Matilda: A Visual Tool for Modeling with Bayesian Networks

T. Boneh,[1,†] A. E. Nicholson,[1,*] E. A. Sonenberg[2,‡]
[1]*Faculty of Information Technology, Monash University, 3800, VIC, Australia*
[2]*Department of Information Systems, University of Melbourne, 3010, VIC, Australia*

A Bayesian Network (BN) consists of a qualitative part representing the structural assumptions of the domain and a quantitative part, the parameters. To date, knowledge engineering support has focused on parameter elicitation, with little support for designing the graphical structure. Poor design choices in BN construction can impact the network's performance, network maintenance, and the explanatory power of the output. We present a tool to help domain experts examine BN structure independently of the parameters. Our qualitative evaluation of the tool shows that it can help in identifying possible structural modeling errors and, hence, improve the quality of BN models. © 2006 Wiley Periodicals, Inc.

1. INTRODUCTION

1.1. Bayesian Networks

Bayesian networks are widely accepted in the Artificial Intelligence community as intuitively appealing, practical representations of knowledge for reasoning under uncertainty.[1] A Bayesian Network (BN) is a representation of a joint probability distribution over a set of statistical variables. It consists of a qualitative part, a graph structure, and an associated quantitative part, the parameters. To date, knowledge engineering support has focused on the quantitative part. However, it has been recognized that current tools and approaches are not sufficient to support domain experts in the construction of a network for their domain (Ref. 2, Chapter 9). Herein, we explain the design of a novel support tool for BN knowledge engineering and present initial evaluation results. The tool, Matilda, is designed to help domain experts determine whether the assumptions about the domain that are

*Author to whom all correspondence should be addressed: e-mail: ann.nicholson@infotech.monash.edu.au.
[†]e-mail: tali.boneh@infotech.monash.edu.au.
[‡]e-mail: l.sonenberg@unimelb.edu.au.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 21, 1127–1150 (2006)
 Published online in Wiley InterScience (www.interscience.wiley.com). • DOI 10.1002/int.20175

implicit in the graphical structure of the BN are consistent with their understanding of that domain.

Formally, the graph structure is a directed acyclic graph (DAG) and represents the structural assumptions of the domain, that is, the variables comprising the domain, the possible values of these variables, and the dependence–independence relationships between the variables. Each node represents one of the variables; an arc between two nodes represents possible direct dependence between the two variables and the absence of an arc between two nodes represents (conditional) independence between the two variables. We say that two variables *A* and *B* are *independent*[a] *if knowing the value of A* does not change the probability of the value of *B* and vice versa, and that *A* and *B* are *conditionally independent given C* if there is a variable *C* such that if the value of *C* is known then *A* and *B* are independent. Note that dependency/independency between two variables can change when information is introduced or retracted. The parameters associated with each node form a conditional probability table (CPT) that encodes the strength of the dependency between the node and its parents. Once both the qualitative and the quantitative parts are defined, given evidence for known variables, posterior probabilities of unknown variables, which are not (conditionally) independent of the known variables, can be calculated.

1.2. Examples

Example 1: The Missing Car Problem. Consider the following example: John and Mary Nguyen arrive home after work to find that their second car is not in the garage. Two explanations occur to them: either the car has been stolen or their daughter Sam has borrowed the car without permission. The Nguyens know that the rate of car theft in their area is about 1 in 2000 each day, and that if the car was stolen, there is a 95% chance that the garage will show signs of forced entry. (There is nothing else worth stealing in the garage, so assume that if the car is not stolen, the garage will not show signs of forced entry.) The Nguyens also know that Sam borrows the car without asking about once a week, and that Sam has a busy social life, so even if she didn't borrow the car, there is a 50% chance that she is out.

The structural assumptions of the domain are concerned with how to represent these facts by random variables, their possible values, and the relationships between them. *Graphical modeling* is the construction of the BN graph according to the model builder's structural assumptions about the domain.

We can represent the structural assumptions of the Missing Car domain using four nodes *CarStatus*, *CarMissing*, *SamOut*, and *ForcedEntry*. The last three nodes have two possible values: True and False. The node *CarStatus* has three possible values: *Home*, *Borrowed*, and *Stolen*. In this modeling we exploit the assumption that *Borrowed* and *Stolen* are mutually exclusive facts and therefore can be values of the same node. Figure 1 illustrates the BN we constructed for the missing car domain.

[a]Also referred to as *marginally independent*.

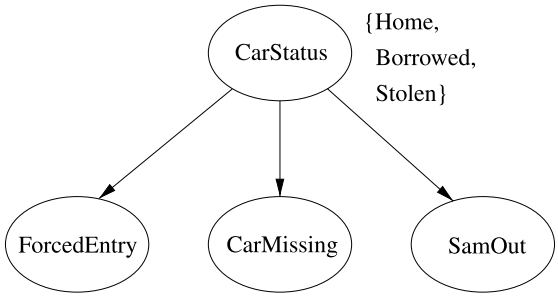


Figure 1. A Bayesian network for the missing car problem.

This structure represents various independence assumptions about the domain. If we know the value of the *CarStatus* then *SamOut* and *CarMissing* are independent, that is, knowing the value of the one does not change our belief in the value of the other. This property, although not stated explicitly in the description, can be inferred using commonsense reasoning. In the network it is represented by the absence of an arc between the nodes *SamOut* and *CarMissing*. (Similarly for the pairs of nodes {*SamOut* and *ForcedEntry*} and {*CarMissing* and *ForcedEntry*}.)

Example 2: Fire alarm problem. Consider the following example. We receive a phone call saying that everyone is leaving the building. The call can come from three different sources. A security center gets a report from a special sensor in the building. If the sensor reports "Leaving" the security center calls us. It could also be a call from kids who are playing practical jokes (mainly during the holidays) as well as from seriously concerned people who notice smoke coming out of the building (mainly after work hours). The sensor is noisy. It sometimes does not report when everyone is leaving and sometimes reports leaving for no reason. If the fire alarm goes off, that causes people in the building to leave. The fire alarm can go off either because there is a fire or because someone tampers with it. The fire also causes smoke to rise from the building. In winter, fireplaces in the building can also cause the presence of smoke.

A BN for this fire alarm domain, (Ref. 2, Figure 9.4, with permission, taken from Ref. 3, which extended one used in Ref. 4) is shown in Figure 2. More complex than the missing car BN, it will also be used throughout this article for illustrative purposes.

1.3. Graphical Modeling

To represent a joint distribution, arcs can join all possible pairs of variables, creating a very complex network. Because BNs exploit the structural assumptions of the domain, not all node pairs need to be joined by arcs, allowing a relatively compact representation of a complex joint probability distribution. This capacity to simplify the representation is what makes BNs a powerful technology and makes

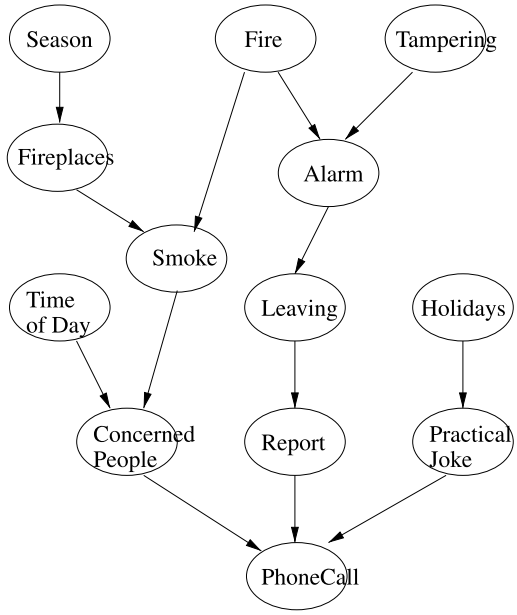


Figure 2. A Bayesian network for the fire alarm problem.

