CO 330, LECTURE 31 SUMMARY

FALL 2017

Summary

Today we talked about how to use recursive random generation and Boltzmann generation on labelled classes.

The first thing to note is that it is sufficient just to generate the shape and impose a labelled afterwards because it is easy to generate a random permutation uniformly. However, this doesn't just mean we should use the unlabelled generator and then apply a random permutation to label the output because that won't be uniform among labelled objects. To illustrate the problem we considered rooted trees of size 3. There are two shapes, so from the unlabelled perspective we should generate each with probability 1/2. From the labelled perspective there are 9 labelled rooted trees of size 3. 6 of them have the stick shape and 3 have the branched shape. So if we are generating shapes first and then labelled we need to generate the stick twice as often as the branched one in order to obtain a labelled object uniformly in the end.

Now consider what happens with recursive random generation on labelled classes. $\mathcal{E}$, $\mathcal{Z}$, and $\cup$ are exactly the same as before. $\times$ needs to be slightly modified by weighting with some binomial coefficients. This is to account for the fact that if $\mathcal{A} = \mathcal{B} \star \mathcal{C}$ then the probability of an element of $\mathcal{A}_n$ has its $\mathcal{B}$ part of size k is

$$\binom{n}{k}\frac{b_k c_{n-k}}{a_n}$$

This gives

```
def genA=B*C
  input n
  x = rand()
  k = 0
  s = binom(n,0)*b_0*c_n/a_n
  while x > s
    k = k+1
    s = s+binom(n,k)*b_k*c_{n-k}/a_n
  return (genB(k), genC(n-k))
```

Next consider what happens with Boltzmann generation on labelled classes. First note that we need to use an *exponential Boltzmann model*, so

$$P_x(c) = \frac{x^{|c|}}{|c|!C(x)}$$

where $C(x)$ is the exponential generating function of $\mathcal{C}$. If don't do this then the sum $\sum_{c\in\mathcal{C}} P_c(c)$ won't be 1.

Now consider the pieces of our specifications. $\mathcal{E}$, $\mathcal{Z}$, and $\cup$ are again exactly the same as before.

$\star$ is actually the same as $\times$ though this is a bit less obvious. Suppose $\mathcal{C} = \mathcal{A} \star \mathcal{B}$ then c is a relabelling of a pair (a, b) with $a \in \mathcal{A}$, $b \in \mathcal{B}$. There are $\binom{|a|+|b|}{|a|}$ of these relabellings.

$$P_x(c) = \frac{x^{|c|}}{|c|!C(x)} = \left(\frac{1}{\binom{|a|+|b|}{|a|}}\right)\left(\frac{x^{|a|}}{|a|!A(x)}\right)\left(\frac{x^{|b|}}{|b|!B(x)}\right)$$

which is the probability of picking an element from $\mathcal{B}$, picking an element from $\mathcal{A}$, and uniformly picking a random relabelling, so each of these three pieces is independent. In particular the Boltzmann generator remains the same as the components are independent.

Finally let's consider SET, SEQ, and CYC. Let's look at SET in detail as the others are similar (*pay attention, this is the argument I messed up in class!*) Suppose $\mathcal{C} = \text{SET}(\mathcal{A})$. Then we know the exponential generating functions relate by $\mathcal{C}(x) = \exp(\mathcal{A}(x))$. Furthermore under the exponential Boltzmann distribution, the probability of an element of $\mathcal{C}$ having k components is

$$\begin{aligned}\sum_{b \in \text{SET}_k(\mathcal{A})} \frac{x^{|b|}}{|b|!C(x)} &= \frac{1}{C(x)} \sum_{b \in \text{SET}_k(\mathcal{A})} \frac{x^{|b|}}{|b|!} \\ &= \frac{1}{C(x)} \frac{A(x)^k}{k!} \\ &= \frac{1}{C(x)} \frac{1}{k!} A(x)^k \\ &= \frac{A(x)^k}{k! \exp(A(x))}\end{aligned}$$

which is the Poisson law with $A(x)$ for the parameter. Thus, to generate an element of $\mathcal{C} = \text{SET}(\mathcal{A})$, first pick k by a poisson random generator with parameter $A(x)$ and then return k independent calls to `BoltzmannA`. That is the Boltzmann generator for SET is the same as for Sequence in the unlabelled case except that in place of choosing k from a geometric distribition we choose k from a Poisson distribution. Namely

```
def BoltzmannC=SetA
  input x
  k = poisson_rand()
  return (BoltzmannA(x), ... BoltzmannA(x)) (k times)
```

If you need to implement a Poisson sampler (having uniform `rand()` available) then follow the same code we showed last class but with $p(k) = e^{-k}\lambda^k/k!$.

By similar arguments SEQ is the same as unlabelled and CYC is similar except that k must be chosen from a logarithmic distribution. Namely, $p(k) = \frac{1}{\log(\frac{1}{1-\lambda})}\frac{\lambda^k}{k}$.

References

http://algo.inria.fr/flajolet/Publications/DuFlLoSc04.pdf