T-79.5102 **Autumn 2006**
Special Course in Computational Logic
Tutorial 4
Solutions

1. An instance of the 3-SAT problem is often given as a set of clauses $S = \{C_1, \ldots, C_n\}$. Each clause C_i is a set of three literals l_i^1, l_i^2, and l_i^3. A literal is either an atom $A \in \{A_1, \ldots, A_m\}$ or its negation $\neg A$. For the sake of simplicity, we assume that $n \geq 2$ and $m \geq 2$ (we can add $\{A_1, \neg A_1, A_2\}$ and $\{A_2, \neg A_2, A_1\}$ to S without affecting the satisfiability of S).

To show that exact inference in Bayesian networks is NP-hard, we should somehow solve the problem of satisfying S using exact inference in a Bayesian network $N(S)$ constructed from S.

Consider a Bayesian network $N(S)$ with Boolean variables $A_1, \ldots, A_m$ for atoms, $C_1, \ldots, C_n$ for clauses and $S_2, \ldots, S_n$ for conjunctions of the clauses so that S_i is to be true whenever $C_1, \ldots, C_i$ are true.

The CPTs associated with these nodes are constructed as follows.

- A node A_i associated with an atom A_i does not have parents and
$$P(a_i) = P(\neg a_i) = \tfrac{1}{2}.$$
- A node C_j associated with a clause C_j depends directly on the k atoms appearing in its literals; $1 \leq k \leq 3$. The node is deterministic (logical or) so that at most one of the 2^k truth value combinations assigned to its parents makes C_j false. As regards CPT entries, $P(c_j) = 0$ for that combination and $P(c_j) = 1$ for others.
- The node S_2 depends on C_1 and C_2 and $P(s_2 \mid c_1, c_2) = 1$ and $P(s_2) = 0$ otherwise. Thus S_2 is also a deterministic node (logical and). Quite similarly, when $i > 2$, S_i depends on S_{i-1} and C_i. The CPT associated with S_i is defined by $P(s_i \mid s_{i-1}, c_i) = 1$ and $P(s_i) = 0$ for other combinations.

Now we have the following interconnection: the 3-SAT instance S is unsatisfiable if and only if $P(s_n) = 0$. It is also important to note that $N(S)$ can be constructed in time polynomial to the *length* of S (number of symbols needed to represent S as a string). To this end, it is really necessary to introduce $S_2, \ldots, S_n$. If we tried to replace these Boolean variables by a single variable S, the respective CPT in $N(S)$ would become exponential in n (which depends linearly on the length of S). The moral is that we can save space substantially by introducing auxiliary variables.

2. (a) Obviously, we have $\sum_{i=1}^{k} p_i = 1$. The cumulative distribution for $1 \leq j \leq k$ is obtained by summing up the first j probability values:
$$P(X \in \{x_1, \ldots, x_j\}) = \sum_{i=1}^{j} P(X = x_i) = \sum_{i=1}^{j} p_j.$$
This distribution can be calculated for each j as follows (assuming an array $\mathsf{p}[1 \ldots k]$ of the probability values):
$$\mathsf{for}\ j = 1\ \mathsf{to}\ k\ \mathsf{do}\ \mathsf{cp}[j] := \mathsf{p}[j] + \mathsf{cp}[j-1];$$
A sample for X is obtained in time linear to k as follows:

```
r := random();
i := 1;
while cp[i] < r and i < k do i := i + 1;
sample := x[i];
```

Here the array $\mathsf{x}[1 \ldots k]$ contains the discrete values of X. The search for the correct index value i can be boosted by binary search ($\log_2 k$ time can be achieved).

(b) Create an array $\mathsf{index}[1 \ldots \mathrm{N}]$ of index values $1 \ldots k$ so that for each $1 \leq i \leq k$ there are $\mathsf{round}(p_i \times N)$ copies of i successively in the array. Then shuffle the array by doing N exchange operations:

```
for j = 1 to N do
        { i := round(random() × (N + 1 − j)) + (j − 1);
                c := index[j]; index[j] := index[i]; index[i] := c; }
```

Individual samples are generated by executing for each j in the range from 1 to N an assignment $\mathsf{sample} := \mathsf{x}[\mathsf{index}[\mathsf{j}]]$. The distribution obtained in this way may appear too "perfect" for small N but nevertheless this might be a good approximation to use.

Another possibility is to create an array $\mathsf{samples}[1 \ldots M]$ that contains for each $1 \leq i \leq k$, $\mathsf{round}(p_i \times M)$ successive copies of x_i. An individual sample is obtained by executing

$$\mathsf{sample} := \mathsf{samples}[\mathsf{round}(\mathsf{ramdom}() \times M)].$$

About the choice of M: one possibility is that $M \approx N$, or alternatively $M \ll N$, e.g., if $p_i \times M$ values turn out to be integers. The quality of the resulting distribution of X is now tightly connected to that of $\mathsf{random}()$.

For the sake of simplicity, it is assumed above that $\mathsf{round}(\mathsf{random}() \times n)$ gives us a random integer in the range $1 \ldots n$.