the graphical structure of a network very important.[5] The feasibility of a BN is a function of both (1) the number of probabilities in the CPT that have to be elicited or learned and (2) the number of inferential steps in updating the posterior probabilities. It should be noted that there may be no single correct graphical representation of the domain, as the same joint distribution can be represented by different BNs.[1]

It has long been accepted that, for the purpose of eliciting graphical modeling decisions, the formal notions of directional conditional independency can be replaced by informal notions of causation and influence, and the considerations can be kept locally.[1] This supposedly enables domain experts to intuitively express their knowledge when constructing the graph. In practice, domain experts have required the support of BN experts to model their domains correctly.[5,6] Mistakes in the graph construction that may not be obvious to domain experts can have crucial effects on model construction and maintenance (as the structure affects the number of parameters that need to be assessed) and tractability of inference, on the one hand, and the accuracy, simplicity, and explanatory power of the output on the other hand.[5,6]

Ample research, literature, and tool support have focused on inference algorithms, obtaining the CPTs, and assessing the parameters by examining how the model matches experts' expectations. In contrast, the acquisition and the assessment of the graph structure have received much less attention. Exacerbating the difficulty of graphical modeling is the fact that current BN software tools require the definition of both the qualitative and quantitative parts (i.e., putting in the

probabilities) *before* the user can assess the relationships between variables. An exception is the Hugin software's new facility for d-separation examination, but the development of Matilda proceeded Hugin's announcement. A comparison of Matilda to Hugin's functionality is made in Section 4.3.

The structure building process is typically iterative and based on trial and error, but existing tools do not provide the user with practical support for this process independent of the parameters. Every time changes are made to the structure, new probabilities need to be elicited—often a difficult and time-consuming process—before the new model can be evaluated by methods such as sensitivity analysis (e.g., Ref. 7) and explanation methods (e.g., Ref. 8).

A BN is a DAG of relationships between random variables where directed arcs between nodes represent possible conditional dependencies. A key graphical property, that of *d-separation*, turns out to be very useful for analyzing BN models.

1.4. D-Separation

In a BN, questions about dependence among variables can be answered by studying the graph alone. It can be shown that the graphical notion termed d-separation (from *direction-dependent separation*) corresponds to the notion of conditional independence. To define d-separation, we need some additional definitions first. For a directed graph, a *path* between two sets of nodes **X** and **Y** is any sequence of nodes between a member of **X** and a member of **Y** such that every adjacent pair of nodes is connected by an arc (regardless of direction) and no node appears in the sequence twice. A path is *blocked*, given a set of nodes **E**, if there is a node Z on the path for which at least one of three conditions holds:

(1) Z is in **E** and Z has one arc on the path leading in and one arc out (chain).
(2) Z is in **E** and Z has both path arcs leading out (common cause).
(3) Neither Z nor any descendant of Z is in **E**, and both path arcs lead in to Z (common effect).

A set of nodes **E** *d-separates* two other sets of nodes **X** and **Y** if every path from a node in **X** to a node in **Y** is blocked given **E**. If **X** and **Y** are d-separated by **E**, then **X** and **Y** are *conditionally independent* given **E** (given the Markov property).[1] Examples of the three blocking situations (simplified by using single nodes rather than sets of nodes) are shown in Figure 3 (taken from Ref. 2); note that the evidence nodes **E** are shaded.

Let us now see instances of these blocking conditions leading to d-separation in our example domains, "missing car" (Figure 1) and "fire alarm" (Figure 2).

- Blocking condition 1. This situation does not appear for the missing car BN, as there is no directed chain structure. In the fire alarm BN, there are many such chains—one example, given evidence for *Alarm* (or *Leaving* or *Report*), *PhoneCall* is d-separated from *Tampering*.
- Blocking condition 2. For the missing car BN, given evidence about *CarStatus*, the nodes *SamOut* and *CarMissing* are d-separated by the node *CarStatus*. This means that if we already know the status of the car, for example, that it is stolen, then getting

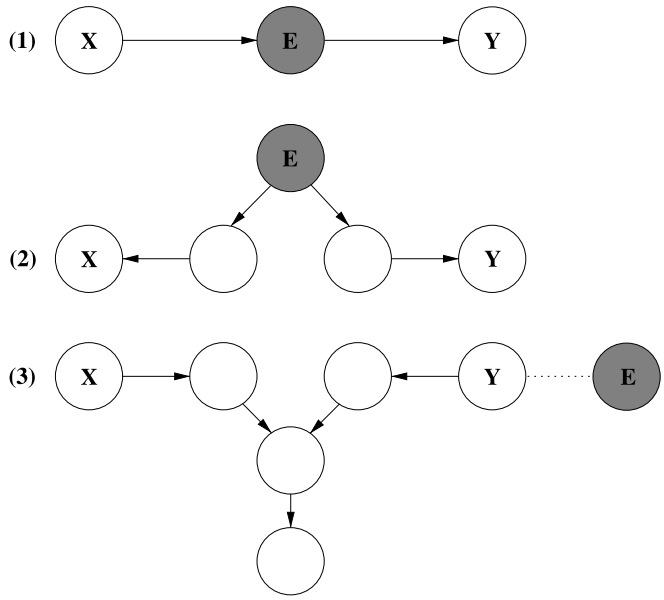


Figure 3. Examples of the three types of situations in which the path from **X** to **Y** can be blocked, given evidence **E**. In each case, **X** and **Y** are *d-separated* by **E**.

information about Sam being out will not change the posterior probabilities for the car being missing.

- Blocking condition 3. Again, this situation does not appear for the missing car example, as there is no common effect structure. In the fire alarm BN, there are a number of common effect substructures: *Fireplace* ← *Smoke* → *Fire*, *Fire* ← *Alarm* → *Tampering*, *TimeOfDay* ← *ConcernedPeople* → *Smoke*, and, finally, *PhoneCall* has three possible direct causes—*ConcernedPeople*, *Report*, and *PracticalJoke*. With no evidence added to the network, these give rise to many pairs of d-separated nodes; examples include *Fireplace* and *Fire* (parents in the common effect structure), *TimeOfDay* and all the following nodes *Season*, *Fireplace*, *Smoke*, *Fire*, *Tampering*, *Alarm*, *Leaving*, *Report*, *PracticalJoke*, and *Holidays*. Another way of looking at this is that *TimeOfDay* is d-separated from all nodes other than *ConcernedPeople* (its child) and *Phonecall* (on a chain, hence, blocking condition 1 holds).

It is important to realize that as the network structure becomes more complex, as for the fire alarm example, determining whether nodes are d-separated also becomes more complex. For example, given evidence for *Fire* and *PhoneCall*, *Smoke* is not d-separated from *Alarm*: blocking condition 2 applies (common cause, with *Fire* evidence), but blocking condition 3 does not (common effect structure, but *PhoneCall* evidence). Note that Matilda's visualization of d-separation in the fire alarm problem is provided after the presentation of the tool, in Section 3.2.

D-separation is a crucial concept, because if the independencies are modeled incorrectly, the probabilistic reasoning process, given new evidence, will be incorrect. The working hypothesis of the research reported in this article is that the

understanding of d-separation can be supported intuitively by visualization and verbal explanation and, hence, that displaying this information to the domain expert will help in the assessment of whether the graph is faithful to the domain expert's understanding of his or her domain.

1.5. Article Outline

Our approach to exploring BN structure, which includes visualization and verbal explanation of relationships between variables, independent of the parameters, is described in Section 2. We have implemented the approach in a software tool called Matilda (Section 3), which has been qualitatively evaluated in three types of case studies—toy networks, an expert-constructed network, and a learned network. In Section 4, we present and analyze the results from these evaluations, which show the approach to be useful. It can help the domain expert gain a better understanding of the model, the domain, and BN technology. Accordingly, it can support the construction of a BN structure independent of the quantitative part of the model and potentially reduce the involvement of a BN expert in the network construction process. Section 5 contains some concluding remarks and pointers to future work.

