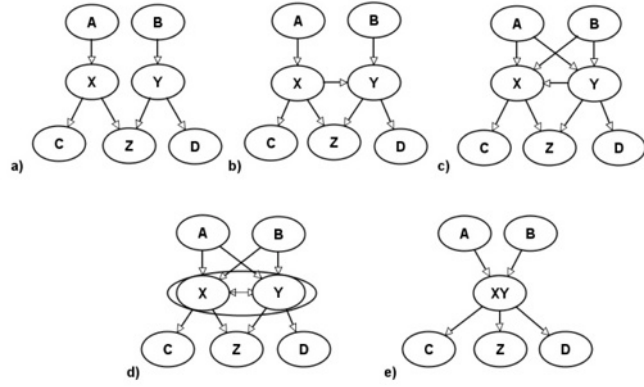


Fig. 7 (a) Initial BN (b) $X \rightarrow Y$ added (c) $X \rightarrow Y$ reversed (d) BNs with $X \rightarrow Y$ and $X \leftarrow Y$ combined (e) XY merged

A BN with merged nodes has the same or fewer CI assertions than the initial BN. The edges added after edge reversal and covering operations may remove some CIs. Moreover, some of the CI encoded in the initial BN disappears after merging since the merged variables cannot be observed separately. For example, A and B is independent given that X and Y is observed in Fig. 9a, but this independence disappears after merging X and Y as shown in Fig. 9b.

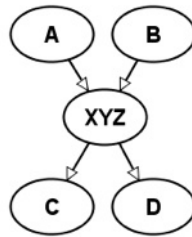


Fig. 8 XY and Z merged

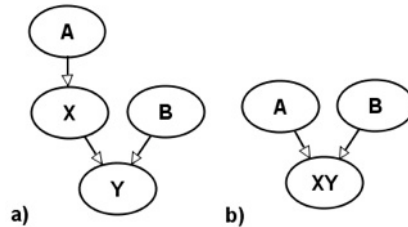


Fig. 9 (a) Initial BN (b) X and Y Merged

The merging operation introduces a cycle if there is a directed path between the nodes to be merged, T_i and T_j , that includes some other node. These cycles have to be eliminated since BNs are acyclic graphs. Our solution is to reverse some of the edges in these directed paths until none of the directed paths remain between T_i and T_j . In other words, if there is a directed path between T_i and T_j that includes other nodes, we break this directed path by reversing some of the edges in it. There are two issues to consider while reversing these edges:

- The reversed edges must not introduce a cycle as well.
- The edges must be covered before reversing (see Section 5.2).

For example, we cannot immediately merge X and Y in Fig. 10 since there is a directed path $X \rightarrow B \rightarrow D \rightarrow Y$ between X and Y that includes the nodes B and D . We have to reverse one

of the edges between $X \rightarrow B$, $B \rightarrow D$ or $D \rightarrow Y$ to prevent the merging operation from introducing a cycle.

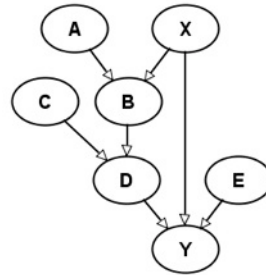


Fig. 10 BN before X and Y are merged

We consider whether any of $X \rightarrow B$, $B \rightarrow D$ or $D \rightarrow Y$ are covered or not. If one of them is covered, it can be reversed; if none are covered we need to add more edges to cover and reverse them:

- Parents of $B = \{A, X\}$, parents of $X = \{ \}$ therefore the edge $X \rightarrow B$ is not covered. The edge $A \rightarrow X$ has to be added in order to make this edge covered.
- Parents of $D = \{C, B\}$, parents of $Y = \{D, X, E\}$; therefore the edge $D \rightarrow Y$ is not covered. The edges $C \rightarrow Y$, $B \rightarrow Y$, $X \rightarrow D$ and $E \rightarrow D$ must be added in order to make this edge covered.
- Parents of $B = \{A, X\}$, parents of $D = \{C, B\}$; therefore the edge $B \rightarrow D$ is not covered. The edges $A \rightarrow D$, $X \rightarrow D$, $C \rightarrow B$ must be added in order to make this edge covered.

