

Extensible Multi-Entity Models: A Framework for Incremental Model Construction

Kathryn Blackmond Laskey
Department of Systems Engineering
MS 5A6
George Mason University
Fairfax, VA 22030
klaskey@gmu.edu

Suzanne M. Mahoney
Information Extraction and Transport, Inc.
1730 N. Lynn Street, Suite 502
Arlington, VA 22209
suzanne@iet.com

Abstract

Graphical models have become common for representing probabilistic models in statistics and artificial intelligence. A Bayesian network is a graphical model which encodes a probability model as a directed graph in which nodes correspond to random variables, together with a set of conditional distributions of nodes given their parents. In most current applications of Bayesian networks, a fixed network is specified to apply to all problem instances. Inference consists of conditioning on certain random variables, called evidence variables, and inferring the distribution of others, called target variables. In more complex problems arising in artificial intelligence, it is useful to use the belief network formalism to represent uncertain relationships among variables in the domain, but it is not possible to use a single, fixed belief network to encompass all problem instances. This is because the number of entities to be reasoned about and their relationships to each other varies from problem instance to problem instance. This paper describes a framework for representing probabilistic knowledge as fragments of belief networks and a method for constructing situation-specific belief networks for particular problem instances.

1. Introduction

The widespread dissemination of the technology of graphical models has enabled a quantum leap in the complexity of problems that can be addressed by Bayesian methods. A graphical model is a parsimonious and modular parameterization for a joint probability distribution on a set of random variables. The probability distribution is expressed as a product of factors, where each factor involves only a small number of variables. The factorization permits the distribution to be specified using a manageable number of parameters. Computationally efficient general purpose methods exist for inferring or approximating conditional distributions of some variables in a graphical model given other variables. A large literature has grown on the topic of inferring structure and parameters of graphical models from exchangeable samples from the model in question. The spread of Bayesian methods for graphical models has been largely responsible for a new boldness among statisticians in attacking problems requiring large numbers of parameters. Graphical models with hierarchical prior distributions can adjust their effective dimensionality in response to patterns in the data. A large number of adjustable parameters provides the flexibility to model problems with complex structure. Protection against overfitting is provided by proper prior distributions that shrink toward lower dimensional subspaces without imposing strong assumptions on the direction of shrinkage or the dimensionality of the subspace. This new technology has made broadly available a heretofore unobtainable level of robustness on high dimensional problems. The artificial intelligence community has embraced graphical models as a cognitively feasible way to elicit complex probability models from domain experts. As a consequence, decision theoretic methods are rapidly increasing in popularity in the field of artificial intelligence. A new unified language for describing and specifying probability models has spurred a cross-disciplinary transfer of ideas, an explosion of interest in Bayesian methods in diverse fields, and a new sense of excitement and promise among Bayesian researchers and practitioners.

A graphical model is a probability distribution represented as a graph together with a set of local functions on subsets of nodes. The graph consists of a set of nodes, each of which represents a random variable, and a set of edges that represent independence assumptions satisfied by the joint probability distribution. In an undirected graphical model, the local functions are defined on cliques, or maximally connected subsets of nodes. In a directed graphical model, the local functions are conditional distributions for nodes given their parents. In either case, the local function maps the cross-product of the state spaces of the relevant nodes into the real numbers. The joint distribution on all the nodes is then defined to be proportional to the product of these local functions.

This paper is concerned with graphical models that may involve large numbers of random variables, perhaps numbering in the hundreds or thousands. Modularity of representation is essential for developing and maintaining such models. Different parts of a model may be constructed by different individuals, and over time the model may be maintained, modified, and extended by still other individuals. Modularity is also important computationally. A literature has grown on the incremental run-time construction of models from modular components (e.g., Breese, 1987; Wellman, et al, 1992; Goldman and Breese, 1992; Charniak and Goldman, 1993; Haddawy, 1994; Glesner and Koller, 1995; Mahoney and Laskey, 1998). We consider the problem of constructing a model for the purpose of inferring the values of some variables in the model given information on the values of other variables. We do not consider the problem of inferring parameters from data. We confine attention to directed graphs for discrete variables with finite state spaces.

2. Template Models

A standard graphical model represents a joint probability distribution for a set of random variables. The set of possible outcomes is implicitly defined as the cross-product of the state spaces of the random variables represented by the nodes in the graph. A graphical model can be used to represent uncertainty about a single individual, or can apply to a population of exchangeable individuals, where all individuals are characterized by the same attributes, the same state spaces for the attributes, the same independence constraints, and the same local functions for their probability models. For example, in a medical diagnosis problem, the population might be a set of patients visiting a clinic and the attributes might be variables relating to medical history, symptoms, test results, and conditions from which the patient might be suffering. In an image restoration problem, the population might be digitized images and the variables might be pixel colors and intensities, edges, pixel texture classifications, etc. We call this kind of graphical model a *template model*, because the same template (variables, states, arcs, local functions) applies to all individuals in the population. With appropriately defined hierarchical prior distributions, inferring both structure and parameters of a template model from a sample of individuals is a well-understood problem, although quite challenging and interesting for some problems (e.g., those with missing data or unobserved variables). Inferring the values of some variables for a given individual conditional on the values of other variables is also a well-understood problem for which general purpose exact and approximate algorithms exist and continue to be refined.