2. GRAPHICAL MODELING

2.1. Problems

Elicited structures are assessed by how well their topology represents the independency assumptions of the domain, namely, how well the representation meets the experts' expectations. It is well known that, in order to check independency assumptions, the d-separation criterion can be used.[1,9,10] As discussed at more length later (Section 4.2), when the "Missing Car" example was given to students as an assignment, almost all chose to represent the domain with five Boolean nodes: *CarStolen*, *SamBorrowed*, *ForcedEntry*, *CarMissing*, and *SamOut*, and then to construct the structure in Figure 4.

The structure in Figure 4 is incorrect, as it indicates that, without any other information, the two nodes *CarStolen* and *SamBorrowed* are independent.

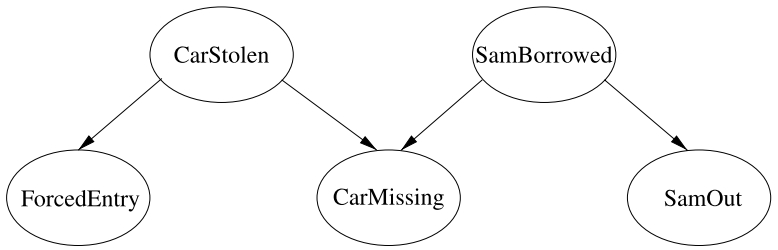


Figure 4. An incorrect BN for the "missing car problem" commonly produced by students.

However, these two nodes are logically related, namely, knowing the value of one determines the value of the other. In fact, particular instantiations of the nodes are mutually exclusive; if Sam has borrowed the car, it could not have been stolen from the garage and vice versa. More than 50% of the students (13 out of 21) had made graphical modeling errors, according to our interpretation of the set problem. We propose that this high level of error is not easily explained by (for example) ambiguities in the problem description, but points to a fundamental problem with translating domain knowledge into the BN formalism. However, by using only the syntactic d-separation criterion, that is, without knowing anything about the parameters, we can get an indication for possible problems in this structure. In this way, working with d-separation can be useful in exposing the independence assumptions that are represented by the graph.

2.2. Previous Work

Several methods for using d-separation for the investigation of the graph have been described in the literature. First, a list of all d-separation situations can be produced by the BN expert for reviewing by the domain experts. However, the difficulty with this approach is that producing such a list is an exponential problem in the number of nodes, as it involves checking all subsets of nodes in the graph. Second, the procedure of testing d-separation can be handled by visual inspection of the graph (Ref. 1, p. 118). However, although this may be the case for BN experts when the network at hand is small, it is far from clear how to proceed to someone who is not a BN expert.[6,11] Third, there are oracles that implement d-separation, for example, Wimberly's applet.[12] The problem is that simply telling the user that two nodes are d-separated is, again, not very helpful unless the user is a BN expert. A fourth method is to assess dependency assumptions by presenting the user with case descriptions. These case descriptions describe the variables of interest and the relationships between them as represented by the graph structure. They present a question for the domain expert to affirm or deny, all in "a story style" and in the domain terminology. Here, the BN expert chooses the relationships to be investigated. This method is very time consuming,[11] as every situation needs a story tailored by the BN expert, with any change needing a new tailored story.

The problems with existing methods can be summarized thus:

(1) The user must be a BN expert to apply the method and/or to understand its output.
(2) The user is provided with too much data to process.
(3) The method is too time consuming.

Overall, although it is widely accepted that it is important to review and assess the independency assumptions that are represented by the topology of the graph, possibly by using the d-separation criterion, the question of how to (efficiently) assess the graph topology is still open. Existing methods are not appropriate for the iterative process of constructing a BN.

2.3. Our Approach

The basic idea is to visually display the independence and conditional independence relationships between variables, using the d-separation criteria, so that domain experts can assess whether or not they are correct.

To overcome the problems identified above with existing methods based on d-separation, we propose an approach that includes:

(1) Visualization and verbal explanation that are intuitive to the user and provide understandable explications for domain experts of the implications of the decisions they make. Thus, it will be suitable for users who are not BN experts.
(2) An interface for the domain expert to choose nodes and relationships of interest to be explored. This enables users to make decisions about what should be investigated, motivated by interest and not by ad hoc exploration of relationships, preventing a flood of information.
(3) Functionality that is easy to reapply to the whole structure or only to the latest changes to the structure, which makes it appropriate for the use within an iterative process.

We want to restrict the information to users' requests yet to support as many users as possible. To achieve this balance we define three types of requests:

Type 1: When does a node become d-separated from the nodes in the rest of the network (i.e., irrelevant) (relationships between a node and the rest of the network)?
Type 2: Given two nodes, is there information that d-separates them (relationships between two nodes)?
Type 3: Given some information, what happens to the relationships between nodes (relationships between groups of nodes)?

We now illustrate these request types with examples for the fire alarm problem. A more formal description is given in the following section.

Suppose that one task of a decision maker using the reasoning system is trying to decide whether a particular piece of additional information should be obtained. For example, whether it is worth sending someone to check the security sensor that is supposed to report someone leaving; if it is working and has not led to a call from the security center, then this would amount to obtaining evidence node *Report = False*. Combinations of other evidence might render the *Report* evidence irrelevant. The knowledge engineer building the network can use request type 1 to check these relationships, that is, to check what other evidence makes the *Report* evidence irrelevant.

Another scenario the knowledge engineer might want to investigate is whether there are situations where a phone call about a possible fire (i.e., evidence *PhoneCall = True*) will not change the probability that the *Alarm* has gone off. Here, request type 2 can be used to see whether there is information that d-separates the nodes *Alarm* and *PhoneCall*.

Finally, the knowledge engineer might investigate the situation where it is already known that there is *Smoke* and there is *Fire*. Request type 3 allows him to find out, given this information, what other evidence is irrelevant and will not

change the probability of a *PhoneCall* being received, that is, the set of nodes that are d-separated from *PhoneCall*.

3. MATILDA: A SUPPORT TOOL FOR GRAPHICAL MODELING

We utilized these concepts to design a visual tool, called *Matilda*, that supports the comparison between the structural assumptions of the domain and the graphical modeling decisions. It operates on the graph structure and helps reveal potential misrepresentations of the domain by the graph. The tool has a window interface and uses the structure as the main display. Matilda allows the user to choose nodes and then to choose one of several types of independencies to be displayed. Upon request, Matilda visualizes the response to the request by highlighting the appropriate nodes and displays a verbal explanation of the situation. For example, the result of the investigation of the relationships between nodes in the missing car BNs (see Figures 1 and 4) is given in Figure 5, as discussed below.

3.1. Design Choices

3.1.1. Language and Terminology

The potential users of a knowledge engineering support tool may have diverse backgrounds, so the language and terminology used need to be as simple and intuitive as possible. To avoid the use of BN technical terminology, terms from graph theory such as "parents" and "ancestors" are replaced by the terms *direct causes* and *causes*, respectively. The term "d-separation" is replaced by a more intuitive sounding notion, that of *blocking*. A nonminimal d-separating set found among parent nodes is called a *simple blocking set*, whereas a minimal one is called an *essential blocking set*. We chose this focus on the parent nodes as they model the most important direct relationships with the node of interest. A minimal d-separating set (from which no node can be removed without impairing the d-separation property) found anywhere in the network (i.e., not necessarily among the parents) is called a *minimal blocking set*.

Given two nodes of interest, the simple and essential blocking sets are found from parent nodes of one of the selected nodes. For example, in the fire alarm BN, suppose the two selected nodes are *Alarm* and *PhoneCall*. A simple blocking set is the set of parents of *PhoneCall*, that is, {*ConcernedPeople*, *Report*, and *PracticalJoke*}, whereas the essential blocking set is {*ConcernedPeople*, *Report*}. The minimal blocking sets contain nodes anywhere in the network and represent the minimal information required to stop the two nodes of interest from influencing each other. For the same selected nodes from the fire alarm example, possible minimal blocking sets are: {*Fire*, *Report*}, {*Fire*, *Leaving*}, {*Report*, *ConcernedPeople*}, {*Report*, *Smoke*}, {*ConcernedPeople*, *Leaving*}, {*Leaving*, *Smoke*}. Thus, as has been suggested,[13] these sets may be helpful in identifying modeling errors. Note that all three types of sets may refer to the same set in the graph. However, when they refer to different sets, each provides the user with a different situation in the domain to consider, and one may be clearer or more helpful than the others.