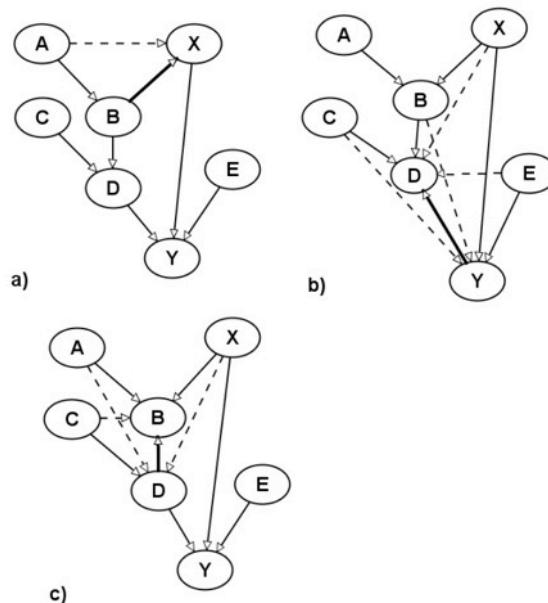


Fig. 11 (a) $X \rightarrow B$ reversed (b) $D \rightarrow Y$ reversed (c) $B \rightarrow D$ reversed

Fig. 11a, Fig. 11b and Fig. 11c shows the model structure when the edges $X \rightarrow B$, $D \rightarrow Y$ and $B \rightarrow D$ are covered and reversed respectively. The edges that were added are shown by dashed lines, and the reversed edges are shown by bold lines in these figures. The BN in Fig. 11a does not have a directed path between X and Y therefore the nodes X and Y can be merged. The BNs in Fig. 11b and Fig. 11c, on the other hand, still have directed paths

$X \rightarrow B \rightarrow Y$ and $X \rightarrow D \rightarrow Y$ respectively. We need to reverse more edges in order to merge X and Y in these BNs. Since we already have a structure without a directed path between X and Y (Fig. 11a), we continue with this structure. The joint probability distribution $P(A, B, C, D, XY, E)$ can be represented in the same way in these two BNs assuming that observing XY is equivalent to observing X and Y at the same time in the initial BN.

The BN in Fig. 12 has one more edge compared to the initial BN. It is possible to select the sequence of reversals that leads to the fewest number of additional edges or to the smallest state-space. In our example, reversing $X \rightarrow B$ (Fig. 11a) added the least number of edges compared to reversing $D \rightarrow Y$ (Fig. 11b) or $B \rightarrow D$ (Fig. 11c).

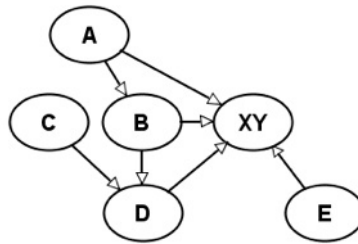


Fig. 12 Compatible Merging of X and Y

In general, it is possible to merge any pair of nodes in a BN. In the worst case the process described above will lead to a fully connected BN, with no CI conditions. Since all fully connected BNs that are formed by the same set of nodes have equivalent CI conditions (i.e. none), any (acyclic) combination of edge directions can be reached by an appropriate sequence of reversals of covered edges. Therefore, any edge can be reversed in a fully connected network, and any pair of nodes can be merged.

Although possible, not all merging operations are sensible from a knowledge engineering perspective. In Fig. 10, X is a cause of Y , and B and D are the intermediate variables between the cause and the effect. Merging the cause and the effect and leaving the intermediate variables in the model would not be sensible in most knowledge engineered BNs. One would prefer to merge an intermediate variable with either its cause or its effect when simplifying a causal BN.

Using Node Merging with State-Space Abstraction

Abstraction by merging variables is suitable when multiple variables in the model are parts of the definition of a more abstract concept. For example, an engine fault can be either explained as an abstract concept with binary states, or it can be explained in detail by describing the exact faults in each component of the engine. The detailed version of a BN for detecting car faults may have variables about faults in pistons, alternators and crankshaft. These variables can be merged into a single variable about engine fault but we would expect this variable to have more abstract states, rather than the Cartesian product of the states of the detailed variables. State-space abstraction will follow node merging in such cases (see Section 5.4). An example of node merging followed by state-space abstraction is shown by the case-study in Section 6.

5.4 State-space Abstraction

State-space abstraction collapses multiple states of a variable into a single state. For example, suppose a variable has 4 states named {None, Moderate, Severe, Profound}. We could collapse ‘Moderate’, ‘Severe’ and ‘Profound’ states into a single state called ‘Present’. As a result, the variable would have 2 states named {None, Present} and it would require fewer parameters to be learnt or elicited. This operation is often used in combination with node merging (See Section 5.3).

State-space abstraction is an incompatible abstraction even though it does not change the BN structure (Wellman and Liu, 1994; Chang and Fung, 1990). The changes in the CI and their effects to domain representation should be discussed with domain experts before state-space abstractions are made.

5.5 Edge Removal

Edge removal is an incompatible abstraction that always adds CI assertions to the BN structure (see Section 3). Therefore, an edge should not be removed if it is not recommended by experts or statistical evidence. By adding CIs, edge removal decreases the parameter space and computational complexity of the BN. Edges added after node removal and merging operations, and weak relations represented in the BN structure are suitable candidates for edge removal. The remainder of this section presents knowledge and data driven approaches for assisting edge removal.

