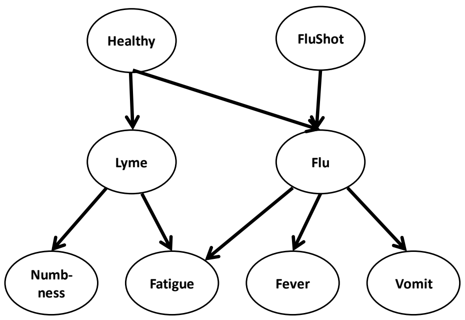


Figure 1: Bayes net defined in `inference.py` as `diagnoseNet`
.

CSE 368 - AI: Homework 5

This homework is about Bayesian networks as modeling tools for probabilistic relationships between random variables (RVs). These models can capture uncertain and weakly informative relationships and at the same time exploit known causal restrictions. There are two written portions (Q1 and Q5). The type annotation is a little wonky, since multiple argument types are permissible for some of the functions you implement. The autograder will only test space separated strings for `sample_space` and `parents` and `dict` as CPT input types.

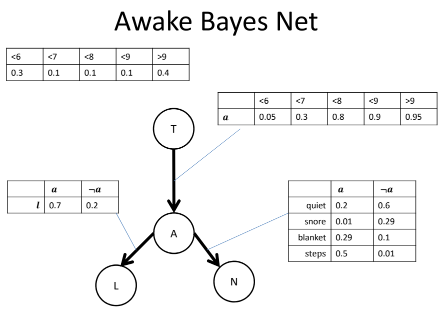
1) **(5) (Short Written) Graphical model questions .** The file `inference.py` defines the Bayes net described in Figure 1. Use the source file and figure to answer the following questions

1. What are the total number of entries in the joint probability distribution without any factoring assumption?
2. What is the probability of $P(fever|\neg flu)$
3. What is the value of $P(flu|\neg healthy, fluShot)$?
4. What is the Markov Blanket of random variable Flu?
5. What is the Markov Blanket of random variable $Fatigue$?

2) **(30) Inference functions.** Implement variable elimination. Your algorithm should use the CPT tables and network structure to enumerate all the entries consistent with some evidence and then marginalize out the hidden variables. In the Inference class define this exact inference method:

```
def enumeration_infer(self,X :str, e :dict = None) -> ProbDist:
```

3) **(15) Approximate inference, Rejection Sampling**. The `Inference` already defines a simple sampling function `sample()` that returns a random assignment according to the joint PDF. Use this to build a rejection



<6	<7	<8	<9	>9
0.3	0.1	0.1	0.1	0.4

	<6	<7	<8	<9	>9
a	0.05	0.3	0.8	0.9	0.95

	a	$\neg a$
l	0.7	0.2

	a	$\neg a$
quiet	0.2	0.6
snore	0.01	0.29
blanket	0.29	0.1
steps	0.5	0.01

Figure 2: Bayes net from class. For the last questions please construct it as `awakeNet`

.

sampling method:
`def rejection_sampling(self,X :str, N :int, e :dict = None) -> ProbDist:`
That returns an estimate based on drawing `N` samples from the network.

4) **(20) Approximate inference, Likelihood Weighting** The rejection sampler ends up having to reject some of the samples since they might not be consistent with evidence. Build a likelihood weighted sampler that only generates consistent samples and then gives them weight proportional to their likelihood of observation in the final average:
`def likelihood_weighting(self, X :str, N:int, e : dict =None) -> ProbDist:` You probably want to define a helper function to return weighted samples.

5) **(15) (Written) Compare Inferences** Compare the three inference algorithms that you have prepared on a plot. Use them to compute $P(Flu \mid \neg fluShot, fever)$, i.e. the distribution over the Flu RV given that the person didn't get a flu shot and is experiencing a fever. Make two plots to examine the convergence behavior of the estimators, for one use 10–100 samples and for the other one use 100–5000 samples. Each plot should have at least 50 and 100 data points respectively. (NOTE: The number of samples is larger due to rejection sampling). The Y-axis should show the estimate using the number of samples given on he X-axis. That means the exact inference algorithm should be a horizontal line of the value the estimates should converge to. In practice how many samples would you use, and which estimator would you choose?
For each point you can either re-sample from scratch, or keep a running estimate and plot that.

6) **(15) Construct a new Bayes net.** Construct the "awake"' Bayes net from class and call the model `awakeNet` (there is a stub in the handout file).