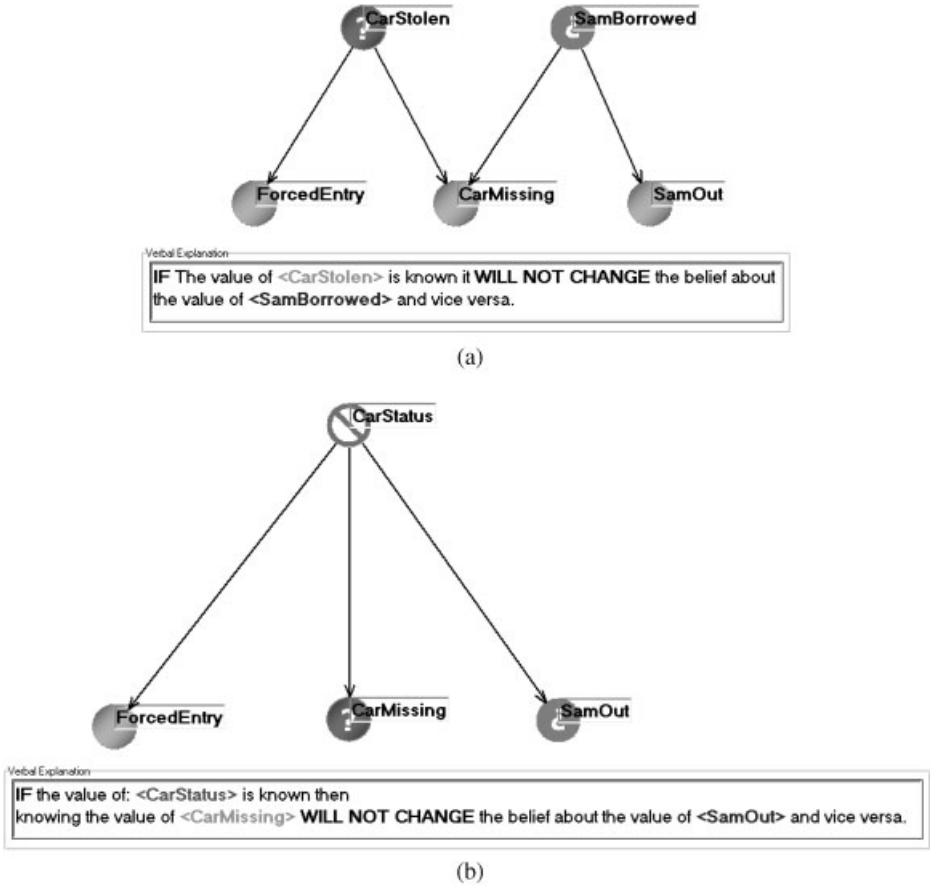


Figure 5. Matilda's visualization of the relationship between nodes (a) *CarStolen* and *SamBorrowed* and (b) *CarMissing* and *SamOut*, given evidence about *CarStatus* in the missing car BNs.

Recall that the fundamental reasoning task of Bayesian networks is to incorporate new information about some nodes by updating the posterior probabilities of other nodes. These posterior probabilities are viewed as the new beliefs about the values of these nodes. The intuitive meaning of d-separation between two nodes is that adding information about one node and performing an update will not change the beliefs in the other node.

To describe the general relation "*X* is d-separated from *Y* by *Z*" the *X*, *Y*, and *Z* nodes are given a specific temporary role in the model. *X* become the *query nodes* (the nodes we want to reason about), *Y* become the *observation nodes* (the nodes we may observe), and *Z* are the *known nodes* (the nodes the value of which we already know).

Using these terms, the relation "*X* is d-separated from *Y* by *Z*" is then described as: *"If the known nodes are observed, then knowing the value of the observation*

nodes will not change the belief about the value of the query nodes and vice versa." The notion "vice versa" has been added to provide the symmetry of the d-separation relation, otherwise lost by giving each set of nodes a role.

3.1.2. Visualization

The main focus of our approach is on the visual display, where the user sees the model, poses a query using the interface, and then sees the result displayed on the graph. This focus is based on the observation that visual query formulation and visual display of results can be combined with the strategies of direct manipulation, thereby utilizing human perceptual skills.[14] We suggest that this visual focus also enhances the users' insight into the notion of d-separation and their understanding of the graph structure and its implications.

From this visual focus arises the need to visually represent the notion of d-separation, that is, to visually represent the relation "*X* is d-separated from *Y* given *Z*." Our approach is to represent d-separation visually by using different colors and symbols for the nodes involved. Matilda uses algorithms for computing the various d-separating sets taken from the literature (e.g., Refs. 15 and 16); details are given in Ref. 3.

The implemented system's visualization of the relation "*X* is d-separated from *Y* given *Z*," is shown in Figure 6. The blocking relationship is shown using a common symbol for "stop" or "not allowed" (in red in Matilda) for the *Z* node, with query nodes *X* indicated by "?" (purple in Matilda), and known nodes *Y* by an upside down question mark, "¿" (blue in Matilda). All paths between two nodes can be highlighted in red. These visualization choices are based on CUErgo guidelines[17]; however we acknowledge that specific choices such as the actual colors are fairly arbitrary.

Despite the fact that colors and signs may be clear and intuitive from a designer's perspective, they require time for the user to get used to. Therefore, we

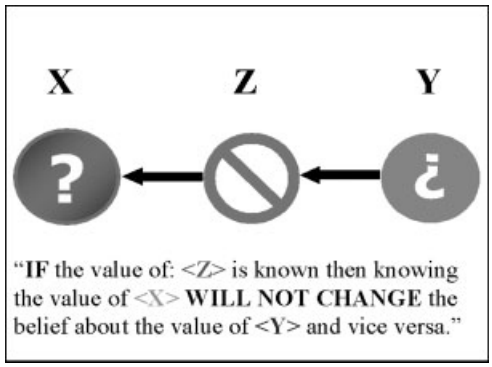


Figure 6. Matilda's visualization of the relation "*X* is d-separated from *Y* given *Z*," with corresponding verbal explanations.

supplement the visual display with an explanatory text shown in a separate text box (see boxes at bottom of Matilda output in Figure 6). This text is generated by combining text fragments and by incorporating node names in the same colors as the respective nodes, which ties it to the visual explanation. Using the terminology described above, the free text is: "IF the value of: ⟨Z⟩ is known, then knowing the value of ⟨X⟩ WILL NOT CHANGE the belief about the value of ⟨Y⟩ and vice versa."

3.2. Examples

Missing Car Example. Figure 5a shows Matilda's visualization of the relationship between nodes *CarStolen* and *SamBorrowed*. This is an example of d-separation through blocking condition 3 and illustrates how Matilda can highlight possibly incorrect relationships. Figure 5b shows Matilda's visualization of the (correct) relationship between nodes *CarMissing* and *SamOut*, given *CarStatus*, an example of d-separation through blocking condition 2.

Fire Alarm Example. Figure 7 shows Matilda's visualization of d-separation in the fire alarm BN, as described in Section 1.4.