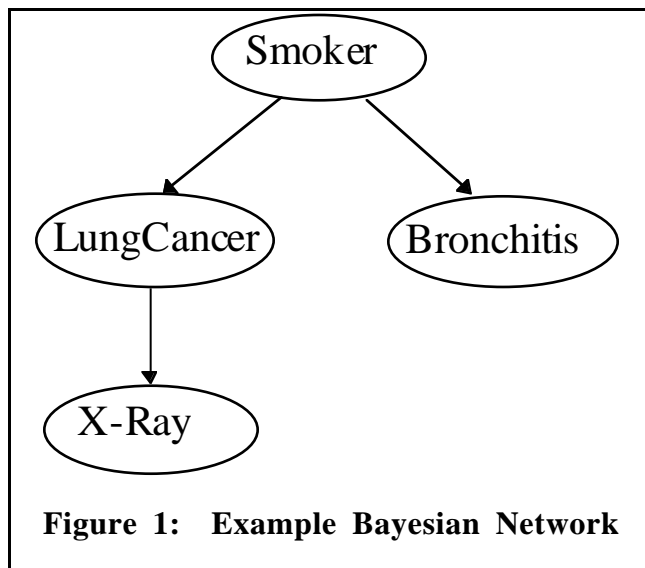
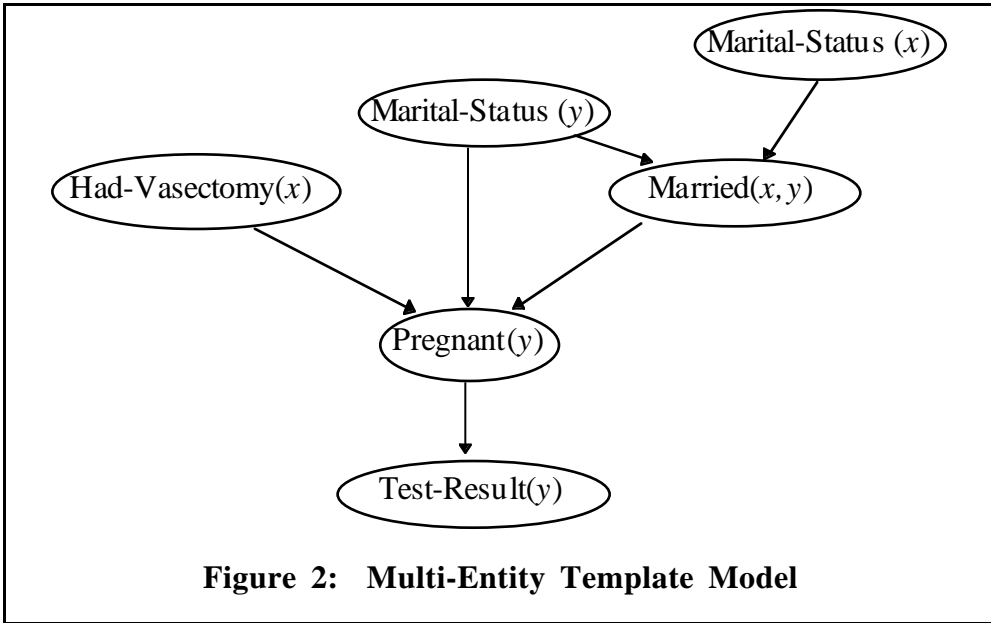


Figure 1: Example Bayesian Network

While template models are quite useful for the problems to which they apply, and have fostered a surge of interdisciplinary interest in decision theoretic methods, many of the most challenging and interesting problems in artificial intelligence cannot be treated with template models. The focus on template models has caused many in the artificial intelligence community to downplay graphical models on the argument that they have only propositional expressive power (cf., Wellman, et al, 1992). This is not strictly true. A template model is implicitly quantified over the population of individuals being modeled. For example, consider the Bayesian network of Figure 1, assumed to apply to a population of patients in a medical clinic. We can let the

symbol x denote an individual patient about which we know nothing except that he/she belongs to the population being modeled by this network. We can represent our knowledge about this patient's attributes as the random variables $Smoker(x)$, $LungCancer(x)$, $Bronchitis(x)$, and $X-Ray(x)$. These random variables have the joint distribution defined by the Bayesian network of Figure 1. If we then learn that patient x has a positive chest X-ray, we can apply belief propagation (Jensen, 1996) to infer posterior marginal distributions for the other nodes in the network. These marginal distributions will be different from those of Patient y who we know to have a negative chest X-Ray, or Patient z whose X-Ray has not yet been taken. In other words, the network of Figure 1 represents a family of probability models, one for each patient in the clinic. These models happen to be identical except for which random variables are known and what are the values of the known random variables. Nevertheless, each patient has his/her unique probability model.

A bit more interesting is the network of Figure 2, defined on a population of pairs of men and women, referred to by the symbols x and y , respectively. This network is characteristic of a more general class of template models where the population consists of n -tuples of entities. In this context, the word *entity* is quite general, and can refer to an object, a person, an action, a plan, a belief, or in general anything about which we may want to reason. Each entity has a set of *roles* of defined type, which are referred to by symbols called *designators*.¹ For example, in Figure 2 the designators are x and y , which refer to the roles “Potential-Father” and “Patient,” which are of type “Man” and “Woman,” respectively. The random variables in the network refer to attributes of individual entities (e.g., Had-Vasectomy(x) refers to an attribute of potential father x) or to relationships between entities (e.g., Spouse(x,y) refers to a relationship between patient y and potential father x). Typing of entities permits different attributes and different relationships to apply to entities of different type. We call this a *multi-entity template model*. It is used to reason about a set of entities, their attributes, and their relationships to each other. It is a template model because the number of entities, the type of each entity, and the attributes and relationships to be reasoned about are fixed at model specification time. Therefore, a single fixed (if very large) Bayesian network can be used for all problems of inferring the distributions of some random variables given the values of others.



A multi-entity template model can be described as an acyclic directed graphical model (Bayesian network), indexed by an n -tuple of designators for its roles, that applies to each element of a population consisting of n -tuples of entities of the appropriate types. Each node in the Bayesian network is a random variable indexed by one or

Figure 2: Multi-Entity Template Model

more of the designators that indexes the network. Each random variable refers to an attribute of an entity or a relationship between a subset of entities of appropriate types. The state space for each random variable consists of the possible values of the attribute or relationship in question.

