BROWN UNIVERSITY
Department of Computer Science
Master's Project
CS-89-M2

"Relationship Between Temporal Bayes Networks and Markov Random Process Transition Tables"

by

Linda I. Mensinger Nunez

Relationship Between Temporal Bayes Networks and Markov Random Process Transition Tables

by
Linda I. Mensinger Nunez
B.A., Douglass College, Rutgers University, 1982

Submitted in partial fulfillment of the requirements for the
Master of Science Degree in Computer Science
at Brown University

April 1989

Thomas L. Dean, advisor

Acknowledgements

This project satisfies the final requirement for a Master's Degree from Brown University. Although it is impossible to mention and thank everyone who helped me reach this point, there are a few whose contributions I'd like specifically to acknowledge.

First, I would like to thank Tom Dean for agreeing to serve as my advisor for the past two years - I couldn't have asked for a better one. I am grateful for the discussions I have had with him and Keiji Kanazawa, which formed the basis for my work. Along these lines, I would like to acknowledge the National Science Foundation grant IRI-8612644 and DARPA/AFOSR contract F49620-88-C-0132DEF which helped to support some of this work.

I am also grateful for the support I received from my friends and colleagues at Nestor, Inc., which was generally more emotional than financial in nature. In particular, I'd like to mention Sushmito Ghosh, because throughout it all, he never lost faith in me and encouraged me to continue even when I was unsure. I would also like to thank Alex Contovounesios, Sandra Mehlmann and Paul Stefancyk for their understanding and support, particularly as the end of my work drew near. A special thank you to Debbie Bossian, who offered many valuable suggestions regarding the style and appearance of this paper.

I am particularly appreciative of the support of my husband, Ray Nunez. He never begrudged me the time or money spent on my education, and never complained about all the times he ate dinner late or alone because I was working. On the contrary, he knew how important it was to me, shared my joys as the various milestones were reached, and encouraged me in so many ways, including spending some evenings with me while I worked, his mere presence giving me the confidence to tackle the next obstacle. Thank you, sweetheart, for your love and patience (I know those were some rather boring nights and uncomfortable chairs you waited in!)

Finally, I would like to thank God for the strength it took to finish a Master's Degree while working at a demanding full-time job and managing so many other priorities. It could only have been by the help of God that I was able to achieve this goal.

Table of Contents

1 Temporal Bayes Networks

1.1 Definition

Most natural and man-made systems contain partial dependencies among their compositional elements. Researchers are particularly interested in developing an intuitive model that can be used to formulate problems and incorporate uncertain knowledge, but at the same time is a precise description of information that can be stored and manipulated by a computer.

The desired model would be a graphical representation of uncertain quantities that explicitly reveals probabilistic dependence and the flow of information. It would be compact and intuitive, emphasizing the relationship among variables, and yet it must represent a complete probabilistic description of the problem.

Numerous researchers have defined and explored the merits of a particular type of probabilistic graphical representation that uses directed acyclic graphs in which the nodes represent propositions (or variables), the arcs signify the existence of direct causal influences between the linked propositions, and the strengths of these influences are quantified by conditional probabilities. These graphical representations have been called *Bayes networks* (Pearl 1988), *belief networks* (Duda, Hart and Nilsson 1981), *influence diagrams* (Shachter 1986) and *probability networks* (Dean and Kanazawa 1989).

A *temporal Bayes network* is a specialization of these networks, that represents the relationships between variables at successive points in discretized time. The network is in the form of a grid where points in discretized time form the columns in the graph, with each of the variables forming a row in the graph. The nodes correspond to states of propositional variables at points in time.

The propositional variables may be of two types: those traditionally referred to as *fluents*, which, if they become true, tend to persist without additional effort; and those corresponding to the occurrence of *events*, which, if true at a point, tend to prompt a change of state of other variables (Dean and Kanazawa 1988). Let *holds(P, t)* indicate that the fluent P is true at time t, and *occurs(E, t)* indicate that an event of type E occurs at time t. The notation E_P indicates an event that generally causes the fluent P to become true, while $E_{\neg P}$ indicates an event that generally causes the fluent P to become false.

The arcs in the graph are used to indicate dependence between two variables. They are always directed from a variable at one point in time to a variable at the next point in time. At each node we must specify conditional probabilities and prior probabilities for each possible combination of values of the dependent variables.

Let A denote the state of the variable at the current node (for instance, $A \equiv holds(fluent, t)$ or $A \equiv occurs(event, t)$), let n denote the number of variables the current variable is dependent upon, and let C_i indicate the value of dependent variable i at the previous point in time, $t - \Delta$. Then we need 2^n conditional probabilities of the form $p(A \mid C_1 \wedge C_2 \wedge \ldots \wedge C_n)$ and we need 2^n prior probabilities of the form $p(C_1 \wedge C_2 \wedge \ldots \wedge C_n)$, that correspond to the 2^n possible combinations of values for the n dependent variables.

To predict the value of A at this node, we use the model

$$p(A) = \sum p(A \mid C_1 \wedge C_2 \wedge \ldots \wedge C_n)\, p(C_1 \wedge C_2 \wedge \ldots \wedge C_n)$$

where the summation is taken over the 2^n possible combinations. As discussed by Pearl (1988), the unique distribution corresponding to the model is given by

$$p(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} p(V_i \mid S_i)$$

where the V_i denote the propositional variables in the model, and S_i is the conjunction of boolean variables associated with those nodes for which there exist arcs to V_i in the network.

A general temporal Bayes network (without the specific conditional and prior probabilities at each node) is shown in Figure 1.

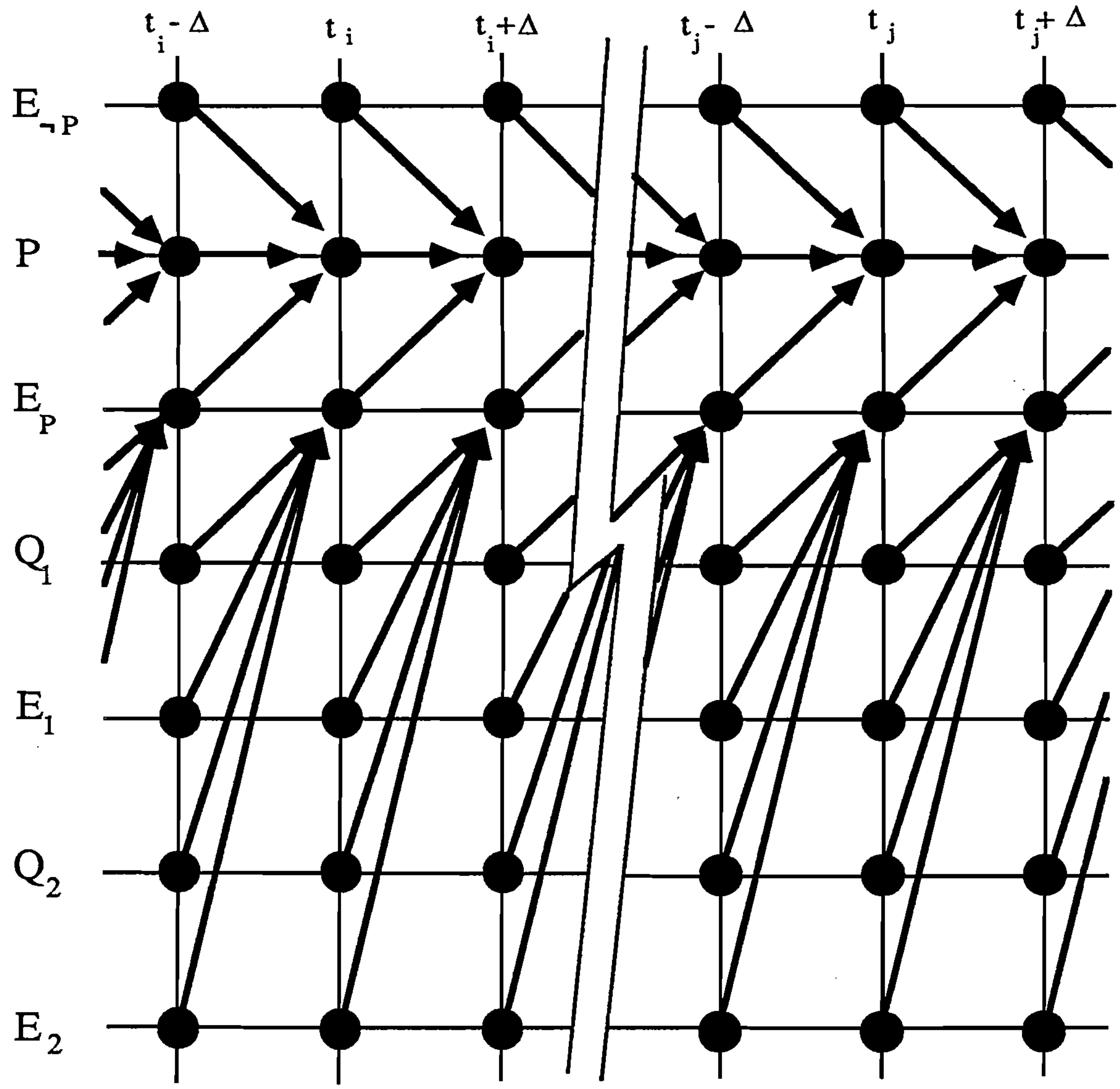

Figure 1: A General Temporal Bayes Network

The fluent P is dependent on events occurring that generally cause P to become true or false, E_P and $E_{\neg P}$ respectively, as well as on P itself at the previous time point - the *persistence* factor. The event E_P is dependent on the conditions for its occurrence being right - the *causation* factor (Dean and Kanazawa 1987). For instance, proposition Q_1 being true and event E_1 occurring or proposition Q_2 being true and event E_2 occurring. Thus E_P is depicted as being dependent on Q_1, E_1, Q_2 and E_2, at each successive time point.

1.2 Uses

A temporal Bayes network is a graphical representation for probabilistic models that clearly indicates the assumptions concerning dependence and independence between the variables. Simply by inspecting the graph, one can identify the conditional dependence inherent in a model. Therefore, a temporal Bayes network is a convenient method of constructing a model and verifying its correctness.

The temporal Bayes network is designed to simplify certain computations generally used in planning and decision support. For instance, the temporal Bayes network can be used to answer questions of the form "What is the probability of P at t given everything else we know about the situation?"

2 Markov Random Process Transition Tables

2.1 Definition

A *Markov chain* is a special type of stochastic process and a stochastic process is a collection of random variables. More specifically, a *stochastic process* is defined to be a family of random variables defined on some sample space, Ω (Grenander and Rosenblatt 1957). The set of distinct values assumed by a stochastic process is called the *state space*. If the state space of a stochastic process is countable, or finite, the process is called a *chain*.

A stochastic process $\{X_k\}$, $k = 1, 2, 3, \ldots$ with state space $S = \{1, 2, 3, \ldots\}$ is said to satisfy the *Markov property* (Isaacson and Madson 1976) if for every n and all states $i_1, i_2, \ldots, i_n$ it is true that

$$p(X_n = i_n \mid X_{n-1} = i_{n-1}, X_{n-2} = i_{n-2}, \ldots, X_1 = i_1) = p(X_n = i_n \mid X_{n-1} = i_{n-1})$$

Roughly speaking, the Markov property is satisfied if the future state of the variables depends on the present state, but not on past states. Once a stochastic process falls into the subclass of a discrete-time Markov chain, the movement of the process among the states of S is determined by the conditional probabilities $p(X_n = j \mid X_{n-1} = i)$, often called the *transition probabilities* (Revuz 1975).

A discrete-time Markov chain is said to be *stationary* or *homogeneous in time* if the probability of going from one state to another is independent of the time at which the step is being made (Adke and Manjunath 1984). Let $\{X_k\}$ denote a discrete-time stationary Markov chain with a finite state space, $S = \{1, 2, \ldots, n\}$. For this chain, there are n^2 transition probabilities,

$$\{p_{ij}\} = p(X_n = j \mid X_{n-1} = i) \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, n.$$

Each transition probability p_{ij} is actually a conditional probability with the following meaning: p_{ij} = p(the process is in state *i* and goes to state *j* in the next step) / p(the process is in state *i*).

The most convenient way of recording these transition probabilities is in the form of a matrix or table T, as in Figure 2. This matrix, typically called the *transition probability matrix* or *transition table*, associates the i^{th} row and column of T with the i^{th} state of S. It contains all of the relevant information regarding the movement of the process among the states in S, and has the following properties (Rosenblatt 1962):

i) all the entries are non-negative,
ii) the sum of the entries in each row is one.

$$T = \begin{bmatrix} p_{11} & p_{12} & p_{13} & \cdots & p_{1n} \\ p_{21} & p_{22} & p_{23} & \cdots & p_{2n} \\ \vdots & & & & \\ p_{n1} & p_{n2} & p_{n3} & \cdots & p_{nn} \end{bmatrix}$$

Figure 2: A General Transition Table

2.2 Uses

The transition matrix contains all the information needed to describe the motion of the chain among the states in S. However, if you are interested in where the process is at any particular time, you must first know where the chain started (Isaacson and Madson 1976).

A vector $\mathbf{a}_0 = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is called a *starting vector* if

$$\sum_{i=1}^{n} \alpha_i = 1 \quad \text{and} \quad \alpha_i \geq 0 \quad \text{for } i = 1, 2, \ldots, n.$$

In the case where the chain starts deterministically at one state, $\mathbf{a}_0$ has a one in the coordinate corresponding to that state and zeros elsewhere. In general, the process can start at various states according to some probability distribution, given by the starting vector. The starting vector is referred to as the distribution at time zero, and $\alpha_k = p(X_0 = k)$, $k = 1, 2, \ldots, n$.

Now consider how to determine where the chain will be after m time steps. First, consider the problem of finding $p(X_1 = i)$, the distribution after 1 step. Using conditional probabilities, this can be written as

$$p(X_1 = i) = p(X_0 = 1)\, p(X_1 = i) \mid X_0 = 1) + p(X_0 = 2)\, p(X_1 = i) \mid X_0 = 2) + \ldots + p(X_0 = n)\, p(X_1 = i) \mid X_0 = n)$$

$$= \sum_{j=1}^{n} \alpha_j\, p_{ji}$$

Similarly, the distribution after two steps can be written as

$$p(X_2 = i) = \sum_{k=1}^{n} \sum_{j=1}^{n} \alpha_j\, p_{jk}\, p_{ki}$$

and in the general case, the distribution after m steps is given as

$$p(X_{j_m} = i) = \sum_{j_m=1}^{n} \sum_{j_{m-1}=1}^{n} \ldots \sum_{j_2=1}^{n} \sum_{j_1=1}^{n} \alpha_{j_1}\, p_{j_1 j_2}\, p_{j_2 j_3} \cdots p_{j_{m-1} j_m}\, p_{j_m i}$$

The matrix notation for the transition table T is ideally suited for this problem. The expression for $p(X_1 = i)$, the probability of being in state i after one step, is simply the i^{th} coordinate of the vector $a_1 = a_0 T$, the vector that represents the distribution of where the Markov chain is after one step. Similarly, the vector that represents the distribution of where the Markov chain is after two steps is $a_2 = a_1 T = (a_0 T)T = a_0 T^2$, and the i^{th} coordinate of a_2 is $p(X_2 = i)$. In general, the distribution of where the process is after m steps, given that the starting vector was a_0, can be determined by $a_m = a_0 T^m$.

3 Equivalence of the Two Models

The first question faced by someone who wants to use the theory of Markov chains is whether or not the process is Markov. The temporal Bayes network described in section one meets the criteria to be a discrete-time Markov chain. First, it is *discrete-time* because it uses a discrete approximation of time with a fixed Δ, the intervals that form the columns of the network. Second, it is a *chain* because there are a finite number of states in the state space. In fact, there are 2^n states in the state space for a temporal Bayes network of n variables, one for each possible combination of the truth or falsehood of each of the variables. Finally, it satisfies the *Markov* property, because the state of each variable at time $t + \Delta$ is independent of the states of variables at time $t - \Delta$, given the state of the variables at time t.

If we add the assumption that each of the conditional probabilities remain the same across each time point, the temporal Bayes network is also *stationary*. This does not appear to be a limiting assumption, since we could simply add another variable to the network, to factor in the time dependency. For instance, if the conditional probabilities at two time points differ because a certain time point has been reached (for example, the end of the work day or the start of a new shift), we could add another variable to the network to indicate whether the specified time has been reached. We would then combine the two sets of conditional probabilities into one consistent set by conditioning on the new variable.

Once you have determined that you are working with a discrete-time stationary Markov chain, the next step is to find the transition matrix. In some cases, it is not hard to determine the appropriate state space and the transition probabilities necessary for describing the Markov chain of interest. However, in some situations this determination is quite difficult.

At this point, the distinction between the theory and applications of Markov chains must be understood. The theory of Markov chains states that each discrete-time stationary Markov chain with finite state space of size m has m^2 transition probabilities that can be formed into a transition table. In practice, however, the appropriate transition table for the experiment in question must be found. The algorithm for determining the appropriate transition table, given a temporal Bayes network involving n variables, is the subject of the next section.