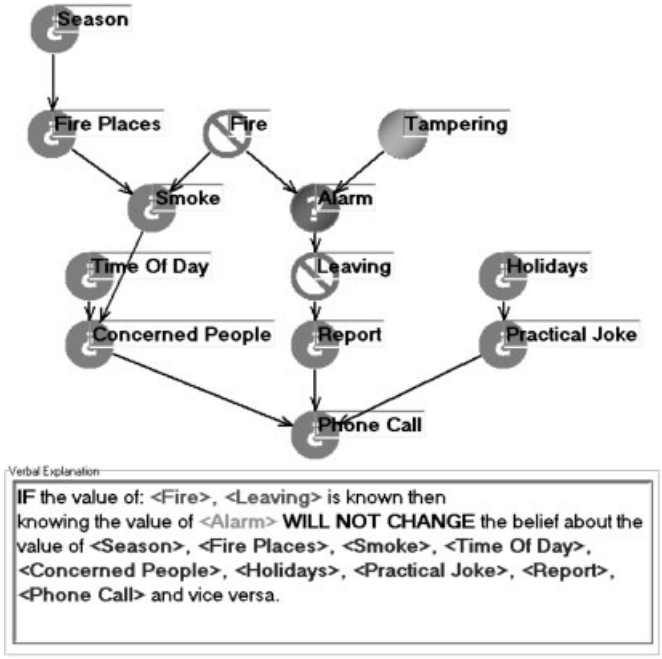
3.3. Graph Assessment Using d-Separation

Matilda allows users to ask direct questions with regard to specific information they are interested in, through the three types of requests described in Section 2.3. The BN for the fire alarm domain shown in Figure 2 will be used to illustrate the different request types and the corresponding answers offered by Matilda.

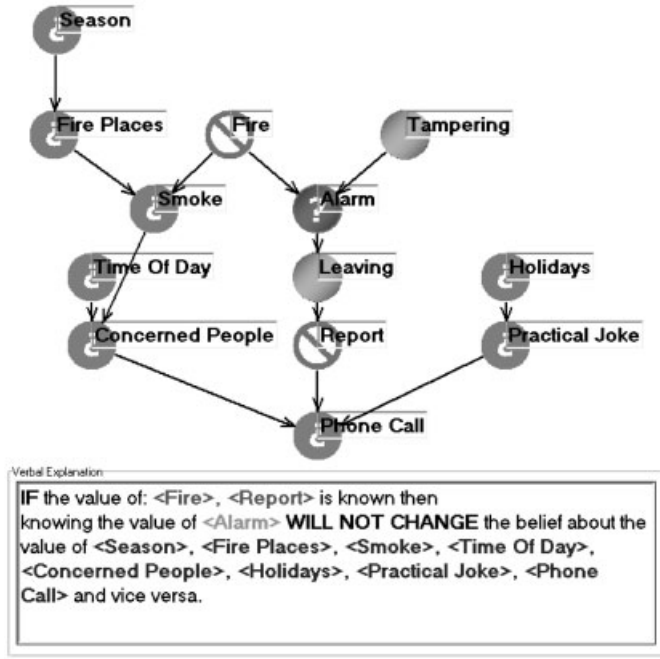
Type 1. When does a node become irrelevant? Matilda displays the relationships between one node and the rest of the nodes in the network. This option allows the user to select a single node X and ask for a set of nodes that d-separates this node from the rest of the nodes in the structure. Matilda highlights only one set, the so-called *Markov Blanket*,[1] which consists of the parents, the children, and the children's parents.

Type 1 Example: Select $X = Report$. The Markov blanket for this node is the set {*Leaving* (parent), *PhoneCall* (child), *PracticalJoke*, *ConcernedPeople* (child's parents)}. The verbal explanation is: "IF the value of ⟨Leaving⟩, ⟨PhoneCall⟩, ⟨PracticalJoke⟩, ⟨ConcernedPeople⟩ is known, then knowing the value of ⟨Report⟩ WILL NOT CHANGE the belief about the value of any other node and vice versa." Matilda's output for this example (without the verbal explanation) is shown in Figure 8; selected node *Report* is displayed with the upright "?" with the blocking set making it irrelevant to the rest of the network, indicated by the "¿" symbol.

Type 2: The relationship between two nodes. This option allows the user to ask "What information d-separates two selected nodes?" by selecting two nodes X and Y and choosing between the three types of blocking sets (our substitute terminology for d-separation): simple, essential, and minimal blocking sets (as described

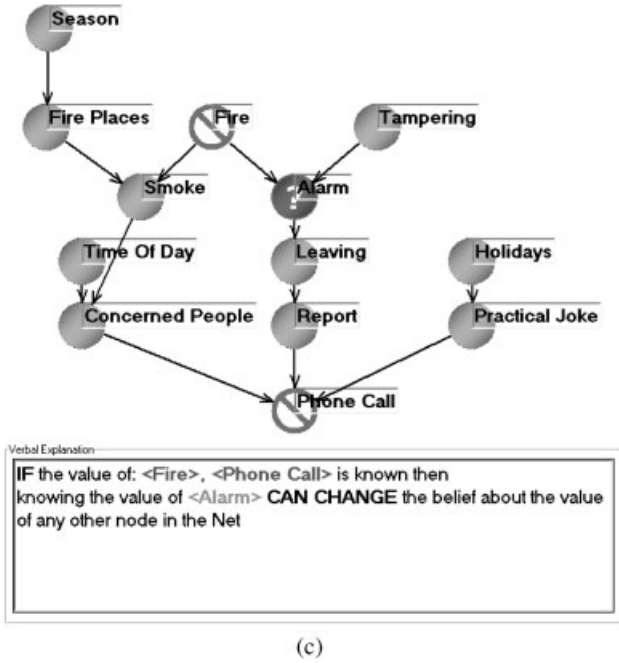


(a)



(b)

Figure 7. *Continues on next page.*



(c)

Figure 7. Matilda's visualizations of d-separation in the fire alarm examples from Section 1.4.

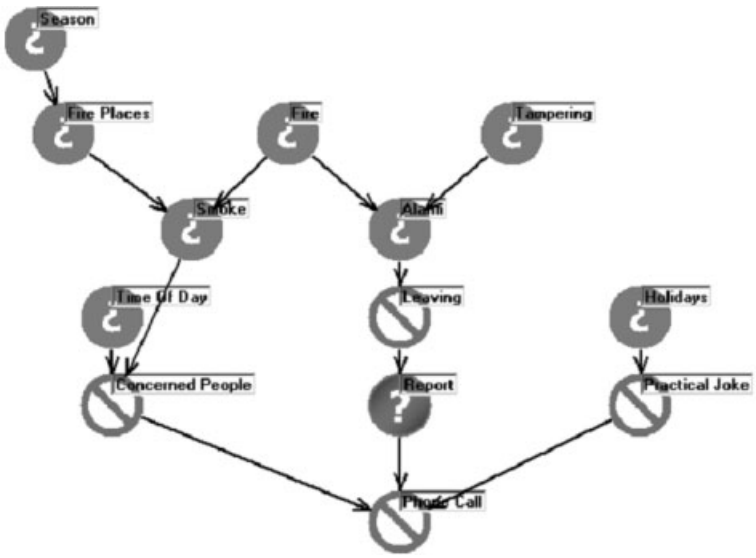


Figure 8. Matilda's visualizations in the fire-alarm example. Type 1: a Markov blanket $Z = \{Leaving, PhoneCall, PracticalJoke, ConcernedPeople\}$ for $X = Report$.

above). Matilda then computes the selected type of d-separating set and highlights it using the blocking symbol (as described in Section 3.1). If there are multiple minimal sets, a list of all minimal sets is displayed in a separate box, and users can choose which one they want to investigate.

Type 2 Example: Recall that above we gave examples of the simple, essential, and minimal blocking sets for the fire alarm BN, with selected nodes $X =$ *Alarm*, Y = *PhoneCall*. Matilda's output for this example (without the verbal explanation box) is shown in Figure 9. Note that all the minimal blocking sets are shown in a separate box. The possible minimal blocking set {*Leaving*, *Smoke*} has been selected in that box, and Matilda displays the blocking symbol on those two nodes in the displayed BN. Matilda's verbal explanation for this example is: "IF the value of ⟨Leaving⟩, ⟨Smoke⟩ is known, then knowing the value of ⟨Alarm⟩ WILL NOT CHANGE the belief about the value of ⟨PhoneCall⟩ and vice versa."