It is useful to be able to specify and store a model in terms of modular components which are assembled at run-time to create a Bayesian network to reason about a particular problem instance. The following definitions provide a framework for specifying a multi-entity template model in

¹ Designators are referred to as variables in the literature on predicate calculus logic. We use designator to avoid confusion with the term random variable as commonly used in statistics. The designator x refers to an arbitrary, unspecified, but particular entity. We may be uncertain about an attribute $A(x)$ even when we know which entity x is. The random variable $A(x)$ models this uncertainty. Uncertainty about which entity is being referred to by the designator x is modeled by the random variable X . The random variable $A(X)$ refers to the uncertain value of attribute A for the uncertain entity X .

terms of components called network fragments. Later we extend the framework to permit inference about the assignment of entities to the designators indexing the model.

Definition 1: Let T be a set consisting of a finite number of primitive entity types. For each $t \in T$, let \mathcal{V}_t be a set of possible values for entities of that type. An *entity* is a pair $e=(t,v)$, where $t \in T$ and $v \in \mathcal{V}_t$.

Types are used to model classes of exchangeable random variables. The basic assumption is that all entities of the same type are exchangeable. It would be useful to extend Definition 1 to encompass type hierarchies. For example, much of what we know about people applies, with minor changes, to women and to Asian women. This is called *inheritance*: as when a woman inherits the property of having eyes from being a vertebrate. Inheritance is important for elicitation, maintainability and extensibility of a knowledge base. A formal statistical model for inheritance is important for the problem of learning multi-entity models. It is also useful to be able to organize entities into compound entities. Here we focus on a simple framework that can handle problems of reasoning about an indefinite number of entities related to each other in unknown ways. We will extend our framework to type hierarchies and compound entities in future work.

Definition 2: A *random variable* is an proposition or question about a tuple of entities that can take on values in a mutually exclusive and collectively exhaustive set of outcomes in a finite *outcome set*. Each random variable is uniquely identified by a name and a tuple of entities that fill a set of *roles*.

We assume that if two random variables differ only in which entities are assigned to their respective roles, but if these entities match in type, then the random variables are exchangeable. Thus, if we quantify over entities of appropriate type, we can define classes of exchangeable random variables.

Definition 3: A *random variable class* is identified by a name, a set of roles, and a type for each role. The class consists of all random variables obtained by assigning entities of appropriate types to the roles.

The exchangeability assumption on random variables in a class is what permits the application of the same model to different individuals. Note that we may use the same name to refer to random variables defined on individuals of different types. In the example of Figure 2, Marital-Status(*Fred*) is not exchangeable with Marital-Status(*Julie*) because *Fred* and *Julie* are entities of different types (“Man” and “Woman,” respectively). However, Marital-Status(*Fred*) is exchangeable with Marital-Status(*Roger*) because they are of the same type.²

Sometimes it is desirable to restrict the entities filling the roles not to range over all entities of legal types. It may be appropriate to impose constraints on how roles can be filled. For example, consider the random variable Sister-Batteries(x,y), which denotes whether x and y are missile batteries belonging to the same regiment. We might not want to represent this random variable in the model if x and y are the same unit. If sister units are known to be always located within a certain distance range of each other, we may wish to exclude pairs of batteries known not to satisfy the distance restriction. Constraints like these can be expressed in our framework by defining a random variable to express the constraint and conditioning on its truth. This works in theory, but it is computationally far more efficient to permit constraints to be applied at the time entities are being assigned to designators. Note also that our simple language has no way to express identity constraints such as that Sister-Batteries(x,y) and Sister-Batteries(y,x) are the same random

² Type hierarchies would permit treating them as exchangeable when reasoning at a higher level of granularity.

variable. Again, this can be expressed by explicitly conditioning on the constraint, but it would be more efficient to extend the representation language and the network construction method to make use of this knowledge. We defer these extensions for future research.

Definition 4: A *network fragment class* represents knowledge about the conditional distribution of a set of random variable classes. A network fragment consists of a name, a finite set of roles, a type for each role, a set of input random variable classes, a set of resident random variable classes, a set of role correspondence functions, a fragment graph, and a set of local distributions. There is a role correspondence function for each random variable. It specifies a fragment role for each random variable role. The mapping of random variable roles to fragment roles is assumed to be one-to-one and type consistent. The fragment graph is an acyclic directed graph with a node for each input and each resident random variable class. Nodes corresponding to input variables must be roots in the graph. With each resident variable is associated a set of probability distributions on its state space. For resident random variables that are roots in the graph, there is one such distribution, called the marginal distribution. For non-roots, there is a distribution for each combination of states for the random variables corresponding to parents of the node.

Every type-consistent assignment of entities to roles in the network fragment specifies an *instance* of the fragment class. The fragment instance assigns random variables to the nodes of the fragment graph by filling in its designators with entities of the appropriate types. All instances of a fragment encode identical conditional distributions on their resident random variables given their input random variables.

Definition 5: A multi-entity template model consists of a finite set of random variable classes, a finite set of network fragments defined on these random variable classes, and a finite set of designators with associated types, that satisfies the following conditions:

1. Each random variable class, identified by its name, roles, and role types, is resident in no more than one network fragment.
2. Each random variable class appearing in some fragment is resident in at least one fragment.
3. The graph union of all the network fragments contains no directed cycles. In performing the graph union, the identity relation is the random variable class. Thus, nodes are considered identical for the purpose of graph union if the associated random variable classes have the same name, roles, and role types.