Using Domain Knowledge for Selecting Edges for Removal

Before removing an edge, the effects of removing the edge should be reviewed with domain experts. The review focuses on the strength of the relation modelled by the edge, and the additional CI conditions caused by removing it. Reviewing the edges between all subsets of variables may not be feasible if the BN is large. In this case, the domain experts can identify the important variables in the domain, and limit the review to these variables.

Edge removal can be used with node removal and node merging as these operations can add a large number of edges to the BN structure (see Section 5.2 and 5.3). In this case, the experts can review the effects of removing each of the edges added by these operations. For example, when the node R is removed from the BN in Fig. 13a edges are added between the variables B , C and D as shown in Fig. 13b. Removing these edges can simplify the BN but it will add CIs that are not present in the initial BN. For example, removing $C \rightarrow D$ and $B \rightarrow C$ as in Fig. 13c adds two CI conditions: $(B \perp C | A)$ and $(C \perp D | A)$. These CIs do not exist in the initial structure G and should be discussed with domain experts.

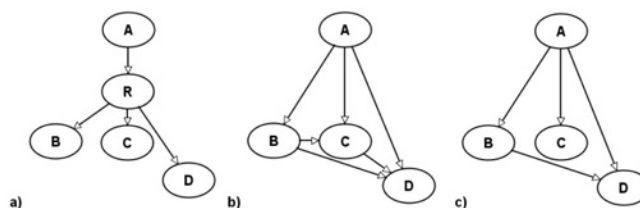


Fig. 13 (a) Initial BN (b) Node R removed (c) Edges $B \rightarrow C$ and $C \rightarrow D$ removed

Using Data for Selecting Edges for Removal

Statistical independence tests can also be used for selecting edges to be removed if there is adequate amount of data. These tests analyse statistically significant CIs between variables. There are widely used statistical tests suitable for this purpose including χ^2 test, and the tests proposed by Margaritis (2004) and Dash and Druzdzel (2003). Model selection scores, such as Bayesian Information Criterion (Schwarz, 1978), can be used to compare the effects of removing edges based on data. Another possible approach is to evaluate the sensitivity of the parameters of a target variable to edge removal (Renooij, 2010).

6 Case-Study: Shock

6.1 Background

Bleeding is one of the most common causes of death following injury. Blood is the medium through which oxygen, which is vital for metabolism, is delivered to tissues. Patients with significant blood loss are unable to adequately perfuse their tissues with blood and thus unable to adequately deliver oxygen to the tissues. The body tissues start to die if starved of oxygen for a prolonged time. The body responds in several ways to compensate for the effects of blood loss. First, the heart rate increases in order to maintain normal perfusion and blood pressure. However, as the blood loss increases the increased heart rate cannot compensate and the blood pressure decreases. Second, as oxygen delivery to the tissues decreases, less efficient (anaerobic) metabolism takes over to compensate for the lack of oxygen. The by-products of anaerobic metabolism increase the acidity of blood and tissues, which in turn causes the respiration rate to increase. A patient in whom these mechanisms are operating as a result of bleeding is said to be in a state of haemorrhagic shock.

It is not possible to observe the state of the many variables explained above directly. For example, the oxygen perfusion to tissues cannot be measured precisely. Instead, these ‘latent’ variables are inferred from related measurements and clinical observations. Reasoning of this type suggests the potential of a BN to detect shock.

6.2 Shock BN

We developed a BN structure for reasoning about the physiology of bleeding patients and for predicting the risk of death from shock. We did not consider the limitations of available data at this stage; our aim was to model all main variables and relations indicated by domain experts. The structure of the BN can be seen in Fig. 14. The circular nodes in this BN are the main clinical variables, and rectangular nodes are the measurements and observations relevant to main clinical variables. Table 2 shows the definition of the main variables in the BN. A possible next step would be to learn the parameters of this BN from data. However, we do not have data about some of these variables since they are either not observed or not recorded in practice. We use abstraction operations to make it possible to learn the model with the available data.