Type 3. Given some information, what happens to the relationships between nodes? Here, Matilda displays the relationships between *sets* of nodes. This option allows the user to select a set of nodes X (the query nodes) and a set of nodes Z (the prior information) and request the set of all Y nodes that are d-separated (blocked) from X. Matilda highlights all the nodes Y in response.

Type 3 Example: Select $X = \{PhoneCall\}$, $Z = \{Smoke, Fire\}$. The nodes that are d-separated from *PhoneCall* by *Smoke* and *Fire* are *FirePlaces* and *Seasons*. Matilda's output for this example (without the visual explanation box) is

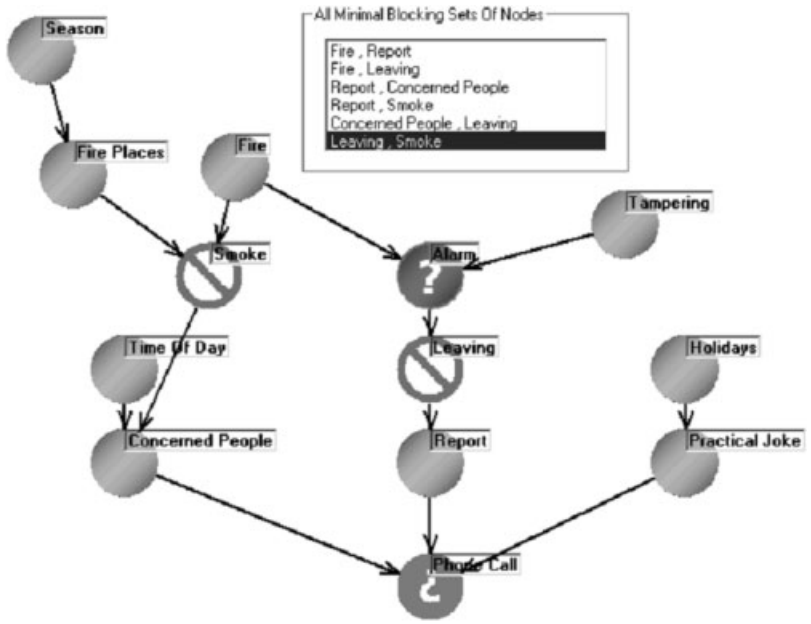


Figure 9. Matilda's visualizations in the fire-alarm example. Type 2: $X = \{Alarm\}$, $Y = \{PhoneCall\}$. A possible minimal blocking set {*Leaving*, *Smoke*}.

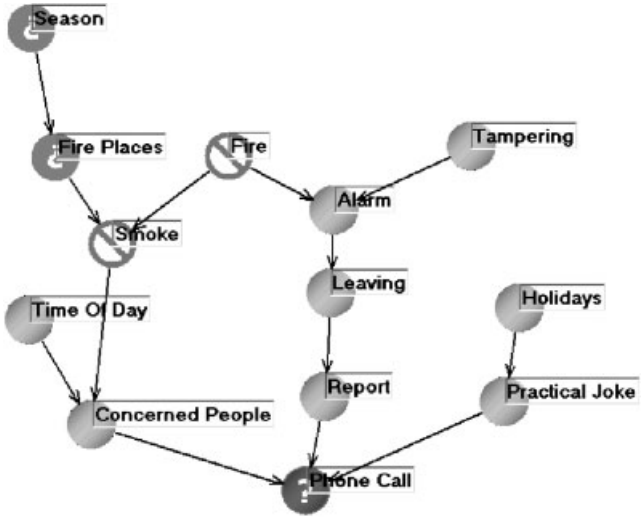


Figure 10. Matilda's visualizations in the fire-alarm example. Type 3: $X = \{PhoneCall\}$, $Z = \{Smoke, Fire\}$ selected, then $Y = \{Season, Fireplaces\}$.

shown in Figure 10, visualizing the relationship that, given information about *Smoke* and *Fire*, getting a *PhoneCall* will not have any effect on the probabilities for *Season* and *Fire*. Matilda's verbal explanation of this example is: "IF the value of: ⟨Fire⟩, ⟨Smoke⟩ is known, then knowing the value of ⟨PhoneCall⟩ WILL NOT CHANGE the belief about the value of ⟨Season⟩, ⟨FirePlaces⟩ and vice versa."

These three request types reveal information with regard to relationships between one node and the rest of the network, two nodes, and sets of nodes respectively. They also represent different types of functionality. In the first, the user chooses the node *X*, the set *Y* is defined as the rest of the nodes in the graph, and Matilda finds the set *Z*. In the second, the user chooses the nodes *X*, *Y* and Matilda finds the set(s) *Z*. In the last question the user chooses the sets *X*, *Z* and Matilda finds the set *Y*.

4. EVALUATION

4.1. Methodology

We have undertaken a qualitative evaluation of our approach and Matilda through case studies.[18] Our evaluation of Matilda was focused on descriptions of the investigation of the structure and on exploration of the tool functionality, goals, and results. The aims of the qualitative evaluation sessions were (1) to examine whether the tool does helpfully visualize the relationships within the graph and, hence, whether it is useful in the knowledge engineering process and (2) to explore the effectiveness and usefulness of the functionality provided by the tool.

We have used three different methods of data collection in three different settings. In the first setting, one of the researchers assessed so-called toy networks using Matilda as a preliminary "reality check." In the second setting, a domain expert used Matilda to analyze a BN for his previously constructed "real-life" domain by working through a structured questionnaire. In the third setting, a domain expert analyzed a learned network of her domain through an unstructured interview. These methods were applied to examine the feasibility of reducing the involvement of a BN expert in the knowledge engineering of a model and to test different aspects of the knowledge engineering process.

4.2. Results

Detailed analyses are available elsewhere.[19] Here we present the highlights.

Case Study 1: Analysis of toy networks. Matilda was used to analyze so-called toy networks, built by students as part of an assignment in a graduate university course.[19] The students were asked to create their own domain, to describe it in an English text, and to construct a BN for that domain. The assignment was given after the main principles of constructing a BN were explained in class and examples were given. The simplicity of the domains, and therefore the networks, made them useful for analyzing and for understanding the problems that may arise when constructing a BN. Using Matilda, we analyzed 19 networks developed for the students' own domains and identified one or more potential problems in 15 of them. We found that the graphical modeling problems could naturally be categorized into three groups, corresponding to different graphical structures, and Matilda was helpful to us in identifying problematic models:

(1) Incorrect independence between causes of common effects, as reported in Ref. 20
(2) Incorrect conditional independency between common effects, as described in Ref. 21
(3) Incorrect conditional independency between variables in different levels/parts, which includes all other possible structures.

Case Study 2: Analysis of previously constructed BN. Using Matilda, a domain expert investigated a network he constructed to represent his domain. The expert had been working with BNs for about 2 years.

Initially, the domain expert constructed, with the help of a BN expert,[b] a large network structure that included 11 nodes, 23 arcs, and over 70,000 parameters. As a result of a trade-off process, they reduced the number of variables, states, and arcs and then divided the model into three simplified submodels. Next, the parameters of each submodel were elicited and the three submodels were recombined into a new model. In the evaluation session, the domain expert examined one of the three submodels using Matilda, shown in Figure 11a. This

[b]Note that the BN expert in this case was not one of the authors of this article.