These three conditions ensure that the graph union of the fragments together with the distributions encoded by the fragments is a Bayesian network. Any type-consistent assignment of designators to random variable roles determines a multi-entity template model. That such a mapping exists is easily established: simply use the same designator for all entities of a given type. However, note that if the model includes constraints such as the distance constraint for sibling units, we may discover inconsistencies when we try to enter the constraints as evidence. The acyclicity condition is too restrictive for temporal reasoning problems. Time cannot be treated like the other designators, because of the need to permit directed paths from a random variable at one time to the same random variable at a future time. We defer treatment of temporal models for future work.

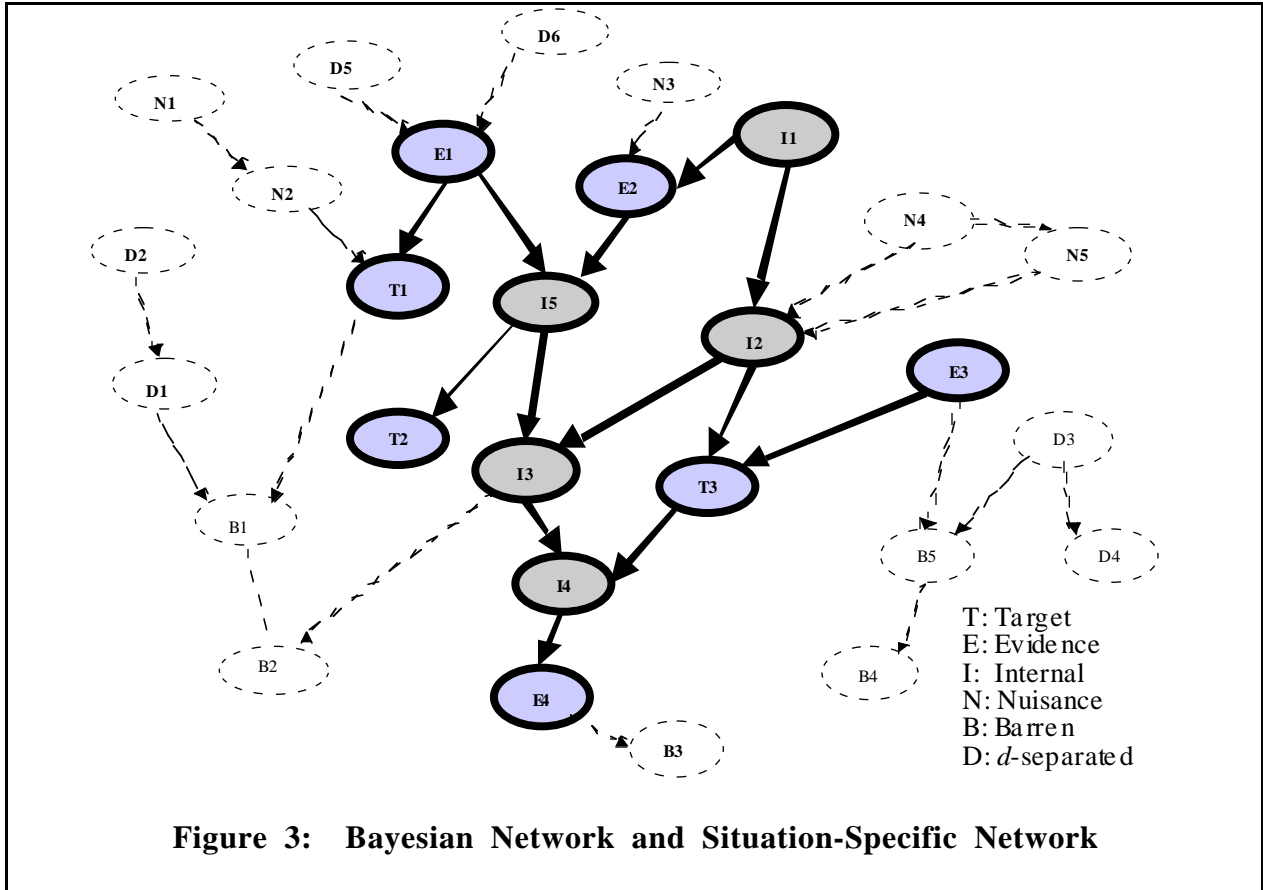
We assume there is a fixed and finite number of possible random variables to be reasoned about. A useful extension to our framework would be to provide a set of operators for constructing new random variables from existing random variables. This capability is called *feature discovery* in the machine learning literature. Another useful extension would be to allow the states of random variables to be entities.

3. Model Construction for Multi-Entity Template Models

The canonical problem we consider in this paper is to infer the values of some random variables given values of other variables. In a multi-entity template model, let (x,y,z) denote a set of entities, let $E(x,y)$ denote a subset of variables for which values are known (*evidence* variables), and let $T(x,z)$ denote a subset of variables whose values we wish to infer (*target* variables). Our objective is to compute the response to a query of the form $Q:[P(T(x,y)|E(x,z))=?]$. We can approach this problem in one of three ways: (1) start with the entire Bayesian network and apply an exact or approximate belief propagation algorithm to compute the response to the query; (2) start with the entire Bayesian network and prune off the parts not needed to respond to the query (Baker and Boulton, 1990); or (3) specify the model in terms of modular fragments and construct only those parts of the model needed to respond to the query (e.g., Haddawy, 1994).