Table 2 Definitions of Variables in Shock BN

Variable	Definition
<i>Bleeding Body Parts</i>	The number of bleeding body compartments.
<i>Hypovolemia</i>	Decrease in the volume of blood in circulatory system. Heart rate (HR) increases as a result of this to maintain normal blood pressure.
<i>Cardiac Output</i>	Volume of blood ejected from left side of the heart in 1 minute. As the blood loss increases cardiac output and blood pressure (SBP) will fall.
<i>Hypoperfusion</i>	Decrease in the volume of blood perfusion to the tissues. Urine output and Glasgow coma scale (GCS) is dependent on kidney and brain perfusion respectively. Body temperature (Temp) indicates the degree of overall perfusion. Therefore, these values can be used to estimate the degree of hypoperfusion.
<i>PVS</i>	Vascular resistance to the flow of blood in peripheral arterial vessels. Degree of PVS can be estimated by capillary refill time (CRT).
<i>Oxygen Delivery</i>	Amount of oxygen delivery to the tissues
<i>Metabolic Acidosis</i>	As oxygen delivery to the tissues decreases, less efficient (anaerobic) metabolism takes over to compensate for the lack of oxygen. The by-products of anaerobic metabolism increase the acidity of blood and tissues, which in turn causes the respiration rate (RR) to increase. The degree of metabolic acidosis can be estimated by pH, base excess (BE) and lactate values in blood.
<i>Death</i>	The risk of death in 48 hours

6.3 Node Removal

We removed the ‘CRT’, ‘Urine Output’, ‘PVS’, ‘Blood Loss’ and ‘Metabolic Acidosis’ variables from the BN structure in this order. These variables were selected with domain experts, considering both the objectives and the knowledge-base of the BN.

The measurement variables ‘CRT’ and ‘Urine Output’ are already barren nodes and can therefore be removed without any edge reversal operations. The ‘PVS’ variable becomes barren after ‘CRT’ is removed so no edge reversal is needed for it either. Removing the ‘Blood Loss’ and ‘Metabolic Acidosis’ variables requires 1 and 4 edge reversal operations respectively. The resulting BN structure after the node removal operations can be seen in Fig. 15.

Removing the ‘Metabolic Acidosis’ variable with a compatible abstraction made the abstracted BN more complex: 6 edges were added between the children of this variable and one of its children has now 4 parents. These edges do not represent causal relations but removing them will introduce CIs that were not present in the initial BN. Before simplifying the graph, the effects of removing each of these edges on CIs should be investigated by domain experts (see Section 5.5). Our method enables a clear documentation of abstraction steps and allows evaluation of each approximation with domain experts.

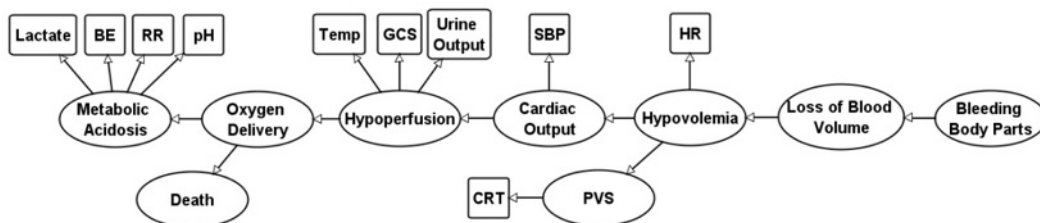


Fig. 14 Initial Structure of the Physiology BN

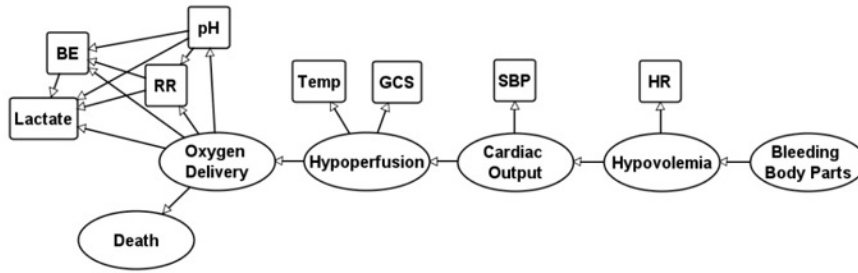


Fig. 15 BN structure after node removals

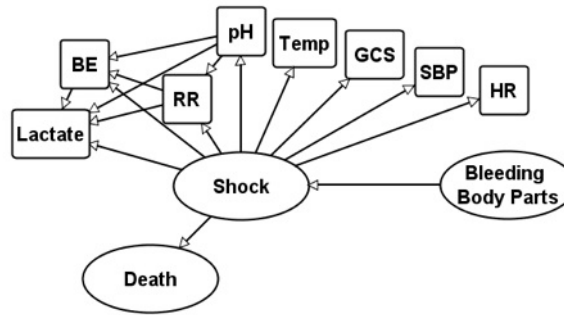


Fig. 16 BN structure after node merging

6.4 Node Merging

We also do not have an adequate amount of data about the ‘Oxygen Delivery’, ‘Hypoperfusion’, ‘Cardiac Output’ and ‘Hypovolemia’ variables. However, the domain experts do not recommended removing these variables since they represent an important physiological mechanism about bleeding. We have to elicit the parameters of these variables from domain experts, but a simplification is considered to be necessary for this task.