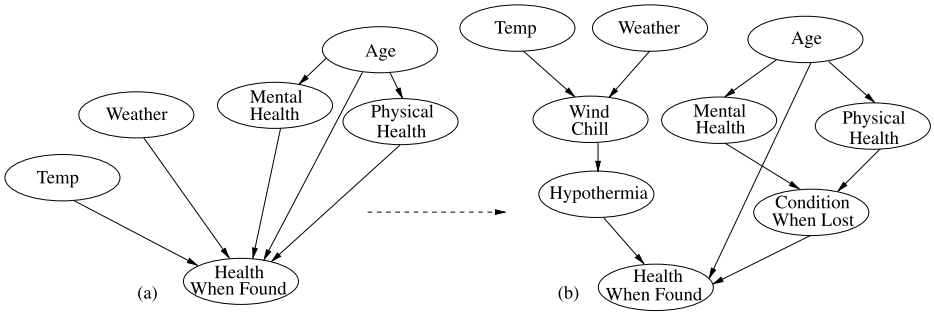


Figure 11. BNs from Case Study 2, "Search and rescue problem" (taken from Ref. 2, Figure 9.14). (a) The original submodel examined using Matilda. (b) The revised structure after evaluation, with two new variables and the "divorced" structure.

submodel included five observation nodes, one query node, and seven arcs. The expert used the three request types that are offered by Matilda, applying request type 1 to the query node and request type 2 to two sets of two observation nodes. This examination included the visualization of the four paths between the first pair of nodes and the one path for the second pair of nodes. The third type of request (request type 3) offered by Matilda was used to evaluate the relationship between an observation node and the query node. The domain expert then evaluated the recombined model, which now included six observation nodes, three query nodes, and 12 arcs. This evaluation followed the same pattern as that of the submodel. Request type 1 was applied to two of the query nodes, request type 2 was applied to two sets of two observation nodes and one set of two query nodes, again exploring all paths (5, 5, 9), and request type 3 was applied to two observation nodes.

It is important to note that both these models are a simplification of the desired model. Both the BN expert and the domain expert started from a very simple network and have been trying to find a satisfactory model, but have not succeeded. From the domain expert's perspective, this evaluation session was not about the need to understand a complicated network but rather about examining a simple prototype model, which he was not happy with, to find out if new understanding could be gained and what improvements could be made.

Through the examination of a substructure, the domain expert investigated the relationships between the observation nodes and found that information from two nodes in his model should be combined, leading to identification of two missing variables. Incorporation of these variables into the network, applying the principle of divorcing parents,[22] led to a more than 20-fold reduction in the number of parameters and improved the explanatory power of the model. The revised submodel is shown in Figure 11b. Interestingly, the expert's confidence in his BN model went from "somewhat confident" to "somewhat unconfident," as, due to the identification of the missing variables, he found his model to be oversimplified.

Case Study 3: Analysis of BN learned from data. Using Matilda, a domain expert investigated a network that had been constructed from data using an automated algorithm, CaMML.[23] This network comprised six observation nodes, one query node, and 11 arcs. Its investigation included:

(1) An examination of the relationships between the query node and the observation nodes: three sets that included the query node and an observation node (applying request type 2) and three sets that included the query node and a set of observation nodes (applying request type 3).
(2) An examination of the relationships between observation nodes: three sets of two observation nodes (applying request type 2).
(3) An examination of the relationships between the two nodes and the rest of the network (applying request type 1). For the first node, four paths were examined, whereas for the second node, only a single path was examined.

The domain expert was aware of possible statistical relationships between the variables of her domain but did not know exactly which ones they were. The investigation led to a better understanding of what statistical relationships are represented by the graph and, hence, a better understanding of the domain.

In addition, the expert wished to know in what way the graph represented the logical structure of the domain. Although the learned graph initially seemed completely nonintuitive to the expert, it gradually became more understandable throughout the investigation process, when the expert found that logical relationships that exist when information is introduced are represented by the graph.

This session led to the expert identifying a potential conflict between a BN for the domain based on the expert's diagnostic intuition and one that reflected causal influences; specifically, the arcs between the query and the observation nodes are in opposite directions. The expert concluded that she had a "mental model" (a cognitively perceived model) of the domain, based on her diagnostic intuition, that could have led to wrong independency assumptions. This conclusion was reached following a better understanding of BN technology, which, in turn, was a result of her work with Matilda.

In this case study, after working with Matilda and reaching these insights regarding the network structure and its representation of different situations in the domain, the expert felt confident enough to suggest a new structure for the domain that incorporated causal and statistical relationships.

4.3. Discussion

These case studies show the approach of displaying d-separation information via visual and verbal means can significantly assist the knowledge engineering process in different settings. It alerts an "outsider" user of a network (constructed by domain experts) to potential problematic situations that need consideration (e.g., first setting). It helps the domain expert who is also the user to investigate an existing network developed by the user at any stage of the development (e.g., second setting). It helps domain experts to investigate a network that represents their domain but which they have not been involved in building (e.g., third setting). It

may help communicate a complex network to the expert or to a new member of a development team (based on all three settings).

These case studies also showed that Matilda can be useful in a number of ways. First, it can help in the assessment of the graph as a representation of the domain, either when used by domain experts to investigate (1) the network they are building themselves or (2) a network, built by others, that represents their domain. The approach resulted in the identification of potential structural problems (such as wrong direction of arcs, missing variables, etc.) in all evaluation settings, by enabling a systematic review of important relationships across the whole network.

Second, the approach can assist in understanding and communicating the domain, the model, and BN technology to a domain expert. The domain experts gained a better insight into the structure of the network and its line of reasoning, the statistical relationships, and how they correspond to the logical relationships. It facilitated the understanding of causal direction of arcs and of how marginal independence and conditional independence assumptions are represented by the graph. We saw that a domain expert's intuitive "mental model" of the domain may be a source of confusion when constructing the causal model, potentially leading to modeling errors; investigation using Matilda may help identify and resolve such conflicts.

Third, it may help in the independent construction of the qualitative part, as part of the knowledge engineering process. In the second setting, the identification of missing variables prompted the domain expert to make changes to the model, specifically adding intermediate variables. As well as being a more logical representation of the domain, this significantly reduced the number of elicited probabilities, making the network more economical and practical.

During the preparation of this article, Hugin (March 2004) announced that they are now supporting d-separation examination, which also uses the structure for choosing nodes and for displaying d-separation. An example of the Hugin display is shown in Figure 12. The evidence nodes are marked with "e" (i.e., *Leaving* and *Smoke*), whereas their so-called source node, which corresponds to Matilda's query node, is shown with a thicker edge (*PhoneCall*). All the nodes that are d-separated from the source node, given the evidence, are indicated by a (red in Hugin) cross over the node. Hugin also allows the specification of a group of so-called target nodes of interest to the user. Hugin's examination tool uses the standard d-separation terminology and requires that the user is familiar with the d-separation criteria (A.L. Madsen, personal communication, January 2005). It does not provide the range of request types that Matilda provides nor does it give a verbal explanation. Finally, Hugin has yet to undertake any evaluation of the d-separation tool. However, the fact that a commercial application includes such an approach can be viewed as further confirmation of its usefulness.