Mahoney and Laskey (1998) define a *situation-specific network* as a Bayesian network that is a minimal subgraph of the original network sufficient to compute the response to a query. Three categories of nodes are removed from the full model to form the situation-specific network for a query. First are the nodes that are d -separated (Pearl, 1988; Jensen, 1996) by the evidence nodes from the target nodes. The d -separation condition implies that the target nodes are independent of the removed nodes given the evidence nodes, and they are therefore not needed to respond to the query. Next to be removed are the barren nodes. A node is barren if it is neither an evidence node nor a query node, and there is no directed path from it to an evidence or a query node. These nodes are not necessarily independent of the target nodes given the evidence nodes. However, in a directed graph representation, the marginal distribution for a set of nodes is specified entirely in terms of the distributions for those nodes and their predecessors. That is, the joint distribution of target and evidence variables, and hence the query response, can be computed entirely from the subgraph from which the barren nodes have been removed. The final nodes to be removed are the nuisance nodes (Lin and Druzdzel, 1997). Nuisance nodes are non-barren and non- d -separated nodes that do not lie on an evidential trail³ between an evidence and a target node. Unlike d -separated and barren nodes, nuisance nodes are computationally relevant to the query. However, the way in which their distribution enters into the query is independent of the values of the evidence nodes. Therefore, they can be marginalized out prior to query processing without affecting the result of the query. Nuisance nodes can easily be removed if marginal distributions for all nodes are pre-computed and stored with the nodes. This is far more efficient than recomputing marginal distributions for nuisance nodes each time a query is processed (Lin and Druzdzel, 1997). Figure 3 shows a Bayesian network and the corresponding situation-specific network, where the nodes not in the situation-specific network are shown as dotted ovals.

³ An evidential trail between two sets of nodes is a minimal active undirected path from a node in one set to a node in the other. A path is active if it includes either a head-to-tail or tail-to-tail connection to a non-evidence node, or a head-to-head connection to a node that is either an evidence node or a predecessor of an evidence node.



Most work on incremental model construction works with a variant of what Mahoney and Laskey (1998) call simple bottom-up construction. Simple bottom-up construction builds up from the evidence and target variables of a query. In a recursive manner, it adds nodes to the constructed Bayesian network moving from child to parent until all ancestors of the evidence and target variables have been constructed. The knowledge base of network fragments provides the distributions and links to the parents for all the variables. If marginal distributions at nodes are stored, then the nuisance nodes can be rapidly marginalized out of the network obtained from simple bottom-up construction to create a situation-specific network. It is interesting to note that one need never build downward to construct a situation-specific network. If there are no evidence nodes below a node, all its descendants are barren and need not be considered. If there are evidence nodes below it, the relevant descendants will be reached using simple bottom-up construction from the evidence nodes below it.⁴

⁴ While this is true in theory, it may be necessary in practice to do some downward construction. Downward construction may be used to reach nodes that simple bottom-up construction ignored as not worth instantiating, or to reintroduce nodes that were introduced but then pruned for computational reasons.

4. Extensible Multi-Entity Models

In complex problems such as those arising in artificial intelligence, it is useful to use the language of graphical models to represent uncertain relationships among variables in the domain, but it is not possible to specify an explicit, fixed belief network that applies to all problem instances. In such problems, the number of entities to be reasoned about and their relationships to each other varies from problem instance to problem instance. Examples of this kind of domain include natural language understanding (Charniak and Goldman, 1993) and military situation assessment (Laskey and Mahoney, 1997; Mahoney and Laskey, 1998; Laskey, 1998). Although they consider only notional applications, the representation framework of Koller and Pfeffer (1997) is intended to address this kind of problem.

As an example, consider the problem of understanding the following simple fragment of text:

Julio went into the store. He emerged carrying a bottle of wine. He gave the bag to Franco, got into the car, and drove away. Franco put it on the seat between them. He said that the party was going to be fun.

Just from this short bit of text, we can draw an enormous number of inferences that are not stated explicitly. A few of these are:

“He” in the second sentence refers to Julio.

The bottle of wine is in the bag.

Franco intends to go to a party, probably accompanied by Julio.

The destination of the car is likely to be the party.

Julio is driving the car.

“He” in the last sentence probably refers to Franco, although it might refer to Julio.

To draw these kinds of inferences, the reasoner needs to postulate very many entities and reason about their relationships to each other. Among the relevant entities for this simple paragraph are:

The person named Julio;

The store;

The referent of “He” in the second sentence;

The bottle of wine;

The place from which the referent of “He” emerged carrying a bottle of wine;

Julio’s reason for buying the wine;

and so on. We may postulate an entity and later infer that it is the same as a previously postulated one (e.g., the store and the place from which the referent of “He” emerged are the same in our wine purchase example).

It is clear just from this simple example that there is no hope of constructing an explicit multi-entity template model for a problem as complex as natural language understanding. To do so, we would have to enumerate all the entities that might be encountered in a complex piece of text, together with all the properties and relationships between these entities that we might wish to reason about. A more natural way to attack this problem would be to hypothesize entities as the need arises in processing the text and discard them when they are no longer needed (i.e., we no longer need to maintain two separate entities for the person named Julio and the referent of the pronoun “He,” once the probability that they are the same individual becomes sufficiently high). We also dynamically postulate random variables as the need arises to refer to attributes and relationships we want to reason about. For example, a party is a highly salient reason for purchasing wine and is not a surprising explanation for why Julio went into the store. Nevertheless, it is quite natural not to consider the question of whether Julio was going to a party until we reach the final sentence.

In complex reasoning tasks such as this, the number of possible entities we might need to reason about and the number of attributes and relationships we might want to consider is effectively unbounded. An extensible multi-entity model is an extension of the multi-entity template model that permits reasoning about which entities are assigned to which roles. Like a multi-entity template model, random variables are quantified by designators that range over a type-specific set of entities. What is new in an extensible multi-entity model is the ability to make multiple “copies” of a random variable to refer to different entities that are hypothesized to fill a given role. Thus, a multi-entity template model is conditioned on an assignment of entities to roles, while an extensible multi-entity model permits the expression of multiple hypotheses about which entities fill which roles.