The domain experts stated that oxygen delivery, hypoperfusion, cardiac output and hypovolemia are elements of a more abstract physiological definition called shock. Shock is clinically defined as a metabolic disturbance due to the failure of the circulatory system to maintain adequate perfusion to vital organs. Hypovolemia and cardiac output is associated with failure of the circulatory system, hypoperfusion represent the failure to maintain adequate perfusion, finally oxygen delivery and its relation to death represents the metabolic disturbance. Therefore these 4 variables can be merged into a single variable called shock. We did not have to reverse any edges during merging since the merged node do not have directed path that includes a non-merged node between them. The resulting BN after merging can be seen in Fig. 16.

6.5 State-Space Abstraction

The node merging operations have simplified the structure of the BN resulting in a ‘Shock’ variable that is clinically easier to elicit. However, the state-space of the ‘Shock’ variable still requires simplification since it is formed by the Cartesian product of each of the states of the merged nodes. For simplicity we defined binary states for the merged variables: {Normal, Abnormal}. The domain experts stated that shock is present when all of the 4 factors are abnormal. In other words, we are interested in two states of the shock variable: the state when

all of the factors are abnormal, and when all of the factors are not abnormal. An illustration of this state-space abstraction is shown in Table 3.

Table 3 States of Shock Before and After State-Space Abstraction

Before state-space abstraction ¹	After state-space abstraction
A,A,A,A	Abnormal
N,N,N,N	Normal
⋮	
N,A,A,A	

¹A: Abnormal, N: Normal

6.6 Edge Removal

Several edges were added between ‘BE’, ‘Lactate’, ‘RR’ and ‘pH’ as a result of removing the ‘Metabolic Acidosis’ variable. We listed the CI conditions that removing each of these edges can bring, and discussed these CIs with the domain experts. For example, when we remove RR → BE, RR becomes independent of BE given that shock or variables that are merged into shock are observed. This CI was not present in the initial BN. The resulting BN structure after edge removals can be seen in Fig. 17.

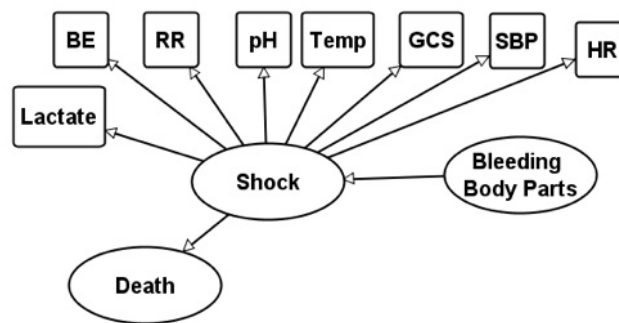

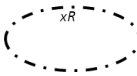



Fig. 17 Final abstracted BN after edge removal

7 Graphical Notation for Abstractions

In this section, we present a graphical notation that shows the order and type of abstraction operations applied to a BN. The notation shows how the abstracted structure is derived from the initial BN by showing each abstraction step. This is essential for communicating the knowledge-base of the BN or for deriving a more detailed version of the BN when, for example, more data become available. We use both the initial and the abstracted BN structures to show the abstraction steps. These structures are annotated by the symbols shown in Table 4. In the remainder of this section, we illustrate the application of these symbols by using the case study about shock.

Table 4 Symbols for Abstraction Operations

Operation	Symbol	Operation	Symbol
Node Merging		Edge added after node merging	\xrightarrow{xM}
Node Removal		Edge reversed after node merging	\xrightarrow{xM}
State-space Abstraction		Edge added after node removal	\xrightarrow{xR}
		Edge Removal	$\xrightarrow{\text{---}}$

Removed nodes are shown with dashed boundaries and annotated by ‘ xR ’ where x is an integer that indicates the order of the abstraction operation and R indicates a removal operation. Edges that are added as a result of node removal operation are shown by an ‘ xR ’ annotation with the same ‘ x ’. In the case study, the first abstraction operation is the removal of the ‘Metabolic Acidosis’ variable, and this operation adds several edges to the model. This variable and the added edges are annotated by ‘ $1R$ ’ in the initial and abstracted BNs respectively as shown in Fig. 18.