Section five gives the algorithm for obtaining the equivalent temporal Bayes network, given a Markov Random Process transition table. Using these two algorithms, you can convert from one format to the other without loss of information. Since the two models are equivalent in predictive or expressive power, you can choose the more appropriate model for any given situation.

4 Conversion from Temporal Bayes Network to Markov Random Process Transition Table

4.1 Algorithm in General

This section describes the algorithm that converts a temporal Bayes network into a Markov Random Process transition table that encodes the same information. Suppose we are given a temporal Bayes network that involves n variables, and for each variable V_j, $j = 1, 2, \ldots, n$, we are given all of the appropriate conditional probabilities $cp_i(V_j) = p(V_j \text{ at } t + \Delta \mid C_i \text{ at } t)$ and all of the prior probabilities $pp_i = p(C_i)$ where the C_i range over all combinations of dependent variables. Since there are n variables, there are 2^n possible combinations of the values of the n variables and the size of the state space is 2^n. The information provided by the conditional probabilities can be used to create the corresponding $2^n \times 2^n$ matrix of transition probabilities, called the transition table.

The information provided by the prior probabilities is not used in the calculation of the entries of the transition table, since each of the transition probabilities is actually a conditional probability. Knowing that the probability of a particular state S_i is initially zero would tempt one to say that the transition probability p_{ij} = p(the process is in state *i* and goes to state *j* in the next step) / p(the process is in state *i*) is undefined, or at the very least, that the chain is not stationary. Instead, the prior probabilities determine the initial distribution, or starting vector for the transition table.

Each row of the transition table corresponds to one of the 2^n possible states at time t. For each of these initial states, there are 2^n possible states at the next time point, $t + \Delta$. The probabilities of going from one initial state to each of those 2^n possible next states are recorded in one row of the transition table.

These probabilities are calculated from the conditional probabilities that were specified as part of the temporal Bayes network. The initial state specifies the subset of conditional probabilities that are applicable for each row in the transition table. The values of the variables in each of the output states determine whether the conditional probability $cp_i(V_j)$ itself is used, or whether its complement $cp_i(\neg V_j) = 1 - cp_i(V_j)$ is used. The appropriate conditional probabilities for each of the variables are multiplied together to obtain the probability of each output state, given the current input state. The next subsection illustrates this process using a simple example.

4.2 Example

Suppose we are given a temporal Bayes network as depicted in Figure 3, with the time-interval set to one day. The 4 variables can be described as follows:

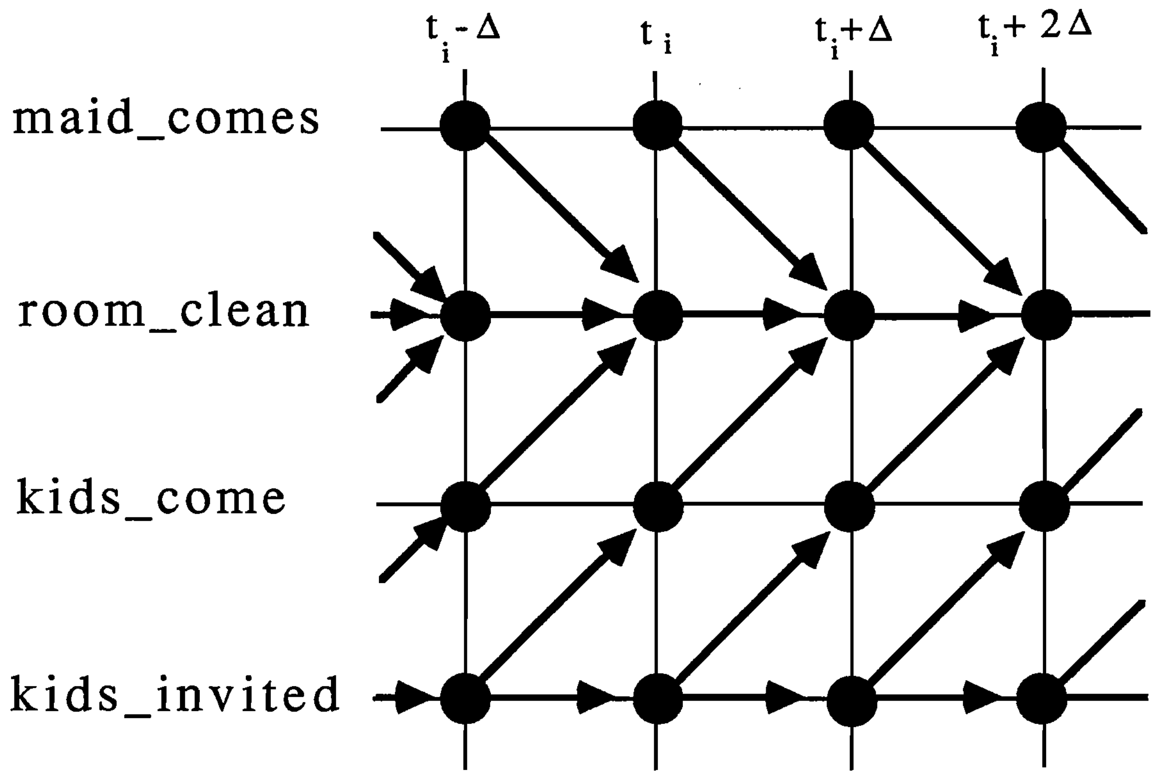

Figure 3: A Specific Temporal Bayes Network

1) maid_comes (MC) - the event of the maid coming to your home to clean up a specified room

2) room_clean (RC) - a fluent that indicates whether the specified room could be called clean, or tidy

3) kids_come (KC) - the event of a group of children coming to your home to play for a while

4) kids_invited (KI) - the event of inviting the group of children to come to your home the next day

The dependencies between these variables are described as follows:

- maid_comes (MC) is independent of all other variables, since the maid has been hired to come once a week.
- room_clean (RC) is dependent upon the state of 3 other variables at the previous time - whether the maid came to clean it up (MC), whether the room was clean to start with (RC), and whether the group of kids came to mess it up (KC)
- kids_come (KC) is dependent on one variable - whether or not they were invited (KI)
- kids_invited (KI) is also dependent on this one variable (KI), that is, the likelihood of inviting them one particular day depends on whether you just invited them the previous day

Suppose also that the conditional probabilities are specified as:

$cp_1(MC) = p(MC \text{ at } t + \Delta) = 0.15$

$cp_1(RC) = p(RC \text{ at } t + \Delta \mid MC \wedge RC \wedge KC \text{ at } t) = 0.3$
$cp_2(RC) = p(RC \text{ at } t + \Delta \mid MC \wedge RC \wedge \neg KC \text{ at } t) = 1.0$
$cp_3(RC) = p(RC \text{ at } t + \Delta \mid MC \wedge \neg RC \wedge KC \text{ at } t) = 0.4$
$cp_4(RC) = p(RC \text{ at } t + \Delta \mid MC \wedge \neg RC \wedge \neg KC \text{ at } t) = 1.0$
$cp_5(RC) = p(RC \text{ at } t + \Delta \mid \neg MC \wedge RC \wedge KC \text{ at } t) = 0.0$
$cp_6(RC) = p(RC \text{ at } t + \Delta \mid \neg MC \wedge RC \wedge \neg KC \text{ at } t) = 0.8$
$cp_7(RC) = p(RC \text{ at } t + \Delta \mid \neg MC \wedge \neg RC \wedge KC \text{ at } t) = 0.0$
$cp_8(RC) = p(RC \text{ at } t + \Delta \mid \neg MC \wedge \neg RC \wedge \neg KC \text{ at } t) = 0.05$

$cp_1(KC) = p(KC \text{ at } t + \Delta \mid KI \text{ at } t) = 1.0$
$cp_2(KC) = p(KC \text{ at } t + \Delta \mid \neg KI \text{ at } t) = 0.1$

and

$cp_1(KI) = p(KI \text{ at } t + \Delta \mid KI \text{ at } t) = 0.0$
$cp_2(KI) = p(KI \text{ at } t + \Delta \mid \neg KI \text{ at } t) = 0.3$

The algorithm described in the previous subsection can be applied to this information to create the corresponding transition table. Since there are 4 variables, there are $2^4 = 16$ possible combinations of the values of these variables. Let the states be numbered as follows:

$$
\begin{array}{lcl}
S_0 & = & (\neg MC \wedge \neg RC \wedge \neg KC \wedge \neg KI) \\
S_1 & = & (\neg MC \wedge \neg RC \wedge \neg KC \wedge KI) \\
S_2 & = & (\neg MC \wedge \neg RC \wedge KC \wedge \neg KI) \\
S_3 & = & (\neg MC \wedge \neg RC \wedge KC \wedge KI) \\
S_4 & = & (\neg MC \wedge RC \wedge \neg KC \wedge \neg KI) \\
S_5 & = & (\neg MC \wedge RC \wedge \neg KC \wedge KI) \\
S_6 & = & (\neg MC \wedge RC \wedge KC \wedge \neg KI) \\
S_7 & = & (\neg MC \wedge RC \wedge KC \wedge KI) \\
S_8 & = & (MC \wedge \neg RC \wedge \neg KC \wedge \neg KI) \\
S_9 & = & (MC \wedge \neg RC \wedge \neg KC \wedge KI) \\
S_{10} & = & (MC \wedge \neg RC \wedge KC \wedge \neg KI) \\
S_{11} & = & (MC \wedge \neg RC \wedge KC \wedge KI) \\
S_{12} & = & (MC \wedge RC \wedge \neg KC \wedge \neg KI) \\
S_{13} & = & (MC \wedge RC \wedge \neg KC \wedge KI) \\
S_{14} & = & (MC \wedge RC \wedge KC \wedge \neg KI) \\
S_{15} & = & (MC \wedge RC \wedge KC \wedge KI)
\end{array}
$$

Let the first row of the transition table correspond to starting with initial state S_0, in which all 4 variables are FALSE. The subset of conditional probabilities that are applicable for this initial state contains

$$
\begin{array}{ll}
cp_1(MC) & = p(MC \text{ at } t + \Delta) = 0.15 \\
cp_8(RC) & = p(RC \text{ at } t + \Delta \mid \neg MC \wedge \neg RC \wedge \neg KC \text{ at } t) = 0.05 \\
cp_2(KC) & = p(KC \text{ at } t + \Delta \mid \neg KI \text{ at } t) = 0.1 \\
cp_2(KI) & = p(KI \text{ at } t + \Delta \mid \neg KI \text{ at } t) = 0.3
\end{array}
$$

and their complements, that are

$$
\begin{array}{llll}
cp_1(\neg MC) & = 1 - cp_1(MC) & = 1 - 0.15 & = 0.85 \\
cp_8(\neg RC) & = 1 - cp_8(RC) & = 1 - 0.05 & = 0.95 \\
cp_2(\neg KC) & = 1 - cp_2(KC) & = 1 - 0.1 & = 0.9 \\
cp_2(\neg KI) & = 1 - cp_2(KI) & = 1 - 0.3 & = 0.7
\end{array}
$$

The appropriate conditional probability (or its complement) for each of the n variables are multiplied together to obtain the transition probability for each of the 2^n possible output states. For example, in output state S_0 all 4 variables are FALSE, so the appropriate conditional probabilities to be multiplied are $cp_1(\neg MC)$, $cp_8(\neg RC)$, $cp_2(\neg KC)$ and $cp_2(\neg KI)$ and the transition probability is

$$\begin{aligned} p(S_0 \text{ at } t + \Delta \mid S_0 \text{ at } t) = p_{00} &= cp_1(\neg MC)\ cp_8(\neg RC)\ cp_2(\neg KC)\ cp_2(\neg KI) \\ &= (0.85)\ (0.95)\ (0.9)\ (0.7) \\ &= 0.508725 \end{aligned}$$

Similarly,

$p_{01} = cp_1(\neg MC)\ cp_8(\neg RC)\ cp_2(\neg KC)\ cp_2(KI) = 0.218025$

$p_{02} = cp_1(\neg MC)\ cp_8(\neg RC)\ cp_2(KC)\ cp_2(\neg KI) = 0.056525$

.

.

.

$p_{0\ 15} = cp_1(MC)\ cp_8(RC)\ cp_2(KC)\ cp_2(KI) = 0.000225$

These probabilities form the first row of the transition table. Let the second row correspond to initial state S_1; the appropriate subset of conditional probabilities and their complements contains

$cp_1(MC) = 0.15$	$cp_1(\neg MC) = 0.85$
$cp_8(RC) = 0.05$	$cp_8(\neg RC) = 0.95$
$cp_1(KC) = 1.0$	$cp_1(\neg KC) = 0.0$
$cp_1(KI) = 0.0$	$cp_1(\neg KI) = 1.0$

As with the first row, the transition probability for each output state is calculated by multiplying the appropriate conditional probability (or its complement) for each variable. Continuing this process for each row produces the transition table shown in Figure 4.

$$
T = \begin{bmatrix}
0.508725 & 0.218025 & 0.056525 & 0.024225 & 0.026775 & 0.011475 & 0.002975 & 0.001275 & 0.089775 & 0.038475 & 0.009975 & 0.004275 & 0.004725 & 0.002025 & 0.000525 & 0.000225 \\
0.000000 & 0.000000 & 0.807500 & 0.000000 & 0.000000 & 0.000000 & 0.042500 & 0.000000 & 0.000000 & 0.000000 & 0.142500 & 0.000000 & 0.000000 & 0.000000 & 0.007500 & 0.000000 \\
0.535500 & 0.229500 & 0.059500 & 0.025500 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.094500 & 0.040500 & 0.010500 & 0.004500 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \\
0.000000 & 0.000000 & 0.850000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.150000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \\
0.107100 & 0.045900 & 0.011900 & 0.005100 & 0.428400 & 0.183600 & 0.047600 & 0.020400 & 0.018900 & 0.008100 & 0.002100 & 0.000900 & 0.075600 & 0.032400 & 0.008400 & 0.003600 \\
0.000000 & 0.000000 & 0.170000 & 0.000000 & 0.000000 & 0.000000 & 0.680000 & 0.000000 & 0.000000 & 0.000000 & 0.030000 & 0.000000 & 0.000000 & 0.000000 & 0.120000 & 0.000000 \\
0.535500 & 0.229500 & 0.059500 & 0.025500 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.094500 & 0.040500 & 0.010500 & 0.004500 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \\
0.000000 & 0.000000 & 0.850000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.150000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.535500 & 0.229500 & 0.059500 & 0.025500 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.094500 & 0.040500 & 0.010500 & 0.004500 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.850000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.150000 & 0.000000 \\
0.321300 & 0.137700 & 0.035700 & 0.015300 & 0.214200 & 0.091800 & 0.023800 & 0.010200 & 0.056700 & 0.024300 & 0.006300 & 0.002700 & 0.037800 & 0.016200 & 0.004200 & 0.001800 \\
0.000000 & 0.000000 & 0.510000 & 0.000000 & 0.000000 & 0.000000 & 0.340000 & 0.000000 & 0.000000 & 0.000000 & 0.090000 & 0.000000 & 0.000000 & 0.000000 & 0.060000 & 0.000000 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.535500 & 0.229500 & 0.059500 & 0.025500 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.094500 & 0.040500 & 0.010500 & 0.004500 \\
0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.850000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.000000 & 0.150000 & 0.000000 \\
0.374850 & 0.160650 & 0.041650 & 0.017850 & 0.160650 & 0.068850 & 0.017850 & 0.007650 & 0.066150 & 0.028350 & 0.007350 & 0.003150 & 0.028350 & 0.012150 & 0.003150 & 0.001350 \\
0.000000 & 0.000000 & 0.595000 & 0.000000 & 0.000000 & 0.000000 & 0.255000 & 0.000000 & 0.000000 & 0.000000 & 0.105000 & 0.000000 & 0.000000 & 0.000000 & 0.045000 & 0.000000
\end{bmatrix}
$$

Figure 4: A Specific Transition Table

5 Conversion from Markov Random Process Transition Table to Temporal Bayes Network

5.1 Algorithm in General

This section describes the algorithm that converts a Markov Random Process transition table of the appropriate format into a temporal Bayes network that encodes the same information. Suppose we are given a model of a problem in the form of a Markov random process in which time is discrete and the state space S corresponds to all possible valuations of a finite set of boolean variables $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$. Given such a model, including the $2^n \times 2^n$ transition table that defines the transition probabilities for all states in S, we can use the following algorithm to transform the description of the problem into the equivalent temporal Bayes network.

Recall that a temporal Bayes network is a graphical representation that indicates dependencies between the n variables, and for each variable, specifies conditional probabilities $cp_i(V_j) = p(V_j \text{ at } t + \Delta \mid C_i \text{ at } t)$ and prior probabilities $pp_i = p(C_i)$ where the C_i range over all combinations of the dependent variables.

If a starting vector is specified with the transition table, this vector uniquely defines the entire set of prior probabilities for the first time point in the temporal Bayes network. If a starting vector is not specified, the entire set of general prior probabilities for an arbitrary time point can be calculated by summing the columns of the transition table and normalizing by the number of initial states. The temporal Bayes network stores a subset of these prior probabilities at the nodes, specified by the combinations of dependent variables.