To develop the theory of extensible multi-entity models, we drop the assumption of a pre-specified mapping from designators to fragment roles and introduce additional machinery to support inference about this mapping. We introduce a special value *, taken to mean “non-existent,” as a possible value for all entities and all random variables. A special “existential quantification” structure is introduced for generating hypothesized entities. This structure plays the same role as the existential links in Charniak and Goldman (1993). Finally, we describe how to modify the network construction algorithm to incorporate random variable replication and uncertainty about the assignment of entities to roles.

Our main concern is the treatment of fragments in which designators appear in a parent node but not in the child. If all designators in the child appear in the parent, then simple bottom-up construction can always unambiguously fill in all designators of the parent of a node once the child designators have been filled in. The only nodes that may be left with unspecified designators when simple bottom-up construction terminates are the barren nodes. There is therefore no uncertainty about the assignment of entities to roles. (See Figure 4 for an illustration.) It is for this reason that Haddawy (1994) assumes that any designators that appear in a parent also appear in the child. This assumption is too restrictive for our purposes and is violated frequently in models we have constructed for practical problems (Laskey, 1998). Considering Figure 2, it is clearly an undue restriction not to be able to consider whether a woman is pregnant without explicitly hypothesizing an entity who is the father of the child and including that entity in the specification of every random variable related to her pregnancy. Imagine not being able to talk about nausea without specifying whether it was nausea due to pregnancy, and if so, who the father of the child was!

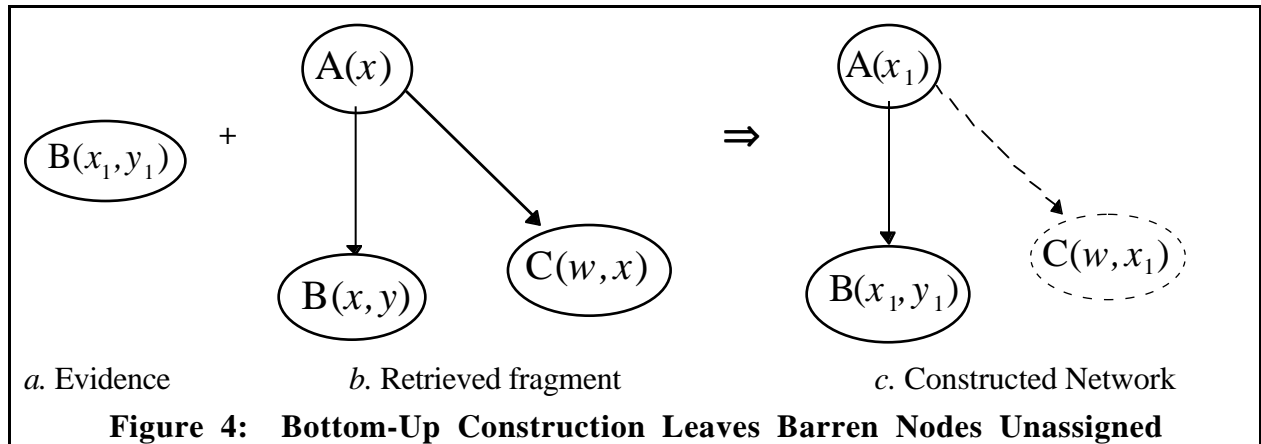
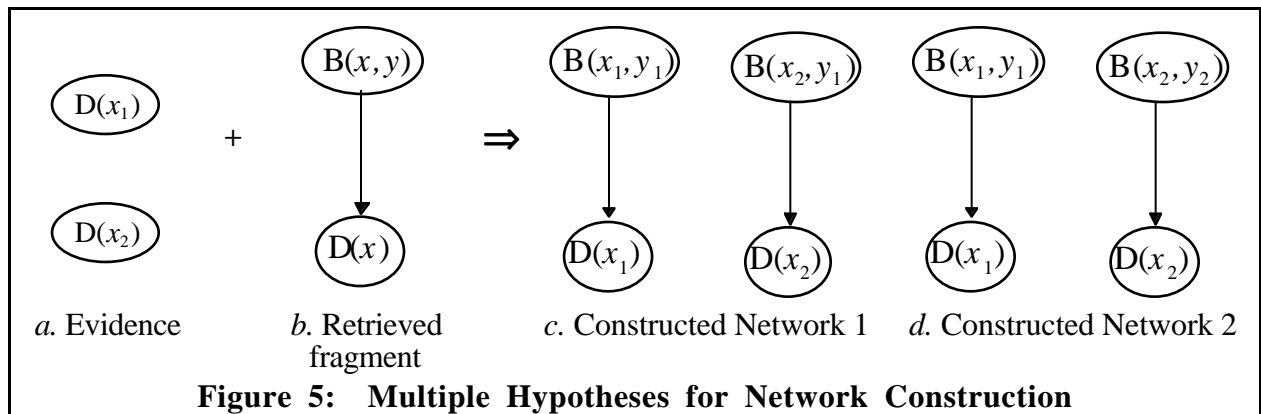


Figure 5 illustrates what happens when we relax this restriction. Simple bottom-up construction leaves node designator y unspecified both times the fragment is instantiated. Both Figure 5c and Figure 5d are consistent with the evidence. It is possible, of course, to explicitly enumerate all possible hypotheses about how many entities there are and what their relationships are, and create a separate model for each, as in Figures 5c and 5d. However, in most problems these models share

a great deal of structure. It is far more tractable to construct a single network where shared structures are represented only once.