5. CONCLUSIONS

In this article, we have presented and evaluated a novel visual and verbal explanatory approach for the examination of BN structure. We have shown that

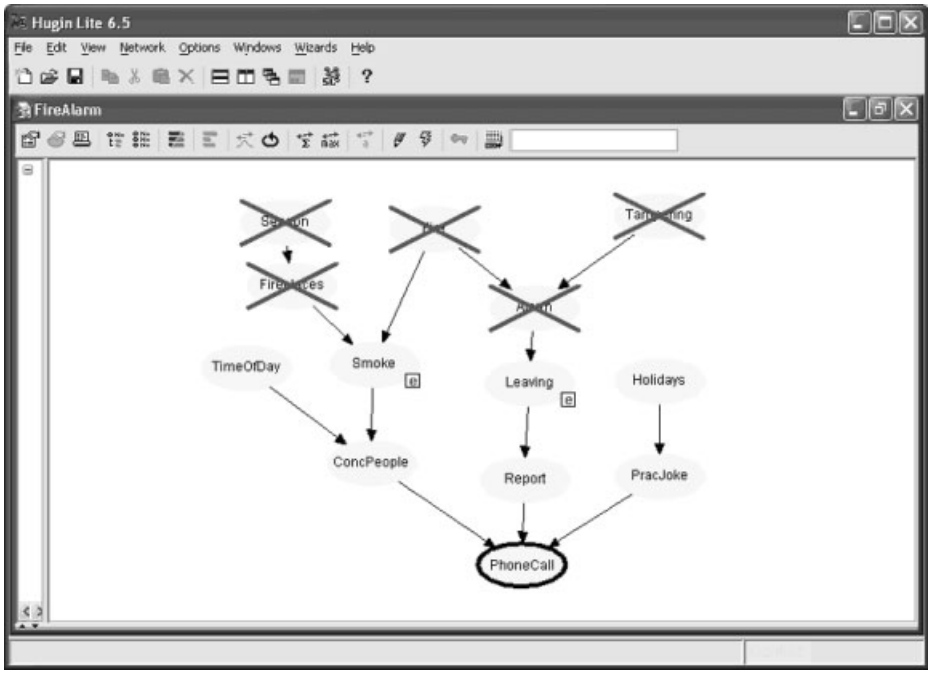


Figure 12. An example of the Hugin d-separation interface.

this approach is helpful in different aspects of the knowledge engineering process. It allows the construction and assessment of a network independent of the quantitative part, thus filling a current deficiency in the modeling tools and methods for knowledge engineering with BNs. Our approach enables the user to revisit the structural assumptions of the domain and gain a new understanding of the domain and/or BN technology, leading to the design and construction of an improved network.

We suggest that the investigation of independency assumptions that can be read from the graph structure by using the d-separation criterion be considered as an integral part of knowledge engineering, and that if it is used as part of the knowledge engineering process, it will support this process. We note, however, that in this set of investigations we have used the tool mainly to find independency assumptions in the BN that do not necessarily exist in the domain. Conversely, when the domain contains independency assumptions that are not represented by the model structure, Matilda is not as useful, because these independencies can still evolve from the parameters; in fact, in many cases this is unavoidable.[1]

Of course, using our approach does not guarantee detection of problems and does not provide modeling solutions. Rather, it can draw the attention of the user to a potential problem with independency relationships by highlighting parts of the model. The user has to decide whether to ignore the potential problem (either because it is not, in fact, a problem, or the user considers it to be only minor) or

to solve it by making changes that result in a more accurate capturing of the independency assumptions in the underlying joint probability distribution.

The approach presented in this article defines important questions regarding the relationships between variables in a BN, presents them as options to the user, and displays the answers to user queries. However, it does not suggest where to start the investigation, in what order to assess the network, or which relationships should be tested. We have already started to address this by describing a structured execution of the approach in Ref. 3 that will lead a user step by step through a comprehensive investigation of the model structure. One strand of future research will be to implement this structured execution methodology and evaluate it in different settings using both qualitative and quantitative methods.

Toward this end, we also need to implement various functionality changes suggested from the case studies to improve the approach and the tool, such as changing the path display to highlight v-structures. Future development of the tool will also include evaluation of the specific language, terminology, and visualization choices and perhaps even exploration of how the methodology can be scaled up, perhaps by using graphical methods such as collapsing subnetworks into a single node or expanding a single node into a subnetwork.

Acknowledgments

This research was undertaken while Tali Boneh was completing a Master of Science degree at the University of Melbourne. We thank our domain experts, Rik Head (Case Study 2—Search and Rescue) and Kaye Stacey (Case Study 3—Decimals tutoring), for their valuable contribution to this research.

References

1. Pearl J. Probabilistic reasoning in intelligent systems. San Mateo, CA: Morgan Kaufmann; 1988.
2. Korb KB, Nicholson AE. Bayesian artificial intelligence. Computer science and data analysis. Boca Raton, FL: Chapman & Hall/CRC; 2004.
3. Boneh T. Support for graphical modelling in Bayesian network knowledge engineering: A visual tool for domain experts. Master's thesis. Melbourne: University of Melbourne; 2003.
4. Poole D, Mackworth A, Goebel R. Computational intelligence: A logical approach. Oxford: Oxford University Press; 1999.
5. Druzdzel MJ, van der Gaag LC. Building probabilistic networks: Where do the numbers come from? IEEE Trans Knowl Data Eng 2001;12:481–486.
6. Laskey KB, Mahoney SM. Network engineering for agile belief network models. IEEE Trans Knowl Data Eng 2000;12:487–498.
7. Laskey KB. Sensitivity analysis for probability assessments in Bayesian networks. In: Heckerman D, Mamdani EH, editors. Proc Ninth Conf on Uncertainty in Artificial Intelligence, UAI93, Washington, D.C.; 1993. pp 136–142.
8. Wick MR, Thompson WB. Reconstructive expert system explanation. Artif Intell 1992; 54:33–70.
9. Helsper EM, van der Gaag LC. Building Bayesian networks through ontologies. In: van Harmelen F, editor. Proc 15th European Conf on AI; 2002. pp 680–684.

10. Laskey KB, Mahoney SM. Knowledge engineering for probabilistic models: A tutorial; 1999. Available at: http://ite.gmu.edu/~klaskey/papers/uai99tutorial.ppt (accessed December 2004).
11. van der Gaag LC, Helsper EM. Experiences with modelling issues in building probabilistic networks. In Gsmez-Pirez A, Benjamins VR, editors. Knowledge Engineering and Knowledge Management: Ontologies and the Proceedings of EKAW 2002. Berlin: Springer-Verlag; 2002. pp 21–26.
12. Wimberly F. Is X d-separated from Y by Z; 2002. Available at: http://www.andrew.cmu.edu/user/wimberly/dsep/dSep.html (accessed December 2004).
13. Acid S, de Campos LM. An algorithm for finding minimum d-separating sets in belief networks. In: Horvitz E, Jensen FV, editors. Proc of the Twelfth Annual Conf on Uncertainty in Artificial Intelligence (UAI'96), Portland, Oregon. San Francisco, CA: Morgan Kaufmann; 1996. pp 3–10.
14. HCIL. University of Maryland Human-Computer Interaction Lab; 2002. Available at: http://www.cs.umd.edu/hcil (accessed December 2004).
15. Geiger D, Verma T, Pearl J. d-separation: From theorems to algorithms. In: Proc 5th Workshop on Uncertainty in AI, Windsor, Ontario, Canada; 1989. pp 118–124.
16. Tian J, Paz A, Pearl J. Finding minimal d-separating sets. Technical Report (R-254), UCLA Cognitive Systems Laboratory; 1998.
17. CUErgo. CUErgo Cornell University Ergonomics Web, Ergonomic guidelines for user-interface design; 2002. Available at: http://ergo.human.cornell.edu/ahtutorials/interface.html (accessed December 2004).
18. Myers MD. Qualitative research in information systems. MIS Q 1997;21:241–242.
19. RUU. Bayesian network toy case studies; 2000–2001. Available at: http://www.csse.monash.edu.au/~annn/bncases.html (accessed December 2004).
20. Lagnado DA, Sloman S. Learning causal structure; 2002. Available at: http://titan.cog.brown.edu: 16080/~sloman/papers/LearningCaualStruct.pdf% (accessed December 2004).
21. Henrion M. Some practical issues in constructing belief networks. Uncertainty Artif Intell 1989;3:161–173.
22. Andreassen S, Jensen FV, Andersen SK, Falck B, Kjærul U, Woldbye M, Sørensen AR, Rosenfalck A, Jensen F. MUNIN—An expert EMG assistant. In: Desmedt JE, editor. Computer-aided electromyography and expert systems, chap. 21. Amsterdam: Elsevier Science Publishers; 1989. pp 255–277.
23. Wallace CS, Korb KB. Learning linear causal models by MML sampling. In: Gammerman A, editor. Causal models and intelligent data management. Berlin: Springer-Verlag; 1999. pp 89–111.