PCFG

Challenge Problem 5

The Model

- A probabilistic context-free grammar (PCFG) is defined by a set of productions and their probabilities, a set of terminals, a set of non-terminals and a distinguished start symbol.
- These are given as part of the challenge problem.

Example

S→AB (0.25)
S→BC (0.25)
S→AC (0.4)
S→CA (0.1)

A→a (0.05)
A→b (0.3)
A→S (0.65)

B→b (0.5)
B→c (0.3)
B→d (0.2)

C→d (0.35)
C→e (0.1)
C→S (0.55)

S	S→AC (0.4)
AC	A→a (0.05)
aC	C→S (0.55)
aS	S→AB (0.25)
aAB	A→b (0.3)
abB	B→d (0.2)
abd	

```
val grammar = Map(
  'S -> (() => { Select(0.25 -> List('A, 'B), 0.2 -> List('B, 'C), 0.4 ->  List('A, 'C), 0.15 -> List('C, 'A)) }),
  'A -> (() => { Select(0.05 -> List('a),     0.3 -> List('b),     0.65 -> List('S)) }),
  'B -> (() => { Select(0.5 ->  List('b),     0.3 -> List('c),     0.2 ->  List('d)) }),
  'C -> (() => { Select(0.35 -> List('d),     0.1 -> List('e),     0.55 -> List('S)) })
)

val terminals = Set('a, 'b, 'c, 'd, 'e)

def isTerminal(s: Symbol) = terminals.contains(s)

class Node(s: Symbol, c: List[Node]) {
  val symbol = s
  val children = c

  override def toString() = {
    if (isTerminal(symbol)) symbol.toString().substring(1) else children.mkString
  }
}

def pcfg(symbol: Symbol): Element[Node] = {
  if (isTerminal(symbol)) Constant(new Node(symbol, List(): List[Node]))

  else {
    val dist = grammar(symbol)()
    Chain(dist, (rule: List[Symbol]) => {
      Apply(Inject(rule.map(pcfg): _*), (x: List[Node]) => new Node(symbol, x))
    })
  }
}

def sampleFromPrior[T](e: Element[T]) = {
  (new OneTimeElementSampler(e, 1).sample)._2.values.toList(0)
}

def main(args: Array[String]) {
  val x = pcfg('S)
  for { i <- 1 to 10} println(sampleFromPrior(x))
}
```

The Challenge

$$P(y = bdcb \mid prefix(y) = bd)$$

- The obvious approach of adding observations and doing inference doesn't work.
- Note that the space of possible sentences is unbounded.

My Solution: Sampling

$$P(y = bdcb \mid prefix(y) = bd) = \frac{P(y = bdcb)}{P(prefix(y) = bd)}$$

Since: $P(prefix(y) = bd \mid y = bdcb) = 1$

Result

$$P(y = bdcb \mid prefix(y) = bd)$$

100K samples: 0.00202

Ground truth: 0.002053273

Improvements

- The query is bounded. Restructure model such that inference can exploit this.
- Unbounded queries can be answered using dynamic programming. Figaro has an experimental algorithm (StructuredVE) which perhaps does something similar. More investigation required.