We can represent our uncertainty about whether the model 5c or 5d should be applied by introducing uncertainty about whether y_2 exists as an entity separate from y_1 . To represent existence uncertainty we introduce the special value $*$. For example, in a military situation assessment problem, suppose we have an imagery report of a vehicle that could be either a missile unit or a truck. If it is a missile unit, we postulate a unit, called its defended element, that it is defending. In case the unit is a truck, the value of the defended element is $*$. We can also use $*$ to reason about whether two entities are the same or different. Suppose we postulate entity e_1 to refer to the person named Julio, and entity e_2 to refer to the referent of the pronoun “He.” In case the referent of the pronoun is Julio, the value of e_2 is $*$, and all evidence we acquire about “He” is applied to e_1 . If the probability of the value $*$ becomes sufficiently high, we can prune the e_2 hypothesis by deleting all random variables referring to e_2 . Finally, we can also use the value $*$ to model constraints and tautologies. We create a random variable expressing the constraint with two possible values, *true* and $*$. We then declare the value *true* as evidence.



We need to specify a probability model for entity to role assignment. We assume that if a proposition is introduced in network construction that contains an unspecified designator, there is a small type-specific probability it refers to a previously unmentioned entity, and it is equally likely to refer to all previously mentioned entities of the appropriate type. The following modifications to the network construction algorithm result in a constructed Bayesian network for this model.

1. To compute the distribution of a node instance $A(x_1)$ of node class $A(x)$ for which all designators named in parent nodes are among the x , use the distribution from the fragment in which $A(x)$ is resident, but add the state $*$ to parents and child, and declare $A(x_1)$ to have value $*$ with probability 1 if any of its parent nodes has value $*$.
2. To compute the distribution of the node instance $A(x_1)$ of node class $A(x)$, for which at least one parent contains designators not mentioned in x , do the following.
 - a) For each such unspecified designator w , create a new node $W(i)$, called an *existential random variable*, to represent uncertainty about the entity assigned to w . Use a previously unused symbol for the designator i (such as the next unused integer). The states of $W(i)$ are all previously created entities of appropriate type, together with a new entity w_i , to represent the hypothesis that a previously unmentioned entity fills the w designator. Assign a fixed, type-specific probability to w_i and divide the remaining probability equally among the other states of $W(i)$. Only one existential random variable is created for each designator in a fragment. If one has already been created for this designator in this fragment, this step is skipped.

- b) Create new instances of the parent classes of $A(x)$ as necessary to make sure that all type-appropriate hypotheses for the unfilled designators using both old and newly created entities have been represented.
 - c) Set the distribution for $A(x_1)$ as follows. Given a specific combination of states of the existential variables that are parents of $A(x_1)$, its distribution is taken from the instances of its parents that match these entities in the appropriate roles as determined by the role correspondence function. Note that $A(x_1)$ is independent of the values of all other copies of a parent variable class other than the one matching the existential variable values.
 - d) An arc is drawn from $W(i)$ to any random variable referring to the new entity w_i that does not have a parent referring to w_i . The distribution of the variable is as specified in its home fragment if $W(i)$ is equal to w_i ; otherwise it is equal to * with probability 1. $W(i)$ is not a parent to other copies of these variable classes that do not mention w_i .
3. We make one minor modification of the above procedure. If there are no existing entities that can fill the role, then no existential node need be created. This is because in this case $W(i)$ would have only one possible value, and there is therefore no need to represent uncertainty about it.