The algorithm for determining the dependencies and the conditional probabilities for each variable is described below. A detailed example is given in the next subsection.

The first step is to determine the probability of each variable being TRUE, given each of the 2^n initial states, $p(V_j \mid S_i)$ $j = 1, 2, \ldots, n$; $i = 0, 1, \ldots, 2^{n-1}$. For each variable, we can then divide these 2^n probabilities into equivalence classes, where membership in an equivalence class is determined via the probability of the variable.

By examining which initial states are grouped into the same equivalence class, we can determine the dependencies for each variable. Then we need only locate each combination of values for the dependent variables among the equivalence classes to determine the appropriate conditional probabilities.

5.2 Example

Suppose we are given the Markov Random Process transition table specified in Figure 4. Each transition probability p_{ij} in the table represents the probability of moving from state i to state j in one time step. Since there are $2^n = 16$ states in the table, the appropriate temporal Bayes network involves 4 variables. The algorithm described in the previous subsection can be applied to determine the remaining information to completely define the corresponding temporal Bayes network.

Since a starting vector is not specified, the set of prior probabilities can be calculated by summing the $2^n = 16$ entries in each column of the matrix and dividing by 16 to normalize the probability. For example,

$$pp_0 = 1/16 \sum_{i=0}^{15} p_{i0}$$

$$= 1/16 \; (0.508725 + 0 + 0.5355 + \ldots + 0.37485 + 0)$$
$$= 0.148936$$

In a similar manner, the prior probabilities for the remaining 15 states can be calculated. The interested reader can refer to Appendix C for a complete listing of the actual values.

To determine the probability of a particular variable V_k being TRUE at time $t + \Delta$, given an initial state S_i at t, we must sum the transition probabilities for all states S_j in which the variable is TRUE. For instance, the variable V_1 is TRUE for the eight states S_8 through S_{15} , the variable V_2 is TRUE for the four states S_4 through S_7 and the four states S_{12} through S_{15} , the variable V_3 is TRUE for the eight states S_2, S_3, S_6, S_7, S_{10}, S_{11}, S_{14} and S_{15} , and the variable V_4 is TRUE for the eight states S_j where j is an odd number. Using this method, we can calculate

$$p(V_1 \mid S_0) = 0.089775 + 0.038475 + \ldots + 0.000225 = 0.15$$
$$p(V_2 \mid S_0) = 0.026775 + 0.011475 + \ldots + 0.000225 = 0.05$$
$$p(V_3 \mid S_0) = 0.056525 + 0.024225 + \ldots + 0.000225 = 0.1$$

.

.

.

$$p(V_4 \mid S_{15}) = 0.0 + 0.0 + \ldots + 0.0 = 0.0$$

Grouping these probabilities into equivalence classes where each member of an equivalence class has the same probability, we observe the following:

- V_1 has 1 equivalence class
  the probability of V_1 being TRUE is 0.15 for all 16 initial states

- V_2 has 6 equivalence classes
  the probability of V_2 being TRUE is 0.05 for 2 initial states
  the probability of V_2 being TRUE is 0.0 for 4 initial states
  the probability of V_2 being TRUE is 0.8 for 2 initial states
  the probability of V_2 being TRUE is 1.0 for 4 initial states
  the probability of V_2 being TRUE is 0.4 for 2 initial states
  the probability of V_2 being TRUE is 0.3 for 2 initial states

- V_3 has 2 equivalence classes
  the probability of V_3 being TRUE is 0.1 for 8 initial states
  the probability of V_3 being TRUE is 1.0 for 8 initial states

- V_4 has 2 equivalence classes
  the probability of V_4 being TRUE is 0.3 for 8 initial states
  the probability of V_4 being TRUE is 0.0 for 8 initial states

We can determine the dependencies for each variable by examining which initial states are grouped into the same equivalence class. There is only one equivalence class for V_1 that contains all the initial states. Therefore, the probability of V_1 being TRUE at a given time $t + \Delta$ is independent of the state of all variables at t. Thus, V_1 is dependent on 0 variables and no conditional probabilities are needed for V_1.

The remaining variables each have more than one equivalence class, which indicates that they are each dependent on at least one variable. Although some heuristics can be applied to improve the speed of determining the number of dependent variables, the algorithm described in this paper does not make use of them. The algorithm simply looks at which initial states are in each equivalence class, and repeatedly applies a combination rule until it is no longer applicable. This combination rule combines two initial states if they differ in the value of at most one variable, indicates that the value of that particular variable does not matter, and reduces the number of states in the equivalence class by one.

For example, the first equivalence class for V_2 contains the two initial states S_0 and S_1. These two initial states differ only in the value of the fourth variable, so they are combined into one state that indicates that V_4 does not matter, and that V_1 is FALSE, V_2 is FALSE and V_3 is FALSE. Since there is now only one state in the equivalence class, the combination rule can no longer be applied.

Applying this logic to the second equivalence class for V_2, which contains the initial states S_2, S_3, S_6, and S_7, we have the following:

1) Initial states S_2 and S_3 differ only in the value of the fourth variable, so they are combined into one state that indicates that V_4 does not matter, and that V_1 is FALSE, V_2 is FALSE and V_3 is TRUE.

2) Initial states S_6 and S_7 differ only in the value of the fourth variable, so they are combined into one state that indicates that V_4 does not matter, and that V_1 is FALSE, V_2 is TRUE and V_3 is TRUE.

3) The two states produced by steps 1 and 2 differ only in the value of the second variable, so they are combined into one state that indicates that V_2 and V_4 do not matter, and that V_1 is FALSE and V_3 is TRUE.

At this point there is only one state in the equivalence class, so the combination rule can no longer be applied.

Continuing this process for the remaining four equivalence classes for V_2, we find that V_4 can be reduced from all of them, and although V_2 can be reduced from some of them, it can not be reduced from all of them. Therefore V_2 is dependent on the 3 variables V_1, V_2 and V_3.

Following a similar reduction process for V_3 and V_4, we find that in both cases, V_1, V_2 and V_3 can be reduced from both equivalence classes, but V_4 can not. Therefore, both V_3 and V_4 are dependent on the state of V_4.

To determine the appropriate conditional probabilities for each combination of the dependent variables, we need only look at the probabilities associated with the equivalence classes. For each combination of the dependent variables, we must look at the equivalence classes to find the one that contains the appropriate combination. For example, the conditional probability

$$p(V_2 \text{ at } t + \Delta \mid \neg V_1 \wedge \neg V_2 \wedge \neg V_3 \text{ at } t) = 0.05$$

since that combination of dependent variables is found in the first equivalence class for V_2, the one with probability 0.05. Similarly, the conditional probability

$$p(V_2 \text{ at } t + \Delta \mid \neg V_1 \wedge \neg V_2 \wedge V_3 \text{ at } t) = 0.0$$

since that combination of dependent variables is found in the second equivalence class for V_2, the one with probability 0.0.

Thus, we have obtained all of the necessary information to define a temporal Bayes network. In fact, the temporal Bayes network we have defined is precisely the one depicted in Figure 3, if we name V_1 as maid_comes, V_2 as room_clean, V_3 as kids_come and V_4 as kids_invited.

6 Summary

I have defined the concepts of a temporal Bayes network and a Markov Random Process transition table, and have shown that they encode equivalent information. The algorithms I detailed in sections four and five can be used to convert from one representation to the other without loss of information.

Therefore, given a description of a problem in either format, one can easily transform the model to the other format. Temporal Bayes networks, being graphical by nature, make it easy to determine the completeness and accuracy of a model. They also provide an encoding that, in most cases, is considerably more compact than the Markov Random Process transition table. Both methods can be used to predict the state of variables in the future, given knowledge of their current values.

Related Work

Although the theory of Markov chains is relatively well-established and well-known, temporal Bayes networks are a subject of current research. An initial motivation for their study was the desire to combine decision theory techniques under conditions of uncertainty with symbolic problem-solving techniques predominant in artificial intelligence (Feldman and Yakimovsky 1974; Feldman and Sproull 1977; Hanks 1987).

Judea Pearl has established a basis for the theory and shown that the centuries-old Bayes formula, the likelihood-ratio updating rule, can be used to propagate the impacts of new beliefs and/or new evidence in large multi-hypotheses inference systems (Pearl 1982). He describes a method of passing new information through the networks in such a way that, when equilibrium is reached, each proposition's belief is consistent with the axioms of probability theory (Pearl 1986).

However, while Kim and Pearl (1983) have described an efficient method for computing the joint distribution for singly-connected networks by local propagation of Bayes factor (the likelihood ratios $p(x) / p(\neg x)$), the general problem of probabilistic inference in multiply-connected networks has been shown to be NP-hard (Cooper 1987). Several researchers have proposed algorithms that consider trade-offs in terms of efficiency, soundness or completeness.

Pearl (1985) presents an algorithm that exploits the topology of the network by instantiating a set of nodes corresponding to a cutset of the underlying graph, thereby making a multiply-connected graph singly-connected. The algorithm involves conditioning on all value combinations of the variables in the cutset and computing a weighted average of the joint conditional probability distributions for all possible instantiations.

The Lauritzen-Spiegelhalter algorithm also exploits the topology of the network, rendering a multiply-connected network singly-connected. It alters the connectivity of the network by adding a set of subsidiary arcs, so that there are no cycles of length 4 or more without a chord or shortcut. Then the joint distribution can be efficiently computed in terms of the cliques of the original graph (Lauritzen and Spiegelhalter 1988).

Several researchers have explored the usefulness of bounding or approximation algorithms, since they can have significant advantage in situations where the time spent in computing is important (Cooper 1984; Dean and Boddy 1988; Horvitz 1988). Bounding algorithms look at the constraints and bound the distribution, typically by supplying an upper and lower bound that are successively refined such that they approach the correct distribution in the limit (Horvitz 1988; Henrion 1988b).

Approximation algorithms, typically known as Monte Carlo simulation algorithms, simulate the states that a network is likely to go through, given a set of constraints. The aim is to develop an algorithm that comes close to the correct answer, by iteratively refining the algorithm at each time step. Unfortunately, the Monte Carlo algorithms developed to date for probability networks (Henrion 1988a; Pearl 1987) have been somewhat less than ideal and exhibit pathological behavior in the case of certain network topologies involving strong dependence between nodes (Chin and Cooper 1987).

Several other researchers have focused on the case of an incomplete causal model, in which only a subset of the conditional probabilities are known. Cheeseman (1983) proposes a method for calculating the conditional probability of any multi-valued predicate, given particular information about the individual case. This method is known as maximum entropy, since the maximum entropy distribution is the one that assumes the least information. Goldman and Rivest (1986) augment this method by integrating it with the planning of data collection and tabulation, since their procedure requires tabulating additional constraints. Geman and Geman (1984) and Lippman (1986) describe a maximum entropy method based on stochastic relaxation.

Wellman (1988) explores the case of an incomplete causal model from the aspect of the conditional probabilities. In his model, the constraints on the joint probability distribution over the variables are encoded only as qualitative relationships, instead of the usual numeric representations.

Along the lines of ease of construction, Henrion (?) proposes some techniques that can facilitate the process of structuring and quantifying uncertain relationships in a diagram of moderate size. He also discusses general issues of analyzing the sensitivity of conclusions to errors and approximations in assessed probabilities.

Once a problem has been defined, Shachter (1987) proposes that a solution can be computed by manipulating the influence diagram through a series of transformations to the model that preserve the solution value, all of which can be accomplished on a personal computer (Shachter 1988). Shachter and Heckerman (1987) suggest that a reasonable approach would be to construct a model with the emphasis on the arcs in one direction, and then to reverse the direction of the arcs.

Shachter and Kenley (1988) discuss the relationships between linear-quadratic Gaussian models and covariance matrix representations for the multivariate normal distributions. Shachter, Eddy and Hasselblad (1988) discuss the use of these networks in the health field in general and give details of its use in one particular example.

References

S. R. Adke and S. M. Manjunath. 1984. An Introduction to Finite Markov Processes, John Wiley & Sons, New York.

Peter Cheeseman. 1983. A Method of computing generalized Bayesian probability values for expert systems. In Proceedings of IJCAI 8, Karlsruhe, West Germany, pp. 198-202.

Homer L. Chin and Gregory F. Cooper. 1987. Stochastic simulation of causal Bayesian models. Stanford University Knowledge Systems Laboratory, Memo KSL-87-22.

Gregory F. Cooper. 1984. NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge. PhD thesis, Stanford University.

Gregory F. Cooper. 1987. Probabilistic inference using belief networks is np-hard. Stanford University Knowledge Systems Laboratory, Memo KSL-87-27.

Thomas Dean and Mark Boddy. 1988. An analysis of time-dependent planning. In Proceedings AAAI-88, St. Paul, Minnesota, pp. 49-54.

Thomas Dean and Keiji Kanazawa. 1987. Persistence and probabilistic inference. Brown University Department of Computer Science, Technical Report CS-87-23.

Thomas Dean and Keiji Kanazawa. 1988. Probabilistic temporal reasoning. In Proceedings AAAI-88, St. Paul, Minnesota, pp. 524-528.

Thomas Dean and Keiji Kanazawa. 1989. A Model for reasoning about persistence and causation. Brown University Department of Computer Science, Technical Report CS-89-04.

R. O. Duda, P. E. Hart and N. J. Nilsson. 1981. Subjective Bayesian methods for rule-based inference systems. In B. W. Webber and N. J. Nilsson, editors, Readings in Artificial Intelligence. Tioga, Palo Alto, California.

Jerome A. Feldman and Robert F. Sproull. 1977. Decision theory and artificial intelligence: II. The hungry monkey. Cognitive Science **1**, pp. 158-192.

Jerome A. Feldman and Yoram Yakimovsky. 1974. Decision theory and artificial intelligence: I. A semantics-based region analyzer. Artificial Intelligence **5**, pp. 349-371.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence **6**, pp. 721-741.

Sally A. Goldman and Ronald L. Rivest. 1986. Making maximum entropy computations easier by adding extra constraints. In Proceedings of the Sixth Annual Workshop on Maximum Entropy and Bayesian Methods in Applied Statistics.

Ulf Grenander and Murray Rosenblatt. 1957. Statistical Analysis of Stationary Time Series, John Wiley & Sons, Canada.

Steve Hanks. 1987. Temporal reasoning about uncertain worlds. pp. 114-122.

Max Henrion. ?. Practical issues in constructing a Bayes' belief network.

Max Henrion. 1988. Propagating uncertainty by logic sampling in Bayes' networks. In John F. Lemmer and Laveen F. Kanal, editors, Uncertainty in Artificial Intelligence Vol II. Elsevier Science Publishers B. V., North-Holland, pp. 149-163.

Max Henrion. 1988. Towards efficient probabilistic diagnosis in multiply connected belief networks. In Proceedings of the Conference on Influence Diagrams.

Eric J. Horvitz. 1988. Reasoning under varying and uncertain resource constraints. In Proceedings AAAI-88, St. Paul, Minnesota, pp. 111-116.

Dean L. Isaacson and Richard W. Madson. 1976. Markov Chains: Theory and Applications, John Wiley & Sons, New York.

Jin H. Kim and Judea Pearl. 1983. A Computational model for causal and diagnostic reasoning in influence systems. In Proceedings of IJCAI 8, Karlsruhe, West Germany.

Stephen L. Lauritzen and David J. Spiegelhalter. 1988. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society Series B, **50**, pp. 157-194.

Alan F. Lippman. 1986. A Maximum entropy method for expert system construction. PhD thesis, Brown University.

Judea Pearl. 1982. Reverend Bayes on inference engines: a distributed hierarchical approach. In Proceedings of AAAI-82, Pittsburgh, Pennsylvania, pp. 133-136.

Judea Pearl. 1985. A Constraint-propagation approach to probabilistic reasoning. In Proceedings of the Workshop on Uncertainty and Probability in AI, Los Angeles, California, pp. 31-42.

Judea Pearl. 1986. Fusion, propagation, and structuring in belief networks. Artificial Intelligence **29**, pp. 241-288.

Judea Pearl. 1987. Evidential reasoning using stochastic simulation of causal models. Artificial Intelligence **32**, pp. 245-257.

Judea Pearl. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan-Kaufman, Los Altos, California.

D. Revuz. 1975. Markov Chains, North Holland Publishing Company, Amsterdam.

Murray Rosenblatt. 1962. Random Processes, Oxford University Press, New York.

Ross D. Shachter. 1986. Evaluating influence diagrams. Operations Research, **34**, pp. 871-882.

Ross D. Shachter. 1987. Probabilistic inference and influence diagrams. Operations Research.

Ross D. Shachter. 1988. DAVID: Influence diagram processing system for the Macintosh. In J. F. Lemmer and L. N. Kanal, editors, Uncertainty in Artificial Intelligence Vol II. Elsevier Science Publishers B. V., North-Holland, pp. 191-196.