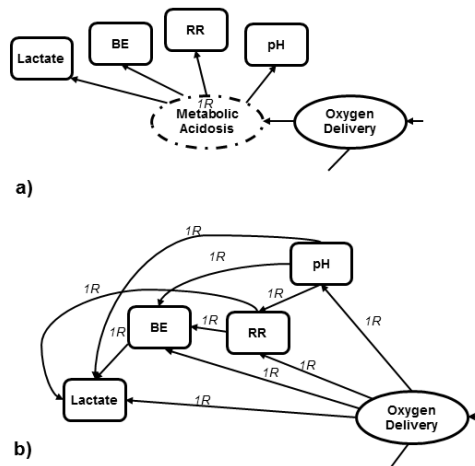


Fig. 18 Notation for initial (a) and abstracted (b) BN fragment for removal of Metabolic Acidosis variable

The nodes undergoing a node merging operation are annotated by ‘ xM ’ where x is the order of the abstraction operation and M stands for ‘merging’. In the shock example, we merged four nodes: ‘Oxygen Delivery’, ‘Hypoperfusion’, ‘Cardiac Output’ and ‘Hypovolemia’. The node resulting from this merging operation is ‘Shock’. We annotate each of these variables with ‘ $2M$ ’ since it is the second abstraction operation and it is a merging operation. The initial and abstracted BNs after this operation can be seen in Fig. 19.

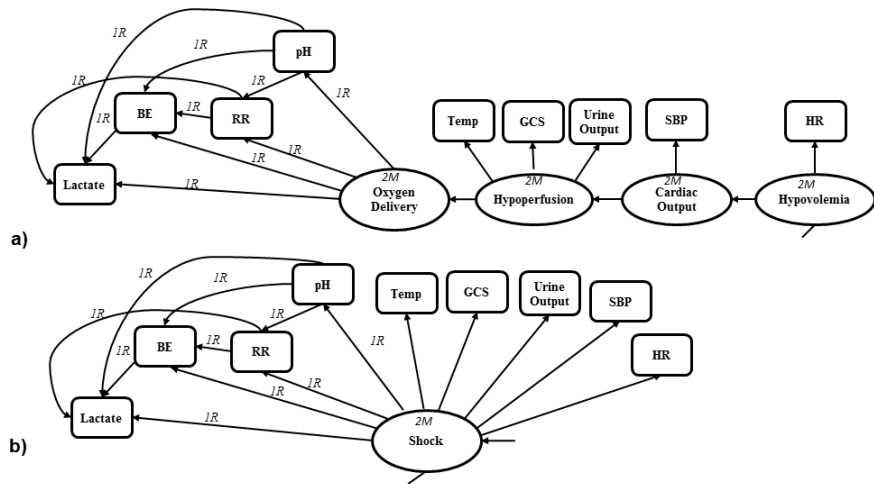


Fig. 19 Notation showing before (a) and after (b) merging multiple variables into Shock variable

A node merging operation may introduce edges (see Section 5.3). In this case, both the added and the reversed edges are annotated by ‘xM’, and the reversed edges are shown by bold lines.

Variables with state-space abstractions are shown by a double lined boundary, and removed edges are shown by dashed lines. The order of state-space abstraction and edge removal does not change the BN structure (other than the removed edge itself) so the order is not annotated. However, the collapsed or removed states after state-space abstraction must be documented (for example, as in Table 3). In the case study, the state-space of ‘Shock’ is abstracted and the edges between ‘pH’, ‘RR’, ‘BE’ and ‘Lactate’ is removed in order to simplify the BN, as shown in Fig. 20.

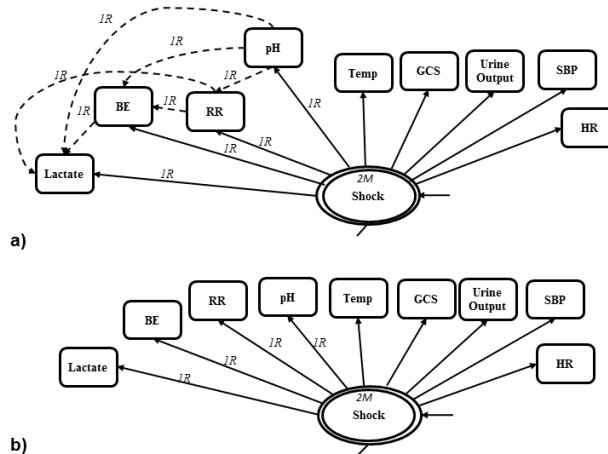


Fig. 20 Notation showing edge removals and state-space abstraction (a) and BN structure after removing edges (b)

Fig. 21 shows the initial and abstracted structure of the Shock BN with all the abstraction operations annotated. The graphical notation presented in this section clarifies the derivation of the final abstracted structure from the initial – detailed – structure. Multiple graphs each showing a particular step of abstraction can be stored if the model is large and requires many complex abstraction operations.