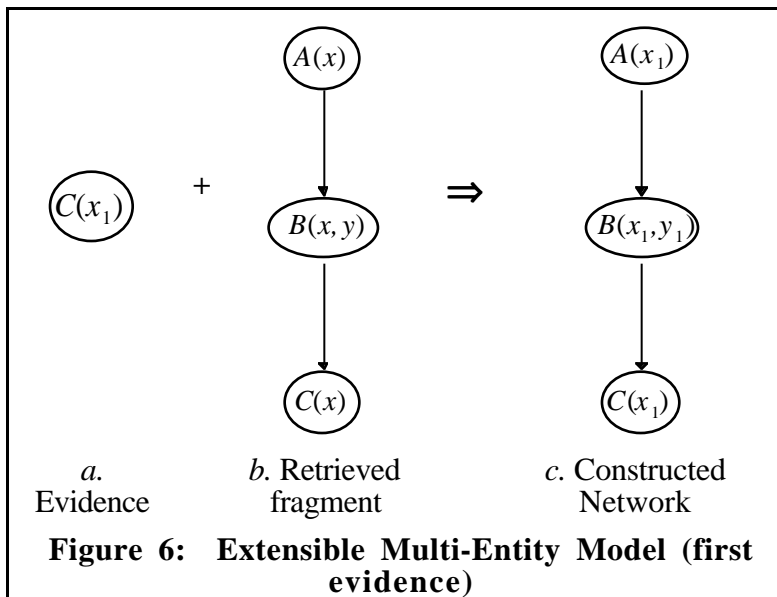


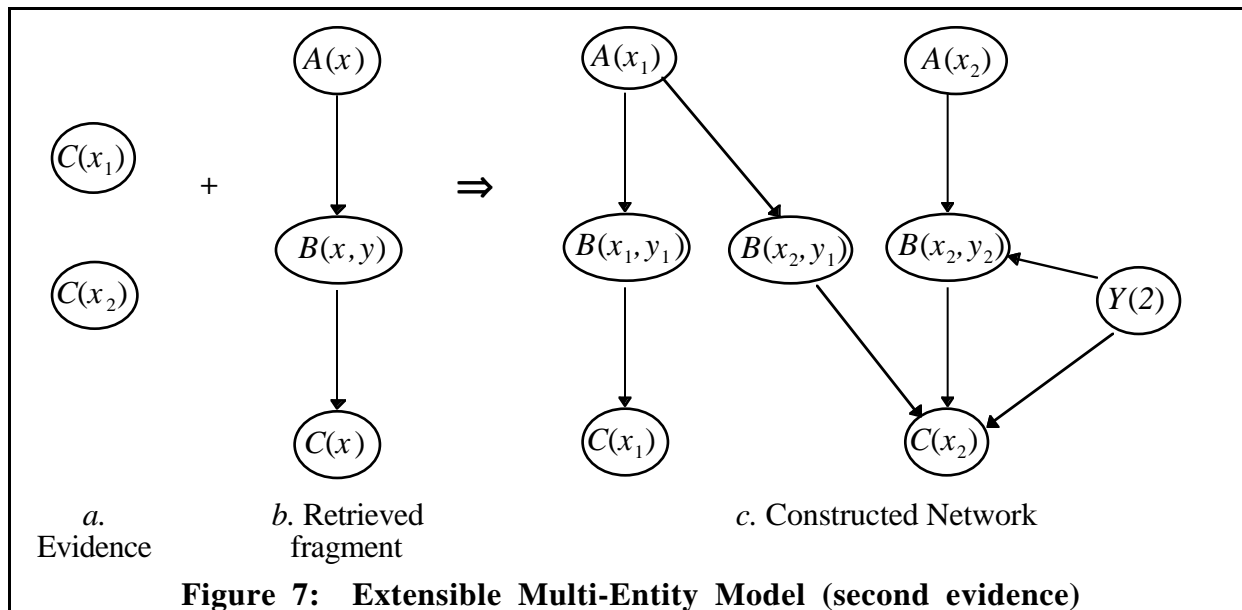
Figure 6: Extensible Multi-Entity Model (first evidence)

These steps are illustrated in Figures 6 and 7. The evidence $C(x_1)$ causes the fragment in Figure 6 to be retrieved. Designator y remains unspecified. Because there are no other variables of this type, we create a new entity y_1 as the explanation and obtain the model of Figure 6c. Next, evidence $C(x_2)$ is introduced. There are now two possible referents for y : the entity y_1 already hypothesized, and an entity y_2 created to explain the new evidence. Both possibilities are considered. Variable $C(x_2)$ takes its distribution conditional on $B(x_2, y_1)$ if $Y(2)=y_1$ and

conditional on $B(x_2, y_2)$ if $Y(2)=y_2$. $Y(2)$ is a parent of $B(x_2, y_2)$ because the latter only exists under the hypothesis that y_2 exists as a separate entity distinct from y_1 .

5. Discussion

A standard template model encodes a single level of exchangeability: among individuals in the population to which the model applies. An extensible multi-entity template model provides an additional level of exchangeability: between entities of the same type. This paper developed a simple language for expressing these exchangeability assumptions, described a framework for constructing probability models that reason about the assignment of individual entities to roles in the model, and sketched a very simple process for constructing such models.



There is a great deal of room for extending this framework. The construction algorithm we sketched is simple but highly inefficient. Methods are needed for limiting the proliferation of hypotheses. These include pruning unlikely hypotheses, combining similar hypotheses, re-introducing pruned hypotheses, and re-splitting combined hypotheses. A formal treatment of inheritance would add another level of exchangeability: between subtypes of the same parent type. A treatment of temporal reasoning would add still another level: between time-translated random variables. Finally, the specification of prior distributions on the parameters of the network fragments would provide the ability to learn multi-entity template models from data.

Acknowledgments

The research reported in this paper was sponsored by DARPA and the U.S. Army Topographic Engineering Center under contract DACA76-93-0025 to Information Extraction and Transport, Inc., with subcontract to George Mason University. Many thanks to Bruce d'Ambrosio for helpful and stimulating discussions.

References

- Baker, M. and T. Boulton, (1990) Pruning Bayesian Networks for Efficient Computation. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, General Electric, Cambridge, MA, pp. 257-264.
- Breese, John S. (1987) *Knowledge Representation and Inference in Intelligent Decision Systems*. Ph.D. dissertation, Department of Engineering-Economic Systems, Stanford University.
- Charniak, E. and Goldman, R (1993) A Bayesian Model of Plan Recognition, *Artificial Intelligence* 64, p. 53-79.
- Glesner, S. and D. Koller (1995) Constructing Flexible Dynamic Belief Networks from First-Order Probabilistic Knowledge Bases, in ECSQARU '95, pp. 217-226.
- Goldman R. P. and J. S. Breese (1992) Integrating Model Construction and Evaluation. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference* Morgan Kaufmann Publishers, San Mateo, CA. pp. 104-111.
- Haddawy, P. (1994) Generating Bayesian networks from Probability Logic Knowledge Bases. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA pp. 262-269.
- Jensen, F. (1996) *An Introduction to Bayesian Networks*, New York: Springer.
- Koller, D. and A. Pfeffer (1997) Object-Oriented Bayesian Networks In Geiger, D. and Shenoy, P. (eds) *Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference*, San Francisco, CA: Morgan Kaufmann.
- Laskey, K.B. and S. M. Mahoney (1997) Network Fragments: Representing Knowledge for Constructing Probabilistic Models. In Geiger, D. and Shenoy, P. (eds) *Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference*, San Francisco, CA: Morgan Kaufmann.
- Laskey, K.B. (1998) *Recognition Fusion: Network Fragments for Inferring Entity Existence and Type*, Rosslyn, VA: Information Extraction and Transport, Inc.
- Mahoney, S.M. and Laskey, K.B. (1998) Constructing Situation-Specific Networks. In Cooper, G. and Moral, S. (eds) *Uncertainty in Artificial Intelligence: Proceedings of the Fourteenth Conference*, San Francisco, CA: Morgan Kaufmann.
- Lin, Y. and M. J. Druzdzel (1997) Computational Advantages of Relevance Reasoning in Bayesian Belief Networks. In Geiger, D. and Shenoy, P. (eds) *Uncertainty in Artificial Intelligence: Proceedings of the Thirteenth Conference*, San Francisco, CA: Morgan Kaufmann. pp. 342-350.
- Wellman, M.P., J.S. Breese, and R.P. Goldman (1992) From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(1):35-53.