Ross D. Shachter, David M. Eddy, and Victor Hasselblad. 1988. An Influence diagram approach to the confidence profile method for health technologies assessment.

Ross D. Shachter and David E. Heckerman. 1987. Thinking backward for knowledge acquisition. AI Magazine **8**, Fall 1987, pp. 55-61.

Ross D. Shachter and C. Robert Kenley. 1988. Gaussian influence diagrams. Center for Health Policy Research and Education, Duke University.

Michael P. Wellman. 1988. Foundations of qualitative probabilistic networks.

Appendix A

Structures

```
File: c:\CONVERT\convert.h    Creation Date: March 16, 1989

/* convert.h
****************************************************************************
*                                Thesis work                               *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.        *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
****************************************************************************
*/

/*
      Define the basic types and defines needed for conversion between a
      temporal Bayes network and a Markov Random Process transition table.
*/

/**********************************/
/* some basic types and defines */
/********************************/

/* null pointer */
#ifndef NULL
#define NULL ((char *) 0)
#endif

/* a boolean type */
typedef short BOOLEAN;

/* a false boolean value */
#define FALSE 0

/* a true boolean value */
#define TRUE  1

/* error code type */
typedef enum {
                no_errors,              /* no errors */
                bad_initial_state,      /* improper input or bad parameter
*/
                close_error,            /* error closing file */
                end_of_file,            /* end of file */
                malloc_error,           /* a memory allocation error */
                open_error,             /* an opening file error */
                read_error,             /* a disk read error */
                seek_error,             /* a seek file error */
                write_error             /* a disk write error */
             }
             ERROR_CODE;

/************************************************************/
/* some basic defines specific to converting from tBn to MRP */
/**********************************************************/

/* maximum number of characters for a variable name */
```

```
#define MAX_VAR_NAME_LEN 16

/* maximum number of variables in a system */
#define MAX_N_VARIABLES 7

/* the possible values for the pos_or_neg_state vector entries */
#define DOESNT_MATTER   -1
#define MUST_BE_NEG      0
#define MUST_BE_POS      1
```

File: c:\CONVERT\draw_tbn.h   Creation Date: March 16, 1989

```
/* draw_tbn.h
******************************************************************************
*                              Thesis work                                    *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.          *
*       This is an unpublished work fully protected by the copyright laws   *
*       and is considered a trade secret by the copyright owner.             *
******************************************************************************
*/

/*
      Define the types and defines needed for drawing the temporal Bayes
      network on a high resolution PC screen.
*/

/* the number of columns in the temporal Bayes network */
#define N_COLUMNS 4

/* the offset from the edge of the area allocated */
#define COL_OFFSET 20

/* the space occupied by the temporal Bayes network (in the x direction) */
#define NETWORK_WIDTH 340

/* the starting column for the temporal Bayes network */
#define NETWORK_COL_START 200

/* the space occupied by the temporal Bayes network (in the y direction) */
#define NETWORK_HEIGHT 250

/* the starting row for the temporal Bayes network */
#define NETWORK_ROW_START 50
```

File: c:\CONVERT\equiv_cl.h   Creation Date: March 16, 1989

```
/* equiv_cl.h
***************************************************************************
*                               Thesis work                               *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
***************************************************************************
*/

/*
      Define the types and defines needed for the equivalence class
      structure used in the conversion between a temporal Bayes network
      and a Markov Random Process transition table.
*/

typedef struct state_info
{
   short initial_state;
   short pos_or_neg_state[MAX_N_VARIABLES];
   struct state_info *previous;
   struct state_info *next;
} STATE_INFO;

typedef struct equivalence_class
{
   float probability;
   short n_states;
   STATE_INFO *state_info_list_p;
} EQUIVALENCE_CLASS;

typedef struct eq_class_info
{
   short n_equiv_classes;
   EQUIVALENCE_CLASS *equiv_class;
} EQ_CLASS_INFO;
```

File: c:\CONVERT\tbn_info.h   Creation Date: March 16, 1989

```
/* tbn_info.h
******************************************************************************
*                              Thesis work                                   *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.          *
*       This is an unpublished work fully protected by the copyright laws    *
*       and is considered a trade secret by the copyright owner.             *
******************************************************************************
*/

/*
      Define the types and defines needed for the temporal Bayes network
      variable information structure used in the conversion between a
      temporal Bayes network and a Markov Random Process transition table.
*/

typedef struct cond_prob_info
{
   short pos_or_neg_state[MAX_N_VARIABLES];
   float probability;
} COND_PROB_INFO;

typedef struct tbn_var_info
{
   short n_dependent;
   short dependent_vars[MAX_N_VARIABLES];
   short n_combinations;
   COND_PROB_INFO *conditional_probs;
} TBN_VAR_INFO;
```

Appendix B

Source Code

File: c:\CONVERT\main.c Creation Date: March 16, 1989

```
/* main.c
*****************************************************************************
*                               Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.            *
*****************************************************************************
*/

/*===========================================================================*/
/*
   Functionality:
      This file contains a program which will start with a temporal Bayes
      network and convert it into a Markov Random Process transition table.
      It will then convert the MRP transition table back into a temporal
      Bayes network.
*/
/*===========================================================================*/

#include <stdio.h>
#include <string.h>

#include "convert.h"
#include "equiv_cl.h"
#include "tbn_info.h"

/*===========================================================================*/
/* A convenient macro for detecting an error, printing a message, and
 * exiting the program.
 */

#define TESTABORT(routine)\
if (stat != (ERROR_CODE) no_errors)\
{\
   fprintf(stderr,"(appl) Error in function: '%s' error code: %hd!\n",\
           routine, (short) stat);\
   exit(0);\
}

/*===========================================================================*/

void main()
{
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
   float **mrp_table;
   short n_states;
   short n_variables;
   BOOLEAN request_stop;
   ERROR_CODE stat;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   float **var_probs;
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
```

```
   ERROR_CODE calculate_cond_probs();
   ERROR_CODE calculate_MRP_table();
   ERROR_CODE calculate_prior_probs();
   ERROR_CODE confirm_and_continue();
   ERROR_CODE determine_dependencies();
   ERROR_CODE draw_temporal_Bayes_network();
   void exit();
   ERROR_CODE free_equiv_classes();
   ERROR_CODE free_MRP_table();
   void free_summed_probs();
   ERROR_CODE free_tbn_info();
   ERROR_CODE get_conditional_probabilities();
   ERROR_CODE get_dependent_variables();
   ERROR_CODE get_variables_info();
   ERROR_CODE group_equiv_classes();
   ERROR_CODE sum_probs();

   /***********************************************/
   /* get number and names of variables involved */
   /*********************************************/
   stat = get_variables_info(&n_variables, variable_name);
   TESTABORT("get_variables_info");

   /**********************************************/
   /* get information about dependent variables */
   /********************************************/
   stat = get_dependent_variables(n_variables, variable_name, tbn_info);
   TESTABORT("get_dependent_variables");

   /********************************/
   /* draw temporal Bayes network */
   /******************************/
   stat = draw_temporal_Bayes_network(n_variables, variable_name, tbn_info)
;
   TESTABORT("draw_temporal_Bayes_network");

   /**********************************/
   /* get conditional probabilities */
   /********************************/
   stat = get_conditional_probabilities(n_variables, variable_name, tbn_inf
o);
   TESTABORT("get_conditional_probabilities");

   /************************************/
   /* calculate MRP transition matrix */
   /**********************************/
   stat = calculate_MRP_table(n_variables, tbn_info, &n_states, &mrp_table)
;
   TESTABORT("calculate_MRP_table");

   /***********************************************/
   /* free temporal Bayes network info structure */
   /*********************************************/
   stat = free_tbn_info(n_variables, tbn_info);
```

```
    TESTABORT("free_tbn_info");

    /**********************************************************************/
    /* confirm MRP table and conversion back to temporal Bayes network */
    /*********************************************************************/
    stat = confirm_and_continue(&request_stop);
    TESTABORT("confirm_and_continue");

    if (request_stop)
    {
       stat = free_MRP_table(n_states, &mrp_table);
       TESTABORT("free_MRP_table");
       exit(0);
    }

    /* now go from the MRP table back to the temporal Bayes network */

    /****************************************/
    /* calculate the prior probabilities */
    /*************************************/
    stat = calculate_prior_probs(n_states, mrp_table);
    TESTABORT("calculate_prior_probs");

    /*******************************************/
    /* sum the probabilities for each variable */
    /******************************************/
    stat = sum_probs(n_variables, n_states, mrp_table, &var_probs);
    TESTABORT("sum_probs");

    /************************************************/
    /* free Markov Random Process transition table */
    /***********************************************/
    stat = free_MRP_table(n_states, &mrp_table);
    TESTABORT("free_MRP_table");

    /*************************************************/
    /* group probabilities into equivalence classes */
    /***********************************************/
    stat = group_equiv_classes(n_variables, n_states, var_probs, equiv_class
es);
    TESTABORT("group_equiv_classes");

    /**********************************************/
    /* free summed probabilities for each variable */
    /*********************************************/
    free_summed_probs(n_variables, &var_probs);

    /*********************************/
    /* determine dependent variables */
    /********************************/
    stat = determine_dependencies(n_variables, variable_name, n_states,
              equiv_classes, tbn_info);
    TESTABORT("determine_dependencies");

    /*****************************************/
```

```
    /* calculate conditional probabilities */
    /*****************************************/
    stat = calculate_cond_probs(n_variables, variable_name, n_states,
            equiv_classes, tbn_info);
    TESTABORT("calculate_cond_probs");

    /*****************************************/
    /* free equivalence classes structure */
    /****************************************/
    stat = free_equiv_classes(n_variables, equiv_classes);
    TESTABORT("free_equiv_classes");

    /**********************************/
    /* draw temporal Bayes network */
    /*******************************/
    stat = draw_temporal_Bayes_network(n_variables, variable_name, tbn_info)
;
    TESTABORT("draw_temporal_Bayes_network2");

    /*************************************************/
    /* free temporal Bayes network info structure */
    /**********************************************/
    stat = free_tbn_info(n_variables, tbn_info);
    TESTABORT("free_tbn_info2");
}
```

File: c:\CONVERT\add_equi.c   Creation Date: March 14, 1989

```
/* add_equi.c
****************************************************************************
*                              Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.        *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
****************************************************************************
*/

/*========================================================================*/
/*
   Functionality:
      This file contains a routine which will add an equivalence class to
      the specified variable's list of equivalence classes.
*/
/*========================================================================*/

#include <stdio.h>

#include "convert.h"
#include "equiv_cl.h"

/*========================================================================*/

ERROR_CODE add_equiv_class(var_i, input_probability, equiv_classes)
   short var_i;
   float input_probability;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
{
   char *malloc();
   char *realloc();

   /***************************************************/
   /* add an equivalence class to this variable's list */
   /***************************************************/

   if (equiv_classes[var_i].n_equiv_classes == 0)
   {
      /* malloc the first one */
      if ((equiv_classes[var_i].equiv_class = (EQUIVALENCE_CLASS *)
            malloc(sizeof(EQUIVALENCE_CLASS))) == (EQUIVALENCE_CLASS *) NUL
L)
      {
         fprintf(stderr, "Ran out of space - Aborting program.\n");
         return((ERROR_CODE) malloc_error);
      }
   }
   else
   {
      /* realloc additional ones */
      if ((equiv_classes[var_i].equiv_class = (EQUIVALENCE_CLASS *)
            realloc(equiv_classes[var_i].equiv_class,
```

```
                (equiv_classes[var_i].n_equiv_classes + 1) *
                sizeof(EQUIVALENCE_CLASS))) == (EQUIVALENCE_CLASS *) NULL)
        {
            fprintf(stderr, "Ran out of space - Aborting program.\n");
            return((ERROR_CODE) malloc_error);
        }
    }
    equiv_classes[var_i].equiv_class[equiv_classes[var_i].n_equiv_classes].
            probability = (float) input_probability;

    equiv_classes[var_i].n_equiv_classes++;


    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\calc_con.c   Creation Date: March 14, 1989

```
/* calc_con.c
**************************************************************************
*                                Thesis work                             *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.      *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.         *
**************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will calculate the conditional
      probabilities for the temporal Bayes network.
*/
/*=========================================================================*/

#include <stdio.h>

#include "convert.h"
#include "equiv_cl.h"
#include "tbn_info.h"

/*=========================================================================*/

ERROR_CODE calculate_cond_probs(n_variables, variable_name, n_states,
         equiv_classes, tbn_info)
   short n_variables;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   short n_states;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short and_value;
   short bit_map;
   short combo_i;
   short d_var_i;
   short n_combinations;
   short pos_or_neg;
   short var_i;

   void init_pos_or_neg_states();
   void locate_pos_or_neg_states();
   char *malloc();

   /****************************************/
   /* calculate conditional probabilities */
   /***************************************/
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      /* now must determine how many combinations there are and what each
       * one's probability is */
```

```
      n_combinations = 1;

      /* determine the number of combinations of dependent variables */
      for (d_var_i = 0; d_var_i < tbn_info[var_i].n_dependent; d_var_i++)
      {
         n_combinations = n_combinations * 2;
      }

      printf("\nNow locating %hd (conditional) probabilities for variable %
hd (%s)\n",
            n_combinations, var_i + 1, variable_name[var_i]);

      tbn_info[var_i].n_combinations = n_combinations;
      /* allocate space for the conditional probabilities */
      if ((tbn_info[var_i].conditional_probs = (COND_PROB_INFO *)
            malloc(n_combinations * sizeof(COND_PROB_INFO))) ==
                  (COND_PROB_INFO *) NULL)
      {
         fprintf(stderr, "Ran out of space - Aborting program.\n");
         return((ERROR_CODE) malloc_error);
      }

      /* initialize the pos_or_neg_state vectors to indicate doesn't matter
 */
      for (combo_i = 0; combo_i < n_combinations; combo_i++)
      {
         init_pos_or_neg_states(tbn_info[var_i].conditional_probs[combo_i].
pos_or_neg_state);
      }

      for (combo_i = 0; combo_i < n_combinations; combo_i++)
      {
         bit_map = (n_combinations - combo_i) - 1;

         /* determine whether the next dependent variable is negated or not
 */
         for (and_value = n_combinations / 2, d_var_i = 0;
               and_value > 0;
                  and_value = and_value / 2, d_var_i++)
         {
            pos_or_neg = bit_map & and_value;
            if (pos_or_neg > 0)
            {
               tbn_info[var_i].conditional_probs[combo_i].
                  pos_or_neg_state[tbn_info[var_i].dependent_vars[d_var_i]]
                        = MUST_BE_POS;
            }
            else
            {
               tbn_info[var_i].conditional_probs[combo_i].
                  pos_or_neg_state[tbn_info[var_i].dependent_vars[d_var_i]]
                        = MUST_BE_NEG;
            }
         }
```

```
        /* locate this pos_or_neg_state vector among the equivalence class
es */
        locate_pos_or_neg_states(variable_name, var_i, combo_i, equiv_clas
ses,
                    tbn_info);
      }
    }

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\calc_mrp.c   Creation Date: March 14, 1989

```
/* calc_mrp.c
*****************************************************************************
*                               Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.            *
*****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will start with a temporal Bayes
      network and calculate the values in a corresponding Markov random
      process transition table.
*/
/*=========================================================================*/

#include <stdio.h>

#include "convert.h"
#include "tbn_info.h"

/*=========================================================================*/

ERROR_CODE calculate_MRP_table(n_variables, tbn_info, n_states_p, mrp_table
_p)
   short n_variables;
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
   short *n_states_p;
   float ***mrp_table_p;
{
   short n_states;
   float **mrp_table;
   float probabilities[MAX_N_VARIABLES];
   short row_state_i;
   short var_i;

   void calculate_probability();
   char *malloc();
   void select_probabilities();

   /************************************/
   /* calculate MRP transition matrix */
   /************************************/
   n_states = 1;
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      n_states = n_states * 2;
   }
   printf("\n\nThe MRP transition table has %hd possible states.\n\n", n_st
ates);
```

```
    /* record the number of states in the MRP transition table */
    *n_states_p = n_states;

    /* allocate space for the n_states possible rows */
    if ((mrp_table = (float **) malloc(n_states * sizeof(float *))) ==
                     (float **) NULL)
    {
       fprintf(stderr, "Ran out of space - Aborting program.\n");
       return((ERROR_CODE) malloc_error);
    }

    /* record the pointer to the MRP transition table */
    *mrp_table_p = mrp_table;

    /* for each row in the MRP transition table */
    for (row_state_i = 0; row_state_i < n_states; row_state_i++)
    {
       printf("\nRow %hd of the MRP transition table is for input state %hd\
n",
               row_state_i + 1, row_state_i);

       /* select the appropriate set of probabilities for this initial state
 */
       select_probabilities(n_variables, n_states, tbn_info, row_state_i,
                  probabilities);

       /* allocate space to hold the entries in one row of the transition ta
ble */
       if ((mrp_table[row_state_i] = (float *)
                  malloc(n_states * sizeof(float))) == (float *) NULL)
       {
          fprintf(stderr, "Ran out of space - Aborting program.\n");
          return((ERROR_CODE) malloc_error);
       }

       /* calculate the probability of each of the output states */
       calculate_probability(n_variables, n_states, probabilities, row_state
_i,
                  mrp_table);
    }

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\calc_pri.c Creation Date: March 15, 1989