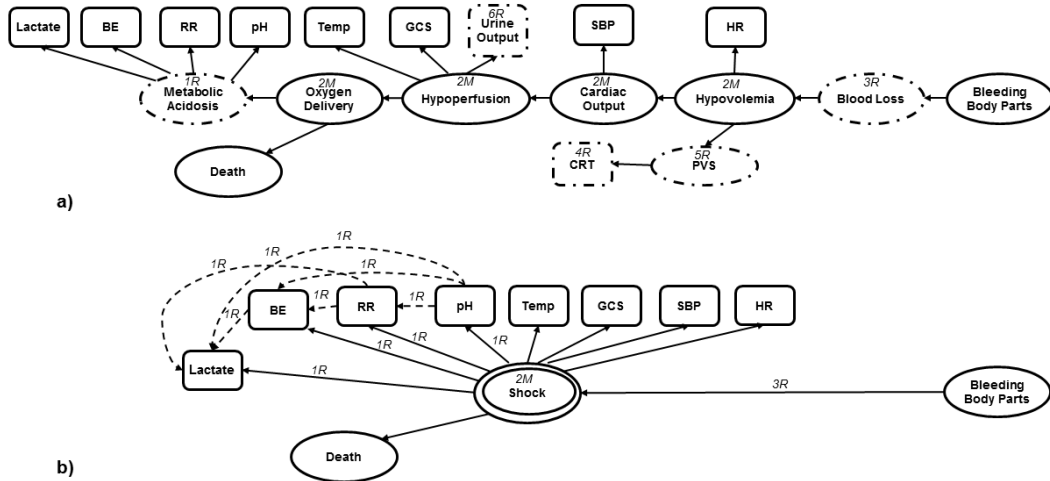


Fig. 21 Initial (a) and abstracted (b) bleeding physiology BN with abstraction notation

8 Conclusion

This paper proposed abstraction as a knowledge engineering method for simplifying BN structure. The method is illustrated by a medical case study about haemorrhagic shock. Our method provided:

1. A sufficient set of operations that simplify a BN by removing and merging nodes, removing edges and reducing the number of states
2. The compatibility properties of each abstraction in terms of CIs added to the BN structure
3. A graphical notation that captures the sequence and type of abstraction operations, and thereby showing the link between the knowledge-base and the abstracted BN

Some of the abstraction operations in our method are based on existing techniques that have been mainly used for learning and inference problems. This paper emphasises the potential of these techniques for following a systematic approach to knowledge engineering. The compatible abstraction operations do not add CIs but they can make the BN structure increasingly complex by adding edges. Incompatible abstraction can be used to simplify the structure but they approximate the BN structure by adding new CI conditions. Trade-offs between the approximations and complexity must be considered carefully since some approximations can significantly change the underlying domain knowledge while simplifying the BN. Our abstraction methodology allows domain experts and knowledge engineers to evaluate these trade-offs by explicitly showing the changes in the underlying domain knowledge.

8.1 Comparison with Existing Abstraction Techniques

The main motivation from ABEL was its ability of reasoning in multiple levels of abstraction by showing the link between different levels (see Section 2.1). Our abstraction methodology can show the link between the underlying knowledge and simplifications in a similar way to ABEL. However, ABEL's abstractions are dynamically made at the inference stage based on its medical knowledge database, strongly guided by available data. ABEL uses its abstraction

mechanism to create causal models that explain the observed state of a patient. BNs are more sophisticated than the causal diagrams in ABEL as they are composed of variables with multiple states. A single BN can be instantiated in different ways according to observed values rather than creating different models for different observations. Therefore, our aim is to build a BN model that represents the best available knowledge and evidence, and our abstraction methodology is applied statically by knowledge engineers and domain experts at the development stage of the model.

The aim of our abstraction methodology is not aligned with idioms, network fragments, OOBNs, parent divorcing approach and similarity networks (see Section 2.2). These techniques aim to assist BN development and maintenance either by using semantically meaningful BN fragments or merging BN fragments or introducing nodes. However, the aim of our abstraction technique is to allow BN simplification without losing the link to domain knowledge supporting the BN. The aims of Wu and Poh's abstraction operations (Wu and Poh, 2000) and Srinivas's hierarchical approach (Srinivas, 1994) are similar to our abstraction methodology. However, Wu and Poh's operations can only be applied to limited and simple modelling tasks. The merged variables, for instance, must share only a single parent or child. The abstraction operations in this paper overcome these limitations as they can be applied to any node, edge or state in a BN. Srinivas uses node removal technique to model different levels of abstraction for a specific engineering problem. Our abstraction methodology is not developed for a specific problem, and contains a wider variety of abstraction techniques. We presented general properties of the abstraction techniques for BNs and use a medical case study only to illustrate the application of these techniques. In summary, our abstraction methodology overcomes the limitations of these studies as it is not developed for a specific problem, and its operations can be applied to any element of a BN.

8.2 Future Work

The case-study illustrates the application of the abstraction methodology to a real-world problem. Further case studies could focus on evaluating the practical impact of the abstraction methodology on BN development. The evaluation could be composed of two steps. The first step could assess the degree of knowledge that becomes unclear to users when BN models are simplified without using the abstraction methodology. The second step could evaluate the impact of the abstraction methodology to improve understanding of the BN.

The next stage in this research is to implement the abstraction operations in BN software such as AgenaRisk (Agena Ltd, 2013). The BN software, supplemented with the abstraction operations, would guide knowledge engineers and support BN development by showing the impacts of compatible abstractions and presenting the approximations resulting from incompatible abstractions. The abstraction operations could also be coupled with an evidence framework that shows evidence supporting every node, edge and state in a BN. As a result, the evidence framework could show how abstraction operations affect the underlying evidence. This would assist knowledge engineers by showing evidence that supports or contradicts with a particular abstraction operation.

Acknowledgments

The authors are grateful for the contribution of Mr Zane Perkins, Research Fellow at the Centre for Trauma Sciences, Queen Mary University of London, to the work described in this paper.

References

- Agena Ltd, 2013. AgenaRisk: Bayesian Network and Simulation Software for Risk Analysis and Decision Support [Online]. URL: <http://www.agenarisk.com> (accessed 7.6.13).
- Chang, K.C., Fung, R.M., 1990. Refinement and coarsening of Bayesian networks, in: Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI-90). Cambridge, MA, pp. 435–446.
- Choi, A., Darwiche, A., 2006. A variational approach for approximating Bayesian networks by edge deletion, in: Proceedings of the Twenty Second Conference on Uncertainty in Artificial Intelligence (UAI-06). Cambridge, MA, pp. 80–89.
- Dash, D., Druzdzel, M.J., 2003. Robust independence testing for constraint-based learning of causal structure, in: Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence. Acapulco, Mexico, pp. 167–174.
- Fenton, N., Neil, M., 2012. The Need for Causal Explanatory Models in Risk Assessment, in: Risk Assessment and Decision Analysis with Bayesian Networks. CRC Press.
- Heckerman, D., 1990. Probabilistic similarity networks. *Networks* 20, 607–636.
- Koller, D., Pfeffer, A., 1997. Object-oriented Bayesian networks, in: Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97). Providence, RI, pp. 302–313.
- Laskey, K.B., Mahoney, S.M., 1997. Network fragments: Representing knowledge for constructing probabilistic models, in: Proceedings of the Thirteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-97). Providence, RI, pp. 334–341.
- Laskey, K.B., Mahoney, S.M., 2000. Network Engineering for Agile Belief Network Models. *IEEE Trans. Knowl. Data Eng.* 12, 487–498.
- Margaritis, D., 2004. Distribution-free learning of graphical model structure in continuous domains. Technical report TR-ISU-CS-04-06, Department of Computer Science, Iowa State University.
- Neil, M., Fenton, N., Nielson, L., 2000. Building large-scale Bayesian networks. *Knowl. Eng. Rev.* 15, 257–284.
- Nielsen, T.D., Jensen, F.V., 2007. Bayesian networks and decision graphs. Springer.
- Patil, R.S., 1981. Causal representation of patient illness for electrolyte and acid-base diagnosis (PhD). MIT, Cambridge, MA.
- Patil, R.S., Szolovits, P., Schwartz, W.B., 1981. Causal understanding of patient illness in medical diagnosis, in: Proceedings of the Seventh International Joint Conference on Artificial Intelligence. Vancouver, BC, pp. 893–899.
- Pearl, J., 1988. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann.
- Renooij, S., 2010. Bayesian network sensitivity to edge-removal, in: Proceedings of the Fifth European Workshop on Probabilistic Graphical Models. Helsinki, Finland, pp. 233–240.
- Schwarz, G., 1978. Estimating the dimension of a model. *Ann. Stat.* 6, 461–464.
- Shachter, R.D., 1986. Evaluating influence diagrams. *Oper. Res.* 34, 871–882.

- Shachter, R.D., 1988. Probabilistic inference and influence diagrams. *Oper. Res.* 36, 589–604.
- Shachter, R.D., 1990. Evidence Absorption and Propagation through Evidence Reversals, in: *Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence (UAI-90)*. Windsor, ON, pp. 173–190.
- Srinivas, S., 1994. A probabilistic approach to hierarchical model-based diagnosis, in: *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence (UAI-94)*. Seattle, WA, pp. 538–545.
- Van Engelen, R.A., 1997. Approximating Bayesian belief networks by arc removal. *IEEE Trans. Pattern Anal. Mach. Intell.* 19, 916–920.
- Wellman, M.P., Liu, C., 1994. State-space abstraction for anytime evaluation of probabilistic networks, in: *Proceedings of the Tenth International Conference on Uncertainty in Artificial Intelligence (UAI-94)*. Seattle, WA, pp. 567–574.
- Wu, X., Poh, K.L., 2000. Decision-model construction with multilevel influence diagrams. *Knowl. Eng. Rev.* 15, 233–256.