```
/* calc_pri.c
*****************************************************************************
*                                Thesis work                                *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.            *
*****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will calculate the prior
      probabilities for each state.
*/
/*=========================================================================*/

#include <stdio.h>

#include "convert.h"

/*=========================================================================*/

ERROR_CODE calculate_prior_probs(n_states, mrp_table)
   short n_states;
   float **mrp_table;
{
   short input_state_i;
   short output_state_i;
   float prior_prob;

   printf("\n\n");
   for (output_state_i = 0; output_state_i < n_states; output_state_i++)
   {
      /* initialize the probabilitiy to zero */
      prior_prob = (float) 0.0;

      /* sum the entries in each row */
      for (input_state_i = 0; input_state_i < n_states; input_state_i++)
      {
         prior_prob += mrp_table[input_state_i][output_state_i];
      }
      prior_prob = prior_prob / (float) n_states;
      printf("The prior probability for state %3hd is %f\n", output_state_i
,
              prior_prob);
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\calc_pro.c   Creation Date: March 14, 1989

```
/* calc_pro.c
******************************************************************************
*                              Thesis work                                   *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.          *
*       This is an unpublished work fully protected by the copyright laws   *
*       and is considered a trade secret by the copyright owner.             *
******************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will calculate the probability
      of each of the output states for a given input state.
*/
/*=========================================================================*/

#include <stdio.h>

#include "convert.h"

/*=========================================================================*/

void calculate_probability(n_variables, n_states, probabilities, row_state_
i,
         mrp_table)
   short n_variables;
   short n_states;
   float probabilities[MAX_N_VARIABLES];
   short row_state_i;
   float **mrp_table;
{
   short output_pos_or_neg_state[MAX_N_VARIABLES];
   short output_state_i;
   short var_i;

   void det_pos_or_neg_states();

   /* calculate the probability of each of the output states */
   for (output_state_i = 0; output_state_i < n_states; output_state_i++)
   {
      /* determine the pos_or_neg_states for the output state */
      det_pos_or_neg_states(output_state_i, n_states, output_pos_or_neg_sta
te);

      mrp_table[row_state_i][output_state_i] = (float) 1.0;

      for (var_i = 0; var_i < n_variables; var_i++)
      {
         if (output_pos_or_neg_state[var_i] == MUST_BE_POS)
         {
            mrp_table[row_state_i][output_state_i] =
```

```
                    mrp_table[row_state_i][output_state_i] *
                        probabilities[var_i];
            }
            else
            {
               mrp_table[row_state_i][output_state_i] =
                    mrp_table[row_state_i][output_state_i] *
                        (((float) 1.0) - probabilities[var_i]);
            }
        }
        printf("The table value for input state %3hd, output state %3hd is %f
\n",
            row_state_i, output_state_i,
            mrp_table[row_state_i][output_state_i]);
    }
}
```

```
File: c:\CONVERT\confirm_.c    Creation Date: March 16, 1989

/* confirm_.c
***************************************************************************
*                              Thesis work                                *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
***************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will confirm that the user is
      satisfied with the Markov Random Process transition table which was
      generated, and that the user desires to convert it back to a
      temporal Bayes network.
*/
/*=========================================================================*/

#include <stdio.h>
#include <string.h>

#include "convert.h"

/*=========================================================================*/

ERROR_CODE confirm_and_continue(request_stop_p)
   BOOLEAN *request_stop_p;
{
   char verify[2];

   /***********************************************************************/
   /* confirm MRP table and conversion back to temporal Bayes network */
   /***********************************************************************/
   printf("\nPlease confirm this Markov Random Process transition table and
\n");
   printf("the desire to convert it back to a temporal Bayes network\n");
   printf("by pressing y now (anything else will terminate processing)\n");
   scanf("%s", verify);

   if (strcmp(verify, "y") != 0)
   {
      *request_stop_p = TRUE;
      printf("Acknowledging desire to exit program.\n");
   }
   else
      *request_stop_p = FALSE;

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\det_depe.c   Creation Date: March 14, 1989

```
/* det_depe.c
***************************************************************************
*                               Thesis work                               *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
***************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will determine the dependent
      variables for each variable.
*/
/*=========================================================================*/

#include <stdio.h>

#include "convert.h"
#include "equiv_cl.h"
#include "tbn_info.h"

/*=========================================================================*/

ERROR_CODE determine_dependencies(n_variables, variable_name, n_states,
        equiv_classes, tbn_info)
   short n_variables;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   short n_states;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short class_i;
   short d_var_i;
   STATE_INFO *next_state_p;
   short pos_or_neg_state[MAX_N_VARIABLES];
   short var_i;
   short var_ii;

   void init_pos_or_neg_states();
   void reduce_equiv_class_states();

   /**********************************/
   /* determine dependent variables */
   /*********************************/
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      /* reduce number of states for each equiv class by combining if possi
ble */
      reduce_equiv_class_states(var_i, equiv_classes);
```

```
      /* now determine what these reduced lists tell us */
      /* initialize the pos_or_neg_state vectors to indicate doesn't matter
 */
      init_pos_or_neg_states(pos_or_neg_state);

      /* look at each equivalence class */
      for (class_i = 0; class_i < equiv_classes[var_i].n_equiv_classes; cla
ss_i++)
      {
         /* look at each state */
         for (next_state_p = equiv_classes[var_i].equiv_class[class_i].stat
e_info_list_p;
                  next_state_p != (STATE_INFO *) NULL;
                     next_state_p = next_state_p->next)
         {
            /* determine whether each variable matters for this state */
            for (var_ii = 0; var_ii < MAX_N_VARIABLES; var_ii++)
            {
               if (next_state_p->pos_or_neg_state[var_ii] != DOESNT_MATTER)
               {
                  pos_or_neg_state[var_ii] =
                        next_state_p->pos_or_neg_state[var_ii];
               }
            }
         }
      }
      tbn_info[var_i].n_dependent = 0;
      for (var_ii = 0; var_ii < MAX_N_VARIABLES; var_ii++)
      {
         if (pos_or_neg_state[var_ii] != DOESNT_MATTER)
         {
            tbn_info[var_i].dependent_vars[tbn_info[var_i].n_dependent] = v
ar_ii;
            tbn_info[var_i].n_dependent++;
         }
      }
      printf("Variable %d (known as %s) is dependent on %hd variables:\n",
                var_i + 1, variable_name[var_i], tbn_info[var_i].n_dependent
);
      for (d_var_i = 0; d_var_i < tbn_info[var_i].n_dependent; d_var_i++)
      {
         printf("     variable %d, known as %s\n",
                tbn_info[var_i].dependent_vars[d_var_i] + 1,
                variable_name[tbn_info[var_i].dependent_vars[d_var_i]]);
      }
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

```
File: c:\CONVERT\draw_tbn.c    Creation Date: March 13, 1989

/* draw_tbn.c
******************************************************************************
*                              Thesis work                                   *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.          *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.             *
******************************************************************************
*/

/*==========================================================================*/
/*
   Functionality:
      This file contains a routine which will draw a temporal Bayes
      network on a high resolution PC screen.
*/
/*==========================================================================*/

#include <stdio.h>
#include <string.h>

#include "convert.h"
#include "draw_tbn.h"
#include "tbn_info.h"

/*==========================================================================*/

ERROR_CODE draw_temporal_Bayes_network(n_variables, variable_name, tbn_info
)
   short n_variables;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short col_i;
   short col_pos[N_COLUMNS];
   short col_spacing;
   short d_var_i;
   short row_spacing;
   short var_i;
   short var_row[MAX_N_VARIABLES];
   char verify[2];

   void circle();
   void fhatsay();
   int fontinit();
   int fontld();
   int fontunld();
   int grline();
   int initgraf();
   void setega();

   /**********************************/
   /* draw temporal Bayes network */
```

```
   /******************************/
   setega();
   initgraf(16, 0, 1);   /* initialize EGA high resolution graphics */
   fontinit(0);
   fontld(0, "IBMROM");

   /* calculate column positions */
   col_spacing = (NETWORK_WIDTH - (2 * COL_OFFSET)) / (N_COLUMNS - 1);
   for (col_i = 0; col_i < N_COLUMNS; col_i++)
   {
      col_pos[col_i] = NETWORK_COL_START + (col_i * col_spacing) + COL_OFFS
ET;
   }
   fhatsay(0, "t-1", 15, col_pos[0], 20);
   fhatsay(0, "t", 15, col_pos[1], 20);
   fhatsay(0, "t+1", 15, col_pos[2], 20);
   fhatsay(0, "t+2", 15, col_pos[3], 20);

   /* calculate row positions */
   row_spacing = NETWORK_HEIGHT / (n_variables + 1);
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      var_row[var_i] = ((var_i + 1) * row_spacing) + NETWORK_ROW_START;
      fhatsay(0, variable_name[var_i], 15, 50, var_row[var_i]);
      grline(NETWORK_COL_START, var_row[var_i],
             NETWORK_COL_START + NETWORK_WIDTH, var_row[var_i], 3);
   }

   /* draw the nodes and remaining lines between them */
   for (col_i = 0; col_i < N_COLUMNS; col_i++)
   {
      grline(col_pos[col_i], NETWORK_ROW_START,
             col_pos[col_i], NETWORK_ROW_START + NETWORK_HEIGHT, 3);
      for (var_i = 0; var_i < n_variables; var_i++)
      {
         circle(col_pos[col_i], var_row[var_i], 5, 3, 1);
      }
   }

   /* draw the arrows between the dependent variables */
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      for (d_var_i = 0; d_var_i < tbn_info[var_i].n_dependent; d_var_i++)
      {
         for (col_i = 1; col_i < N_COLUMNS; col_i++)
         {
            grline(col_pos[col_i], var_row[var_i], col_pos[col_i - 1],
                   var_row[tbn_info[var_i].dependent_vars[d_var_i]], 15);
         }
      }
   }

   fhatsay(0, "Please confirm this drawing of the Temporal Bayes Network",
        15, 100, 330);
   fhatsay(0, "by pressing y now (anything else will terminate processing)"
```

```
,
          15, 100, 340);
   scanf("%s", verify);

   fontunld(0);
   initgraf(3, 0, 0);    /* reset text mode */

   if (strcmp(verify, "y") != 0)
   {
      printf("Acknowledging error.  Please start again.\n");
      return((ERROR_CODE) bad_initial_state);
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

```
File: c:\CONVERT\find_sta.c    Creation Date: March 14, 1989

/* find_sta.c
***************************************************************************
*                              Thesis work                                *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
***************************************************************************
*/

/*=======================================================================*/
/*
   Functionality:
      This file contains a routine which will determine the states for
      which each variable is positive.
*/
/*=======================================================================*/

#include <stdio.h>

#include "convert.h"

/*=======================================================================*/

ERROR_CODE find_states_to_sum(n_variables, n_states, use_states_p)
   short n_variables;
   short n_states;
   short ***use_states_p;
{
   short and_value;
   short next_state;
   short n_states_used;
   short pos_or_neg;
   short state_i;
   short **use_states;
   short var_i;

   char *malloc();

   /*****************************************/
   /* find states to sum for each variable */
   /****************************************/

   /* allocate space for the n_variables possible variables */
   if ((use_states = (short **) malloc(n_variables * sizeof(short *))) ==
                  (short **) NULL)
   {
      fprintf(stderr, "Ran out of space - Aborting program.\n");
      return((ERROR_CODE) malloc_error);
   }

   /* record the pointer to the use_states matrix */
   *use_states_p = use_states;
```

```
    and_value = n_states;
    n_states_used = n_states / 2;

    for (var_i = 0; var_i < n_variables; var_i++)
    {
       and_value = and_value / 2;

       /* allocate space to hold the states to use for one variable */
       if ((use_states[var_i] = (short *)
                malloc(n_states_used * sizeof(short))) == (short *) NULL)
       {
          fprintf(stderr, "Ran out of space - Aborting program.\n");
          return((ERROR_CODE) malloc_error);
       }

       next_state = 0;
       /* determine which entries in each row would be summed */
       for (state_i = 0; state_i < n_states; state_i++)
       {
          /* if the variable is positive in this state, include it in the su
m */
          pos_or_neg = state_i & and_value;
          if (pos_or_neg)
          {
             use_states[var_i][next_state] = state_i;
             next_state++;
          }
       }
    }

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\free_equ.c   Creation Date: March 13, 1989

```
/* free_equ.c
****************************************************************************
*                              Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.        *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
****************************************************************************
*/

/*========================================================================*/
/*
   Functionality:
      This file contains a routine which will free the equivalence classes
      structure.
*/
/*========================================================================*/

#include "convert.h"
#include "equiv_cl.h"

/*========================================================================*/

ERROR_CODE free_equiv_classes(n_variables, equiv_classes)
   short n_variables;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
{
   short class_i;
   STATE_INFO *current_state_p;
   STATE_INFO *next_state_p;
   short var_i;

   void free();

   /****************************************/
   /* free equivalence classes structure */
   /**************************************/

   for (var_i = 0; var_i < n_variables; var_i++)
   {
      for (class_i = 0; class_i < equiv_classes[var_i].n_equiv_classes; cla
ss_i++)
      {
         /* free each of the STATE_INFO structures in the linked list */
         for (current_state_p = equiv_classes[var_i].equiv_class[class_i].s
tate_info_list_p;
                 current_state_p != (STATE_INFO *) NULL;
                    current_state_p = next_state_p)
         {
            next_state_p = current_state_p->next;
            free(current_state_p);
         }
         equiv_classes[var_i].equiv_class[class_i].state_info_list_p =
```

```
                (STATE_INFO *) NULL;
            equiv_classes[var_i].equiv_class[class_i].n_states = 0;
        }
        /* free the array of equivalence class structures */
        free(equiv_classes[var_i].equiv_class);
        equiv_classes[var_i].equiv_class = (EQUIVALENCE_CLASS *) NULL;
        equiv_classes[var_i].n_equiv_classes = 0;
    }

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\free_mrp.c   Creation Date: March 16, 1989

```
/* free_mrp.c
*****************************************************************************
*                              Thesis work                                  *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.            *
*****************************************************************************
*/

/*===========================================================================*/
/*
   Functionality:
      This file contains a routine which will free the Markov Random
      Process transition table, a matrix of probabilities of an output
      state, given an input state.
*/
/*===========================================================================*/

#include "convert.h"

/*===========================================================================*/

ERROR_CODE free_MRP_table(n_states, mrp_table_p)
   short n_states;
   float ***mrp_table_p;
{
   float **mrp_table;
   short state_i;

   void free();

   /************************************************/
   /* free Markov Random Process transition table */
   /***********************************************/
   mrp_table = *mrp_table_p;

   for (state_i = 0; state_i < n_states; state_i++)
   {
      /* free space allocated for output probabilities for one input state
*/
      free(mrp_table[state_i]);
   }

   /* free space allocated for the n_states initial states */
   free(mrp_table);

   /* reset the pointer to the mrp_table matrix */
   *mrp_table_p = (float ***) NULL;

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\free_sta.c Creation Date: March 13, 1989

```
/* free_sta.c
***************************************************************************
*                            Thesis work                                  *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
***************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will free the array which was
      allocated to hold the list of states for which each variable is
      positive.
*/
/*=========================================================================*/

#include "convert.h"

/*=========================================================================*/

ERROR_CODE free_states_to_sum(n_variables, use_states_p)
   short n_variables;
   short ***use_states_p;
{
   short **use_states;
   short var_i;

   void free();

   /***************************************************/
   /* free array of states to sum for each variable */
   /*************************************************/

   use_states = *use_states_p;

   for (var_i = 0; var_i < n_variables; var_i++)
   {
      /* free space allocated to hold the states to use for one variable */
      free(use_states[var_i]);
   }

   /* free space allocated for the n_variables possible variables */
   free(use_states);

   /* reset the pointer to the use_states matrix */
   *use_states_p = (short ***) NULL;

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\free_sum.c   Creation Date: March 13, 1989

```
/* free_sum.c
****************************************************************************
*                              Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.        *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will free the matrix which was
      allocated to hold the probabilities for each initial state for each
      variable.
*/
/*=========================================================================*/

#include "convert.h"

/*=========================================================================*/

void free_summed_probs(n_variables, var_probs_p)
   short n_variables;
   float ***var_probs_p;
{
   float **var_probs;
   short var_i;

   void free();

   /*************************************************/
   /* free summed probabilities for each variable */
   /***********************************************/

   var_probs = *var_probs_p;

   for (var_i = 0; var_i < n_variables; var_i++)
   {
      /* free space allocated to hold the probabilities for one variable */
      free(var_probs[var_i]);
   }

   /* free space allocated for the n_variables possible variables */
   free(var_probs);

   /* reset the pointer to the var_probs matrix */
   *var_probs_p = (float ***) NULL;
}
```

File: c:\CONVERT\free_tbn.c   Creation Date: March 13, 1989

```
/* free_tbn.c
****************************************************************************
*                              Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will free the temporal Bayes
      network information structure.
*/
/*=========================================================================*/

#include "convert.h"
#include "tbn_info.h"

/*=========================================================================*/

ERROR_CODE free_tbn_info(n_variables, tbn_info)
   short n_variables;
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short var_i;

   void free();

   /***********************************************/
   /* free temporal Bayes network info structure */
   /**********************************************/

   for (var_i = 0; var_i < n_variables; var_i++)
   {
      tbn_info[var_i].n_dependent = 0;
      tbn_info[var_i].n_combinations = 0;
      free(tbn_info[var_i].conditional_probs);
      tbn_info[var_i].conditional_probs = (COND_PROB_INFO *) NULL;
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\get_cond.c   Creation Date: March 16, 1989

```
/* get_cond.c
******************************************************************************
*                                   Thesis work                              *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.          *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.             *
******************************************************************************
*/

/*==========================================================================*/
/*
   Functionality:
      This file contains a routine which will obtain the conditional
      probabilities for the temporal Bayes network.
*/
/*==========================================================================*/

#include <stdio.h>
#include <string.h>

#include "convert.h"
#include "tbn_info.h"

/*==========================================================================*/

ERROR_CODE get_conditional_probabilities(n_variables, variable_name, tbn_in
fo)
   short n_variables;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short and_value;
   short bit_map;
   short combo_i;
   short d_var_i;
   short n_combinations;
   short pos_or_neg;
   short var_i;

   void init_pos_or_neg_states();
   char *malloc();

   /*********************************/
   /* get conditional probabilities */
   /*********************************/
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      n_combinations = 1;

      /* determine the number of combinations of dependent variables */
      for (d_var_i = 0; d_var_i < tbn_info[var_i].n_dependent; d_var_i++)
      {
```

```
            n_combinations = n_combinations * 2;
        }

        printf("\nNow accepting %hd (conditional) probabilities for variable
%hd (%s)\n",
                n_combinations, var_i + 1, variable_name[var_i]);

        tbn_info[var_i].n_combinations = n_combinations;
        /* allocate space for the conditional probabilities */
        if ((tbn_info[var_i].conditional_probs = (COND_PROB_INFO *)
                malloc(n_combinations * sizeof(COND_PROB_INFO))) ==
                    (COND_PROB_INFO *) NULL)
        {
            fprintf(stderr, "Ran out of space - Aborting program.\n");
            return((ERROR_CODE) malloc_error);
        }

        /* initialize the pos_or_neg_state vectors to indicate doesn't matter
 */
        for (combo_i = 0; combo_i < n_combinations; combo_i++)
        {
            init_pos_or_neg_states(tbn_info[var_i].conditional_probs[combo_i].
pos_or_neg_state);
        }

        for (combo_i = 0; combo_i < n_combinations; combo_i++)
        {
            bit_map = (n_combinations - combo_i) - 1;

            printf("Please enter the probability P\(%s at t", variable_name[va
r_i]);

            if (n_combinations > 1)
            {
                printf(" \¦ ");
            }

            /* determine whether the next dependent variable is negated or not
 */
            for (and_value = n_combinations / 2, d_var_i = 0;
                    and_value > 0;
                        and_value = and_value / 2, d_var_i++)
            {
                pos_or_neg = bit_map & and_value;
                if (pos_or_neg > 0)
                {
                    printf("%s", variable_name[tbn_info[var_i].dependent_vars[d_
var_i]]);
                    tbn_info[var_i].conditional_probs[combo_i].
                        pos_or_neg_state[tbn_info[var_i].dependent_vars[d_var_
i]]
                        = MUST_BE_POS;
                }
                else
                {
```

```
                printf("NOT %s", variable_name[tbn_info[var_i].dependent_var
s[d_var_i]]);
                tbn_info[var_i].conditional_probs[combo_i].
                      pos_or_neg_state[tbn_info[var_i].dependent_vars[d_var_
i]]
                      = MUST_BE_NEG;
             }
             if (d_var_i < (tbn_info[var_i].n_dependent - 1))
             {
                printf(" AND ");
             }
             else
             {
                printf(" at t-1");
             }
          }
          printf ("\)\n");
          scanf("%f", &(tbn_info[var_i].conditional_probs[combo_i].probabili
ty));

          if ((tbn_info[var_i].conditional_probs[combo_i].probability < (flo
at) 0) ||
                (tbn_info[var_i].conditional_probs[combo_i].probability > (f
loat) 1))
          {
             fprintf(stderr, "The probability was out of acceptable range.
Please start again.\n");
             return((ERROR_CODE) bad_initial_state);
          }

          printf("The probability stored was %f\n",
                tbn_info[var_i].conditional_probs[combo_i].probability);
       }
    }

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\get_depe.c   Creation Date: March 16, 1989

```
/* get_depe.c
******************************************************************************
*                               Thesis work                                  *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.          *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.             *
******************************************************************************
*/

/*==========================================================================*/
/*
   Functionality:
      This file contains a routine which will obtain information about the
      dependent variables in a temporal Bayes network.
*/
/*==========================================================================*/

#include <stdio.h>
#include <string.h>

#include "convert.h"
#include "tbn_info.h"

/*==========================================================================*/

ERROR_CODE get_dependent_variables(n_variables, variable_name, tbn_info)
   short n_variables;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short d_var_i;
   short var_i;
   short var_n;

   /**********************************************/
   /* get information about dependent variables */
   /*********************************************/
   printf("\n");
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      printf("How many variables is variable %d (known as %s) dependent on?
\n",
                  var_i + 1, variable_name[var_i]);
      scanf("%d", &tbn_info[var_i].n_dependent);

      if ((tbn_info[var_i].n_dependent < 0) ||
                  (tbn_info[var_i].n_dependent > n_variables))
      {
         fprintf(stderr, "The number was out of acceptable range.  Please s
tart again.\n");
         return((ERROR_CODE) bad_initial_state);
      }
```

```
      for (d_var_i = 0; d_var_i < tbn_info[var_i].n_dependent; d_var_i++)
      {
         printf("Please enter the number of dependent variable number %hd.\
n",
                 d_var_i + 1);
         scanf("%hd", &var_n);

         if ((var_n < 1) || (var_n > n_variables))
         {
            fprintf(stderr, "The variable number was out of acceptable rang
e.  Please start again.\n");
            return((ERROR_CODE) bad_initial_state);
         }
         tbn_info[var_i].dependent_vars[d_var_i] = var_n - 1;
      }
      printf("Variable %d (known as %s) is dependent on %hd variables:\n",
              var_i + 1, variable_name[var_i], tbn_info[var_i].n_dependent
);
      for (d_var_i = 0; d_var_i < tbn_info[var_i].n_dependent; d_var_i++)
      {
         printf("     variable %d, known as %s\n",
              tbn_info[var_i].dependent_vars[d_var_i] + 1,
              variable_name[tbn_info[var_i].dependent_vars[d_var_i]]);
      }
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\get_vars.c   Creation Date: March 16, 1989

```
/* get_vars.c
*************************************************************************
*                               Thesis work                             *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.     *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.        *
*************************************************************************
*/

/*=====================================================================*/
/*
   Functionality:
      This file contains a routine which will obtain information about the
      number of variables and their names.
*/
/*=====================================================================*/

#include <stdio.h>
#include <string.h>

#include "convert.h"

/*=====================================================================*/

ERROR_CODE get_variables_info(n_variables_p, variable_name)
   short *n_variables_p;
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
{
   short n_variables;
   short var_i;

   /*************************************/
   /* get number of variables involved */
   /*************************************/
   printf("How many variables does the temporal Bayes network involve?\n");
   printf("(The maximum that this program can handle is %d)\n", MAX_N_VARIA
BLES);
   scanf("%d", &n_variables);

   if ((n_variables < 0) || (n_variables > MAX_N_VARIABLES))
   {
      fprintf(stderr, "Program cannot handle %d variables.  Please start ag
ain.\n",
              n_variables);
      return((ERROR_CODE) bad_initial_state);
   }
   else
   {
      *n_variables_p = n_variables;
      printf("This temporal Bayes network involves %hd variables.\n", n_var
iables);
   }
```

```
   /**********************************/
   /* get a name for each variable */
   /*******************************/
   for (var_i = 0; var_i < n_variables; var_i++)
   {
      printf("Please enter a name of no more than %hd characters for variab
le %hd\n",
               MAX_VAR_NAME_LEN - 1, var_i + 1);
      scanf("%s", variable_name[var_i]);
      printf("The name stored for variable %hd is %s.\n", var_i + 1,
               variable_name[var_i]);
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\group_eq.c   Creation Date: March 14, 1989

```
/* group_eq.c
***************************************************************************
*                              Thesis work                                *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
***************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will partition the variables'
      probabilities into equivalence classes.
*/
/*=========================================================================*/

#include <stdio.h>
#include <math.h>

#include "convert.h"
#include "equiv_cl.h"

/*=========================================================================*/

ERROR_CODE group_equiv_classes(n_variables, n_states, var_probs, equiv_clas
ses)
   short n_variables;
   short n_states;
   float **var_probs;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
{
   short class_i;
   BOOLEAN found_prob;
   STATE_INFO *new_state_p;
   ERROR_CODE stat;
   short state_i;
   short var_i;

   ERROR_CODE add_equiv_class();
   ERROR_CODE locate_equiv_class();
   ERROR_CODE init_state_entry();

   /*************************************************/
   /* group probabilities into equivalence classes */
   /************************************************/

   /* initialize the equivalence class structures */
   for (var_i = 0; var_i < MAX_N_VARIABLES; var_i++)
   {
      equiv_classes[var_i].n_equiv_classes = 0;
      equiv_classes[var_i].equiv_class = (EQUIVALENCE_CLASS *) NULL;
```

```
    }

    /* divide each variable's probabilities into classes */
    for (var_i = 0; var_i < n_variables; var_i++)
    {
       /* decide upon an equivalence class for each input state's probabilit
y */
       for (state_i = 0; state_i < n_states; state_i++)
       {
          found_prob = FALSE;

          /* add this state to the equiv class with the appropriate probabil
ity */
          stat = locate_equiv_class(n_states, var_i, state_i, equiv_classes,
                   var_probs[var_i][state_i], &found_prob);
          if (stat != (ERROR_CODE) no_errors)
             return(stat);

          /* if could not find this probability among the existing equiv cla
sses */
          if (!found_prob)
          {
             /* add an equivalence class to this variable's list */
             stat = add_equiv_class(var_i, var_probs[var_i][state_i], equiv_
classes);
             if (stat != (ERROR_CODE) no_errors)
                return(stat);

             /* get local copy of class index */
             class_i = equiv_classes[var_i].n_equiv_classes - 1;

             /* allocate and initialize a new state entry */
             stat = init_state_entry(n_states, state_i, &new_state_p);
             if (stat != (ERROR_CODE) no_errors)
                return(stat);

             /* add it to the beginning of the linked list */
             new_state_p->previous = (STATE_INFO *) NULL;
             equiv_classes[var_i].equiv_class[class_i].state_info_list_p = n
ew_state_p;

             equiv_classes[var_i].equiv_class[class_i].n_states = 1;
          }
       }
    }

    /* just a printf to see how many equivalence classes there are */
    printf("\n\n");
    for (var_i = 0; var_i < n_variables; var_i++)
    {
       printf("variable %hd has %hd equivalence classes.\n", var_i + 1,
          equiv_classes[var_i].n_equiv_classes);
       for (class_i = 0; class_i < equiv_classes[var_i].n_equiv_classes; cla
ss_i++)
       {
```

```
            printf("The probability is %f for %hd states.\n",
               equiv_classes[var_i].equiv_class[class_i].probability,
               equiv_classes[var_i].equiv_class[class_i].n_states);
        }
    }
    printf("\n\n");

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\init_sta.c Creation Date: March 14, 1989

```
/* init_sta.c
*****************************************************************************
*                                 Thesis work                               *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws  *
*       and is considered a trade secret by the copyright owner.            *
*****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will allocate a new state entry
      and initialize it.
*/
/*=========================================================================*/

#include <stdio.h>

#include "convert.h"
#include "equiv_cl.h"

/*=========================================================================*/

ERROR_CODE init_state_entry(n_states, state_i, new_state_p_p)
   short n_states;
   short state_i;
   STATE_INFO **new_state_p_p;
{
   STATE_INFO *new_state_p;

   void det_pos_or_neg_states();
   char *malloc();

   /**********************************************/
   /* allocate and initialize a new state entry */
   /*********************************************/

   /* allocate space for this state */
   if ((new_state_p = (STATE_INFO *) malloc(sizeof(STATE_INFO)))
         == (STATE_INFO *) NULL)
   {
      fprintf(stderr, "Ran out of space - Aborting program.\n");
      return((ERROR_CODE) malloc_error);
   }

   /* record the pointer to the new state entry */
   *new_state_p_p = new_state_p;

   new_state_p->initial_state = state_i;

   /* determine the pos_or_neg_states for this state */
```

```
    det_pos_or_neg_states(state_i, n_states, new_state_p->pos_or_neg_state);

    new_state_p->next = (STATE_INFO *) NULL;

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\locate_e.c   Creation Date: March 13, 1989

```
/* locate_e.c
****************************************************************************
*                                Thesis work                               *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.        *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
****************************************************************************
*/

/*==========================================================================*/
/*
   Functionality:
      This file contains a routine which will locate the equivalence class
      with the desired probability and add the specified state to it.
*/
/*==========================================================================*/

#include <math.h>

#include "convert.h"
#include "equiv_cl.h"

/*==========================================================================*/

ERROR_CODE locate_equiv_class(n_states, var_i, state_i, equiv_classes,
         input_probability, found_prob_p)
   short n_states;
   short var_i;
   short state_i;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
   float input_probability;
   BOOLEAN *found_prob_p;
{
   short class_i;
   STATE_INFO *new_state_p;
   STATE_INFO *next_state_p;
   ERROR_CODE stat;

   ERROR_CODE init_state_entry();

   /************************************************************************
/
   /* add this state to the equiv class with the appropriate probability */
   /***********************************************************************/

   /* look at each equivalence class to see if it's the right one */
   for (class_i = 0; class_i < equiv_classes[var_i].n_equiv_classes; class_
i++)
   {
      /* if the probability is within a small tolerance */
      if ((float)(fabs((double) (input_probability -
           equiv_classes[var_i].equiv_class[class_i].probability))) <
```

```
                    (float) 0.0001)
      {
         /* found the right probability - just add to list */
         *found_prob_p = TRUE;

         /* allocate and initialize a new state entry */
         stat = init_state_entry(n_states, state_i, &new_state_p);
         if (stat != (ERROR_CODE) no_errors)
            return(stat);

         /* add it to the end of the linked list */
         for (next_state_p = equiv_classes[var_i].equiv_class[class_i].stat
e_info_list_p;
               next_state_p->next != (STATE_INFO *) NULL;
                  next_state_p = next_state_p->next)
         {
            /* just walk down the list (at the end of this for loop,
             * next_state_p will point to the tail of the list)
             */
         }

         new_state_p->previous = next_state_p;
         next_state_p->next = new_state_p;

         equiv_classes[var_i].equiv_class[class_i].n_states++;

         /* break out of the for loop over equivalence classes */
         break;
      }
   }

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\locate_p.c Creation Date: March 14, 1989

```
/* locate_p.c
*************************************************************************
*                              Thesis work                              *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.     *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.        *
*************************************************************************
*/

/*=======================================================================*/
/*
   Functionality:
      This file contains a routine which will locate the indicated state
      among the equivalence classes, and use the probability for that
      equivalence class.
*/
/*=======================================================================*/

#include <stdio.h>

#include "convert.h"
#include "equiv_cl.h"
#include "tbn_info.h"

/*=======================================================================*/

void locate_pos_or_neg_states(variable_name, var_i, combo_i, equiv_classes,
        tbn_info)
   char variable_name[MAX_N_VARIABLES][MAX_VAR_NAME_LEN];
   short var_i;
   short combo_i;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
{
   short class_i;
   short d_var_i;
   BOOLEAN found_vector;
   STATE_INFO *next_state_p;
   short var_ii;

   /*********************************************************************/
   /* locate this pos_or_neg_state vector among the equivalence classes */
   /*********************************************************************/
   printf("Locating the probability P\(%s at t", variable_name[var_i]);

   if (tbn_info[var_i].n_combinations > 1)
   {
      printf(" \| ");
   }

   d_var_i = 0;
   for (var_ii = 0; var_ii < MAX_N_VARIABLES; var_ii++)
```

```
   {
      if (tbn_info[var_i].conditional_probs[combo_i].pos_or_neg_state[var_i
i]
              == MUST_BE_POS)
      {
         printf("%s", variable_name[var_ii]);
         d_var_i++;
         if (d_var_i < tbn_info[var_i].n_dependent)
         {
            printf(" AND ");
         }
      }
      if (tbn_info[var_i].conditional_probs[combo_i].pos_or_neg_state[var_i
i]
              == MUST_BE_NEG)
      {
         printf("NOT %s", variable_name[var_ii]);
         d_var_i++;
         if (d_var_i < tbn_info[var_i].n_dependent)
         {
            printf(" AND ");
         }
      }
   }
   if (tbn_info[var_i].n_dependent > 0)
   {
      printf(" at t-1");
   }
   printf ("\)\n");

   for (class_i = 0; class_i < equiv_classes[var_i].n_equiv_classes; class_
i++)
   {
      for (next_state_p = equiv_classes[var_i].equiv_class[class_i].state_i
nfo_list_p;
              next_state_p != (STATE_INFO *) NULL;
                 next_state_p = next_state_p->next)
      {
         found_vector = TRUE;
         for (var_ii = 0; var_ii < MAX_N_VARIABLES; var_ii++)
         {
            if ((next_state_p->pos_or_neg_state[var_ii] != DOESNT_MATTER)
                  && (next_state_p->pos_or_neg_state[var_ii] !=
                        tbn_info[var_i].conditional_probs[combo_i].
                           pos_or_neg_state[var_ii]))
            {
               /* this one does not match - break out of compare loop */
               found_vector = FALSE;
               break;
            }
         }
         /* if we have found the correct vector, use its probability */
         if (found_vector)
         {
            tbn_info[var_i].conditional_probs[combo_i].probability =
```

```
                  equiv_classes[var_i].equiv_class[class_i].probability;
               /* no need to look at the remaining states in this class */
               break;
            }
         }
         if (found_vector)
         {
            /* no need to look at remaining equivalence classes */
            break;
         }
      }
      printf("The probability stored was %f\n",
            tbn_info[var_i].conditional_probs[combo_i].probability);
   }
```

File: c:\CONVERT\posorneg.c Creation Date: March 13, 1989

```
/* posorneg.c
******************************************************************************
*                              Thesis work                                   *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
******************************************************************************
*/

/*==========================================================================*/
/*
   Functionality:
      This file contains routines which fill in appropriate value in a
      positive or negative state vector.
*/
/*==========================================================================*/

#include "convert.h"

/*==========================================================================*/

/****************************************************************************
 * initialize the pos_or_neg_states to indicate either state is OK
 */
void init_pos_or_neg_states(pos_or_neg_vector)
   short pos_or_neg_vector[MAX_N_VARIABLES];
{
   short var_i;

   for (var_i = 0; var_i < MAX_N_VARIABLES; var_i++)
   {
      pos_or_neg_vector[var_i] = DOESNT_MATTER;
   }
}

/****************************************************************************
 * determine the pos_or_neg_states based on the input number
 */
void det_pos_or_neg_states(input_number, max_number, pos_or_neg_vector)
   short input_number;
   short max_number;
   short pos_or_neg_vector[MAX_N_VARIABLES];
{
   short and_value;
   short pos_or_neg;
   short var_i;

   void init_pos_or_neg_states();

   /* initialize the "pos_or_neg_state" vector to indicate doesn't matter *
/
   init_pos_or_neg_states(pos_or_neg_vector);
```

```
   /* determine whether each variable is negated or not */
   for (and_value = max_number / 2, var_i = 0;
         and_value > 0;
            and_value = and_value / 2, var_i++)
   {
      pos_or_neg = input_number & and_value;
      if (pos_or_neg > 0)
      {
         pos_or_neg_vector[var_i] = MUST_BE_POS;
      }
      else
      {
         pos_or_neg_vector[var_i] = MUST_BE_NEG;
      }
   }
}
```

File: c:\CONVERT\reduce_e.c   Creation Date: March 16, 1989

```
/* reduce_e.c
*****************************************************************************
*                              Thesis work                                  *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.           *
*****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will reduce the number of states
      in each equivalence class, by combining two states which differ by
      the value of only one variable.
*/
/*=========================================================================*/

#include "convert.h"
#include "equiv_cl.h"

/*=========================================================================*/

void reduce_equiv_class_states(var_i, equiv_classes)
   short var_i;
   EQ_CLASS_INFO equiv_classes[MAX_N_VARIABLES];
{
   BOOLEAN can_combine;
   short class_i;
   STATE_INFO *first_combine_ele;
   short n_diff;
   STATE_INFO *second_combine_ele;
   STATE_INFO *temp_p;
   short var_different;
   short var_ii;

   void free();

   /***********************************************************************
***/
   /* reduce number of states for each equiv class by combining if possible
 */
   /***********************************************************************
*/
   for (class_i = 0; class_i < equiv_classes[var_i].n_equiv_classes; class_
i++)
   {
      can_combine = TRUE;
      while (can_combine)
      {
         can_combine = FALSE;
```

```
        /* try to combine each pair of states */
        for (first_combine_ele = equiv_classes[var_i].equiv_class[class_i]
.state_info_list_p;
                first_combine_ele != (STATE_INFO *) NULL;
                   first_combine_ele = first_combine_ele->next)
        {
           for (second_combine_ele = first_combine_ele->next;
                   second_combine_ele != (STATE_INFO *) NULL;
                      second_combine_ele = second_combine_ele->next)
           {
              n_diff = 0;
              /* states can be combined if they differ by only one value *
/
              for (var_ii = 0; var_ii < MAX_N_VARIABLES; var_ii++)
              {
                 if (first_combine_ele->pos_or_neg_state[var_ii] !=
                       second_combine_ele->pos_or_neg_state[var_ii])
                 {
                    n_diff++;
                    if (n_diff > 1)
                    {
                       /* break out of for loop over variables */
                       break;
                    }
                    else
                    {
                       /* record which feature was different */
                       var_different = var_ii;
                    }
                 }
              }

              /* if these two elements should be combined */
              if (n_diff == 1)
              {
                 can_combine = TRUE;

                 /* indicate that the feature's value doesn't matter */
                 first_combine_ele->pos_or_neg_state[var_different] =
                          DOESNT_MATTER;

                 /* now delete the second_combine_ele from the list */
                 second_combine_ele->previous->next =
                          second_combine_ele->next;
                 /* if it's the last in the list, there is no next */
                 if (second_combine_ele->next != (STATE_INFO *) NULL)
                 {
                    second_combine_ele->next->previous =
                          second_combine_ele->previous;
                 }
                 /* save pointer to previous element for next loop iterati
on */
                 temp_p = second_combine_ele->previous;
                 free(second_combine_ele);
```

```
                    /* reset the second_combine_ele pointer */
                    second_combine_ele = temp_p;
                    equiv_classes[var_i].equiv_class[class_i].n_states--;
                }
            }
        }
    }
}
}
```

```
File: c:\CONVERT\sel_prob.c    Creation Date: March 14, 1989

/* sel_prob.c
*****************************************************************************
*                              Thesis work                                  *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.         *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.            *
*****************************************************************************
*/

/*=========================================================================*/
/*
   Functionality:
      This file contains a routine which will select the appropriate set
      of conditional probabilities for this initial state.
*/
/*=========================================================================*/

#include "convert.h"
#include "tbn_info.h"

/*=========================================================================*/

void select_probabilities(n_variables, n_states, tbn_info, row_state_i,
            probabilities)
   short n_variables;
   short n_states;
   TBN_VAR_INFO tbn_info[MAX_N_VARIABLES];
   short row_state_i;
   float probabilities[MAX_N_VARIABLES];
{
   short combo_i;
   BOOLEAN found_cond_prob;
   short input_pos_or_neg_state[MAX_N_VARIABLES];
   short var_i;
   short var_ii;

   void det_pos_or_neg_states();

   /***********************************************************************
/
   /* select the appropriate set of probabilities for this initial state */
   /**********************************************************************/
   /* determine the pos_or_neg_states for the initial state */
   det_pos_or_neg_states(row_state_i, n_states, input_pos_or_neg_state);

   for (var_i = 0; var_i < n_variables; var_i++)
   {
      /* for each variable, find the one conditional probability which appl
ies */
      for (combo_i = 0; combo_i < tbn_info[var_i].n_combinations; combo_i++
)
      {
```

```
          found_cond_prob = TRUE;

          for (var_ii = 0; var_ii < n_variables; var_ii++)
          {
             if ((tbn_info[var_i].conditional_probs[combo_i].
                      pos_or_neg_state[var_ii] != DOESNT_MATTER) &&
                    (tbn_info[var_i].conditional_probs[combo_i].
                      pos_or_neg_state[var_ii] != input_pos_or_neg_state[var
_ii]))
             {
                /* this conditional probability does not apply, no need
                 * to check the remaining variables
                 */
                found_cond_prob = FALSE;
                break;
             }
          }
          if (found_cond_prob)
          {
             probabilities[var_i] =
                   tbn_info[var_i].conditional_probs[combo_i].probability;
             /* found the correct one, no need to check the remaining
              * conditional probabilities */
             break;
          }
       }
    }
}
```

File: c:\CONVERT\sum_prob.c   Creation Date: March 13, 1989

```
/* sum_prob.c
****************************************************************************
*                              Thesis work                                 *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.       *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
****************************************************************************
*/

/*========================================================================*/
/*
   Functionality:
      This file contains a routine which will determine the states for
      which each variable is positive.  Then these states will be used to
      sum the probability for that variable.
*/
/*========================================================================*/

#include "convert.h"

/*========================================================================*/

ERROR_CODE sum_probs(n_variables, n_states, mrp_table, var_probs_p)
   short n_variables;
   short n_states;
   float **mrp_table;
   float ***var_probs_p;
{
   ERROR_CODE stat;
   short **use_states;

   ERROR_CODE find_states_to_sum();
   ERROR_CODE free_states_to_sum();
   ERROR_CODE sum_states();

   /*****************************************/
   /* find states to sum for each variable */
   /****************************************/
   stat = find_states_to_sum(n_variables, n_states, &use_states);
   if (stat != (ERROR_CODE) no_errors)
      return(stat);

   /****************************************/
   /* sum these entries for each variable */
   /***************************************/
   stat = sum_states(n_variables, n_states, mrp_table, use_states,
            var_probs_p);
   if (stat != (ERROR_CODE) no_errors)
      return(stat);

   /***********************************************/
   /* free array of states to sum for each variable */
```

```
   /**************************************************/
   stat = free_states_to_sum(n_variables, &use_states);
   if (stat != (ERROR_CODE) no_errors)
      return(stat);

   /* denote successful return */
   return((ERROR_CODE) no_errors);
}
```

File: c:\CONVERT\sum_stat.c   Creation Date: March 14, 1989

```
/* sum_stat.c
****************************************************************************
*                                Thesis work                               *
*   (C) Copyright 1989 Linda Mensinger Nunez.  All Rights Reserved.        *
*       This is an unpublished work fully protected by the copyright laws *
*       and is considered a trade secret by the copyright owner.          *
****************************************************************************
*/

/*========================================================================*/
/*
  Functionality:
     This file contains a routine which will sum the specified states for
     each variable.  The resulting sum is the probability for that
     variable (given the initial state).
*/
/*========================================================================*/

#include <stdio.h>

#include "convert.h"

/*========================================================================*/

ERROR_CODE sum_states(n_variables, n_states, mrp_table, use_states, var_pro
bs_p)
   short n_variables;
   short n_states;
   float **mrp_table;
   short **use_states;
   float ***var_probs_p;
{
   short n_states_used;
   float prob;
   short state_i;
   short used_state_i;
   float **var_probs;
   short var_i;

   char *malloc();

   /*****************************************/
   /* sum these entries for each variable */
   /***************************************/

   /* allocate space for the n_variables possible variables */
   if ((var_probs = (float **) malloc(n_variables * sizeof(float *))) ==
                  (float **) NULL)
   {
      fprintf(stderr, "Ran out of space - Aborting program.\n");
      return((ERROR_CODE) malloc_error);
   }
```

```
    /* record the pointer to the var_probs matrix */
    *var_probs_p = var_probs;

    n_states_used = n_states / 2;

    for (var_i = 0; var_i < n_variables; var_i++)
    {
       /* allocate space to hold the probabilities for each input state */
       if ((var_probs[var_i] = (float *)
                malloc(n_states * sizeof(float))) == (float *) NULL)
       {
          fprintf(stderr, "Ran out of space - Aborting program.\n");
          return((ERROR_CODE) malloc_error);
       }

       /* sum the specified entries in each row */
       for (state_i = 0; state_i < n_states; state_i++)
       {
          /* initialize the probabilitiy to zero */
          prob = (float) 0.0;

          /* sum each of the specified entries */
          for (used_state_i = 0; used_state_i < n_states_used; used_state_i+
+)
          {
             prob += mrp_table[state_i][use_states[var_i][used_state_i]];
          }
          var_probs[var_i][state_i] = prob;
       }
    }

    /* denote successful return */
    return((ERROR_CODE) no_errors);
}
```

Appendix C

Sample Input and Output

File: c:\CONVERT\input Creation Date: March 16, 1989

```
4
maid_comes
room_clean
kids_come
kids_invited
0
3
1
2
3
1
4
1
4
y
.15
.3
1
.4
1
0
.8
0
.05
1
.1
0
.3
y
y
```

```
File: c:\CONVERT\output    Creation Date: March 16, 1989

How many variables does the temporal Bayes network involve?
(The maximum that this program can handle is 7)
This temporal Bayes network involves 4 variables.
Please enter a name of no more than 15 characters for variable 1
The name stored for variable 1 is maid_comes.
Please enter a name of no more than 15 characters for variable 2
The name stored for variable 2 is room_clean.
Please enter a name of no more than 15 characters for variable 3
The name stored for variable 3 is kids_come.
Please enter a name of no more than 15 characters for variable 4
The name stored for variable 4 is kids_invited.

How many variables is variable 1 (known as maid_comes) dependent on?
Variable 1 (known as maid_comes) is dependent on 0 variables:
How many variables is variable 2 (known as room_clean) dependent on?
Please enter the number of dependent variable number 1.
Please enter the number of dependent variable number 2.
Please enter the number of dependent variable number 3.
Variable 2 (known as room_clean) is dependent on 3 variables:
     variable 1, known as maid_comes
     variable 2, known as room_clean
     variable 3, known as kids_come
How many variables is variable 3 (known as kids_come) dependent on?
Please enter the number of dependent variable number 1.
Variable 3 (known as kids_come) is dependent on 1 variables:
     variable 4, known as kids_invited
How many variables is variable 4 (known as kids_invited) dependent on?
Please enter the number of dependent variable number 1.
Variable 4 (known as kids_invited) is dependent on 1 variables:
     variable 4, known as kids_invited

Now accepting 1 (conditional) probabilities for variable 1 (maid_comes)
Please enter the probability P(maid_comes at t)
The probability stored was 0.150000

Now accepting 8 (conditional) probabilities for variable 2 (room_clean)
Please enter the probability P(room_clean at t | maid_comes AND room_clean
AND kids_come at t-1)
The probability stored was 0.300000
Please enter the probability P(room_clean at t | maid_comes AND room_clean
AND NOT kids_come at t-1)
The probability stored was 1.000000
Please enter the probability P(room_clean at t | maid_comes AND NOT room_cl
ean AND kids_come at t-1)
The probability stored was 0.400000
Please enter the probability P(room_clean at t | maid_comes AND NOT room_cl
ean AND NOT kids_come at t-1)
The probability stored was 1.000000
Please enter the probability P(room_clean at t | NOT maid_comes AND room_cl
ean AND kids_come at t-1)
The probability stored was 0.000000
Please enter the probability P(room_clean at t | NOT maid_comes AND room_cl
```

```
ean AND NOT kids_come at t-1)
The probability stored was 0.800000
Please enter the probability P(room_clean at t ¦ NOT maid_comes AND NOT roo
m_clean AND kids_come at t-1)
The probability stored was 0.000000
Please enter the probability P(room_clean at t ¦ NOT maid_comes AND NOT roo
m_clean AND NOT kids_come at t-1)
The probability stored was 0.050000

Now accepting 2 (conditional) probabilities for variable 3 (kids_come)
Please enter the probability P(kids_come at t ¦ kids_invited at t-1)
The probability stored was 1.000000
Please enter the probability P(kids_come at t ¦ NOT kids_invited at t-1)
The probability stored was 0.100000

Now accepting 2 (conditional) probabilities for variable 4 (kids_invited)
Please enter the probability P(kids_invited at t ¦ kids_invited at t-1)
The probability stored was 0.000000
Please enter the probability P(kids_invited at t ¦ NOT kids_invited at t-1)
The probability stored was 0.300000


The MRP transition table has 16 possible states.


Row 1 of the MRP transition table is for input state 0
The table value for input state   0, output state   0 is 0.508725
The table value for input state   0, output state   1 is 0.218025
The table value for input state   0, output state   2 is 0.056525
The table value for input state   0, output state   3 is 0.024225
The table value for input state   0, output state   4 is 0.026775
The table value for input state   0, output state   5 is 0.011475
The table value for input state   0, output state   6 is 0.002975
The table value for input state   0, output state   7 is 0.001275
The table value for input state   0, output state   8 is 0.089775
The table value for input state   0, output state   9 is 0.038475
The table value for input state   0, output state  10 is 0.009975
The table value for input state   0, output state  11 is 0.004275
The table value for input state   0, output state  12 is 0.004725
The table value for input state   0, output state  13 is 0.002025
The table value for input state   0, output state  14 is 0.000525
The table value for input state   0, output state  15 is 0.000225

Row 2 of the MRP transition table is for input state 1
The table value for input state   1, output state   0 is 0.000000
The table value for input state   1, output state   1 is 0.000000
The table value for input state   1, output state   2 is 0.807500
The table value for input state   1, output state   3 is 0.000000
The table value for input state   1, output state   4 is 0.000000
The table value for input state   1, output state   5 is 0.000000
The table value for input state   1, output state   6 is 0.042500
The table value for input state   1, output state   7 is 0.000000
The table value for input state   1, output state   8 is 0.000000
The table value for input state   1, output state   9 is 0.000000
The table value for input state   1, output state  10 is 0.142500
```

```
The table value for input state   1, output state  11 is 0.000000
The table value for input state   1, output state  12 is 0.000000
The table value for input state   1, output state  13 is 0.000000
The table value for input state   1, output state  14 is 0.007500
The table value for input state   1, output state  15 is 0.000000

Row 3 of the MRP transition table is for input state 2
The table value for input state   2, output state   0 is 0.535500
The table value for input state   2, output state   1 is 0.229500
The table value for input state   2, output state   2 is 0.059500
The table value for input state   2, output state   3 is 0.025500
The table value for input state   2, output state   4 is 0.000000
The table value for input state   2, output state   5 is 0.000000
The table value for input state   2, output state   6 is 0.000000
The table value for input state   2, output state   7 is 0.000000
The table value for input state   2, output state   8 is 0.094500
The table value for input state   2, output state   9 is 0.040500
The table value for input state   2, output state  10 is 0.010500
The table value for input state   2, output state  11 is 0.004500
The table value for input state   2, output state  12 is 0.000000
The table value for input state   2, output state  13 is 0.000000
The table value for input state   2, output state  14 is 0.000000
The table value for input state   2, output state  15 is 0.000000

Row 4 of the MRP transition table is for input state 3
The table value for input state   3, output state   0 is 0.000000
The table value for input state   3, output state   1 is 0.000000
The table value for input state   3, output state   2 is 0.850000
The table value for input state   3, output state   3 is 0.000000
The table value for input state   3, output state   4 is 0.000000
The table value for input state   3, output state   5 is 0.000000
The table value for input state   3, output state   6 is 0.000000
The table value for input state   3, output state   7 is 0.000000
The table value for input state   3, output state   8 is 0.000000
The table value for input state   3, output state   9 is 0.000000
The table value for input state   3, output state  10 is 0.150000
The table value for input state   3, output state  11 is 0.000000
The table value for input state   3, output state  12 is 0.000000
The table value for input state   3, output state  13 is 0.000000
The table value for input state   3, output state  14 is 0.000000
The table value for input state   3, output state  15 is 0.000000

Row 5 of the MRP transition table is for input state 4
The table value for input state   4, output state   0 is 0.107100
The table value for input state   4, output state   1 is 0.045900
The table value for input state   4, output state   2 is 0.011900
The table value for input state   4, output state   3 is 0.005100
The table value for input state   4, output state   4 is 0.428400
The table value for input state   4, output state   5 is 0.183600
The table value for input state   4, output state   6 is 0.047600
The table value for input state   4, output state   7 is 0.020400
The table value for input state   4, output state   8 is 0.018900
The table value for input state   4, output state   9 is 0.008100
The table value for input state   4, output state  10 is 0.002100
The table value for input state   4, output state  11 is 0.000900
```

The table value for input state 4, output state 12 is 0.075600
The table value for input state 4, output state 13 is 0.032400
The table value for input state 4, output state 14 is 0.008400
The table value for input state 4, output state 15 is 0.003600

Row 6 of the MRP transition table is for input state 5
The table value for input state 5, output state 0 is 0.000000
The table value for input state 5, output state 1 is 0.000000
The table value for input state 5, output state 2 is 0.170000
The table value for input state 5, output state 3 is 0.000000
The table value for input state 5, output state 4 is 0.000000
The table value for input state 5, output state 5 is 0.000000
The table value for input state 5, output state 6 is 0.680000
The table value for input state 5, output state 7 is 0.000000
The table value for input state 5, output state 8 is 0.000000
The table value for input state 5, output state 9 is 0.000000
The table value for input state 5, output state 10 is 0.030000
The table value for input state 5, output state 11 is 0.000000
The table value for input state 5, output state 12 is 0.000000
The table value for input state 5, output state 13 is 0.000000
The table value for input state 5, output state 14 is 0.120000
The table value for input state 5, output state 15 is 0.000000

Row 7 of the MRP transition table is for input state 6
The table value for input state 6, output state 0 is 0.535500
The table value for input state 6, output state 1 is 0.229500
The table value for input state 6, output state 2 is 0.059500
The table value for input state 6, output state 3 is 0.025500
The table value for input state 6, output state 4 is 0.000000
The table value for input state 6, output state 5 is 0.000000
The table value for input state 6, output state 6 is 0.000000
The table value for input state 6, output state 7 is 0.000000
The table value for input state 6, output state 8 is 0.094500
The table value for input state 6, output state 9 is 0.040500
The table value for input state 6, output state 10 is 0.010500
The table value for input state 6, output state 11 is 0.004500
The table value for input state 6, output state 12 is 0.000000
The table value for input state 6, output state 13 is 0.000000
The table value for input state 6, output state 14 is 0.000000
The table value for input state 6, output state 15 is 0.000000

Row 8 of the MRP transition table is for input state 7
The table value for input state 7, output state 0 is 0.000000
The table value for input state 7, output state 1 is 0.000000
The table value for input state 7, output state 2 is 0.850000
The table value for input state 7, output state 3 is 0.000000
The table value for input state 7, output state 4 is 0.000000
The table value for input state 7, output state 5 is 0.000000
The table value for input state 7, output state 6 is 0.000000
The table value for input state 7, output state 7 is 0.000000
The table value for input state 7, output state 8 is 0.000000
The table value for input state 7, output state 9 is 0.000000
The table value for input state 7, output state 10 is 0.150000
The table value for input state 7, output state 11 is 0.000000
The table value for input state 7, output state 12 is 0.000000

```
The table value for input state   7, output state  13 is 0.000000
The table value for input state   7, output state  14 is 0.000000
The table value for input state   7, output state  15 is 0.000000

Row 9 of the MRP transition table is for input state 8
The table value for input state   8, output state   0 is 0.000000
The table value for input state   8, output state   1 is 0.000000
The table value for input state   8, output state   2 is 0.000000
The table value for input state   8, output state   3 is 0.000000
The table value for input state   8, output state   4 is 0.535500
The table value for input state   8, output state   5 is 0.229500
The table value for input state   8, output state   6 is 0.059500
The table value for input state   8, output state   7 is 0.025500
The table value for input state   8, output state   8 is 0.000000
The table value for input state   8, output state   9 is 0.000000
The table value for input state   8, output state  10 is 0.000000
The table value for input state   8, output state  11 is 0.000000
The table value for input state   8, output state  12 is 0.094500
The table value for input state   8, output state  13 is 0.040500
The table value for input state   8, output state  14 is 0.010500
The table value for input state   8, output state  15 is 0.004500

Row 10 of the MRP transition table is for input state 9
The table value for input state   9, output state   0 is 0.000000
The table value for input state   9, output state   1 is 0.000000
The table value for input state   9, output state   2 is 0.000000
The table value for input state   9, output state   3 is 0.000000
The table value for input state   9, output state   4 is 0.000000
The table value for input state   9, output state   5 is 0.000000
The table value for input state   9, output state   6 is 0.850000
The table value for input state   9, output state   7 is 0.000000
The table value for input state   9, output state   8 is 0.000000
The table value for input state   9, output state   9 is 0.000000
The table value for input state   9, output state  10 is 0.000000
The table value for input state   9, output state  11 is 0.000000
The table value for input state   9, output state  12 is 0.000000
The table value for input state   9, output state  13 is 0.000000
The table value for input state   9, output state  14 is 0.150000
The table value for input state   9, output state  15 is 0.000000

Row 11 of the MRP transition table is for input state 10
The table value for input state  10, output state   0 is 0.321300
The table value for input state  10, output state   1 is 0.137700
The table value for input state  10, output state   2 is 0.035700
The table value for input state  10, output state   3 is 0.015300
The table value for input state  10, output state   4 is 0.214200
The table value for input state  10, output state   5 is 0.091800
The table value for input state  10, output state   6 is 0.023800
The table value for input state  10, output state   7 is 0.010200
The table value for input state  10, output state   8 is 0.056700
The table value for input state  10, output state   9 is 0.024300
The table value for input state  10, output state  10 is 0.006300
The table value for input state  10, output state  11 is 0.002700
The table value for input state  10, output state  12 is 0.037800
The table value for input state  10, output state  13 is 0.016200
```

The table value for input state 10, output state 14 is 0.004200
The table value for input state 10, output state 15 is 0.001800

Row 12 of the MRP transition table is for input state 11
The table value for input state 11, output state 0 is 0.000000
The table value for input state 11, output state 1 is 0.000000
The table value for input state 11, output state 2 is 0.510000
The table value for input state 11, output state 3 is 0.000000
The table value for input state 11, output state 4 is 0.000000
The table value for input state 11, output state 5 is 0.000000
The table value for input state 11, output state 6 is 0.340000
The table value for input state 11, output state 7 is 0.000000
The table value for input state 11, output state 8 is 0.000000
The table value for input state 11, output state 9 is 0.000000
The table value for input state 11, output state 10 is 0.090000
The table value for input state 11, output state 11 is 0.000000
The table value for input state 11, output state 12 is 0.000000
The table value for input state 11, output state 13 is 0.000000
The table value for input state 11, output state 14 is 0.060000
The table value for input state 11, output state 15 is 0.000000

Row 13 of the MRP transition table is for input state 12
The table value for input state 12, output state 0 is 0.000000
The table value for input state 12, output state 1 is 0.000000
The table value for input state 12, output state 2 is 0.000000
The table value for input state 12, output state 3 is 0.000000
The table value for input state 12, output state 4 is 0.535500
The table value for input state 12, output state 5 is 0.229500
The table value for input state 12, output state 6 is 0.059500
The table value for input state 12, output state 7 is 0.025500
The table value for input state 12, output state 8 is 0.000000
The table value for input state 12, output state 9 is 0.000000
The table value for input state 12, output state 10 is 0.000000
The table value for input state 12, output state 11 is 0.000000
The table value for input state 12, output state 12 is 0.094500
The table value for input state 12, output state 13 is 0.040500
The table value for input state 12, output state 14 is 0.010500
The table value for input state 12, output state 15 is 0.004500

Row 14 of the MRP transition table is for input state 13
The table value for input state 13, output state 0 is 0.000000
The table value for input state 13, output state 1 is 0.000000
The table value for input state 13, output state 2 is 0.000000
The table value for input state 13, output state 3 is 0.000000
The table value for input state 13, output state 4 is 0.000000
The table value for input state 13, output state 5 is 0.000000
The table value for input state 13, output state 6 is 0.850000
The table value for input state 13, output state 7 is 0.000000
The table value for input state 13, output state 8 is 0.000000
The table value for input state 13, output state 9 is 0.000000
The table value for input state 13, output state 10 is 0.000000
The table value for input state 13, output state 11 is 0.000000
The table value for input state 13, output state 12 is 0.000000
The table value for input state 13, output state 13 is 0.000000
The table value for input state 13, output state 14 is 0.150000

```
The table value for input state  13, output state  15 is 0.000000

Row 15 of the MRP transition table is for input state 14
The table value for input state  14, output state   0 is 0.374850
The table value for input state  14, output state   1 is 0.160650
The table value for input state  14, output state   2 is 0.041650
The table value for input state  14, output state   3 is 0.017850
The table value for input state  14, output state   4 is 0.160650
The table value for input state  14, output state   5 is 0.068850
The table value for input state  14, output state   6 is 0.017850
The table value for input state  14, output state   7 is 0.007650
The table value for input state  14, output state   8 is 0.066150
The table value for input state  14, output state   9 is 0.028350
The table value for input state  14, output state  10 is 0.007350
The table value for input state  14, output state  11 is 0.003150
The table value for input state  14, output state  12 is 0.028350
The table value for input state  14, output state  13 is 0.012150
The table value for input state  14, output state  14 is 0.003150
The table value for input state  14, output state  15 is 0.001350

Row 16 of the MRP transition table is for input state 15
The table value for input state  15, output state   0 is 0.000000
The table value for input state  15, output state   1 is 0.000000
The table value for input state  15, output state   2 is 0.595000
The table value for input state  15, output state   3 is 0.000000
The table value for input state  15, output state   4 is 0.000000
The table value for input state  15, output state   5 is 0.000000
The table value for input state  15, output state   6 is 0.255000
The table value for input state  15, output state   7 is 0.000000
The table value for input state  15, output state   8 is 0.000000
The table value for input state  15, output state   9 is 0.000000
The table value for input state  15, output state  10 is 0.105000
The table value for input state  15, output state  11 is 0.000000
The table value for input state  15, output state  12 is 0.000000
The table value for input state  15, output state  13 is 0.000000
The table value for input state  15, output state  14 is 0.045000
The table value for input state  15, output state  15 is 0.000000

Please confirm this Markov Random Process transition table and
the desire to convert it back to a temporal Bayes network
by pressing y now (anything else will terminate processing)


The prior probability for state   0 is 0.148936
The prior probability for state   1 is 0.063830
The prior probability for state   2 is 0.252955
The prior probability for state   3 is 0.007092
The prior probability for state   4 is 0.118814
The prior probability for state   5 is 0.050920
The prior probability for state   6 is 0.201795
The prior probability for state   7 is 0.005658
The prior probability for state   8 is 0.026283
The prior probability for state   9 is 0.011264
The prior probability for state  10 is 0.044639
The prior probability for state  11 is 0.001252
```

```
The prior probability for state  12 is 0.020967
The prior probability for state  13 is 0.008986
The prior probability for state  14 is 0.035611
The prior probability for state  15 is 0.000998


variable 1 has 1 equivalence classes.
The probability is 0.150000 for 16 states.
variable 2 has 6 equivalence classes.
The probability is 0.050000 for 2 states.
The probability is 0.000000 for 4 states.
The probability is 0.800000 for 2 states.
The probability is 1.000000 for 4 states.
The probability is 0.400000 for 2 states.
The probability is 0.300000 for 2 states.
variable 3 has 2 equivalence classes.
The probability is 0.100000 for 8 states.
The probability is 1.000000 for 8 states.
variable 4 has 2 equivalence classes.
The probability is 0.300000 for 8 states.
The probability is 0.000000 for 8 states.


Variable 1 (known as maid_comes) is dependent on 0 variables:
Variable 2 (known as room_clean) is dependent on 3 variables:
     variable 1, known as maid_comes
     variable 2, known as room_clean
     variable 3, known as kids_come
Variable 3 (known as kids_come) is dependent on 1 variables:
     variable 4, known as kids_invited
Variable 4 (known as kids_invited) is dependent on 1 variables:
     variable 4, known as kids_invited

Now locating 1 (conditional) probabilities for variable 1 (maid_comes)
Locating the probability P(maid_comes at t)
The probability stored was 0.150000

Now locating 8 (conditional) probabilities for variable 2 (room_clean)
Locating the probability P(room_clean at t ¦ maid_comes AND room_clean AND
kids_come at t-1)
The probability stored was 0.300000
Locating the probability P(room_clean at t ¦ maid_comes AND room_clean AND
NOT kids_come at t-1)
The probability stored was 1.000000
Locating the probability P(room_clean at t ¦ maid_comes AND NOT room_clean
AND kids_come at t-1)
The probability stored was 0.400000
Locating the probability P(room_clean at t ¦ maid_comes AND NOT room_clean
AND NOT kids_come at t-1)
The probability stored was 1.000000
Locating the probability P(room_clean at t ¦ NOT maid_comes AND room_clean
AND kids_come at t-1)
The probability stored was 0.000000
Locating the probability P(room_clean at t ¦ NOT maid_comes AND room_clean
AND NOT kids_come at t-1)
```

The probability stored was 0.800000
Locating the probability P(room_clean at t ¦ NOT maid_comes AND NOT room_clean AND kids_come at t-1)
The probability stored was 0.000000
Locating the probability P(room_clean at t ¦ NOT maid_comes AND NOT room_clean AND NOT kids_come at t-1)
The probability stored was 0.050000

Now locating 2 (conditional) probabilities for variable 3 (kids_come)
Locating the probability P(kids_come at t ¦ kids_invited at t-1)
The probability stored was 1.000000
Locating the probability P(kids_come at t ¦ NOT kids_invited at t-1)
The probability stored was 0.100000

Now locating 2 (conditional) probabilities for variable 4 (kids_invited)
Locating the probability P(kids_invited at t ¦ kids_invited at t-1)
The probability stored was 0.000000
Locating the probability P(kids_invited at t ¦ NOT kids_invited at t-1)
The probability stored was 0.300000