#13: EuPhony and Constraint Solving

Sankha Narayan Guria

EECS 700: Introduction to Program Synthesis

Euphony: strengths

- Efficient way to guide search by a probabilistic grammar
  - Much better than DeepCoder's sort-and-add
  - First to use A* and propose a sound heuristic
- Transfer learning for PHOGs
  - Abstraction is key to learning models of code!
- Extend observational equivalence to top-down search

Euphony: weaknesses

- Requires high-quality training data
  - for each problem domain!
- Transfer learning requires manually designed features

Paper Discussion: Euphony

- **Q2:** What does Euphony use as behavioral constraints? Structural constraint? Search strategy?
  - IO Examples (or first-order formula via CEGIS)
  - PHOG
  - Weighted top-down search via A*

Euphony

Rep x “-” S

- **Q3:** What would these productions look like if we replaced the PHOG with a PCFG? With 3-grams?

PHOG:

```
S[“-”,Rep] ->  “.”     0.72
S[“-”,Rep] ->  “-”     0.001
S[“-”,Rep] ->  x       0.12
S[“-”,Rep] ->  S + S   0.02
…
```

PCFG:

```
S ->  “.”     0.2
S ->  “-”     0.2
S ->  x       0.3
S ->  S + S   0.2
…
```

Sequential 3-grams:

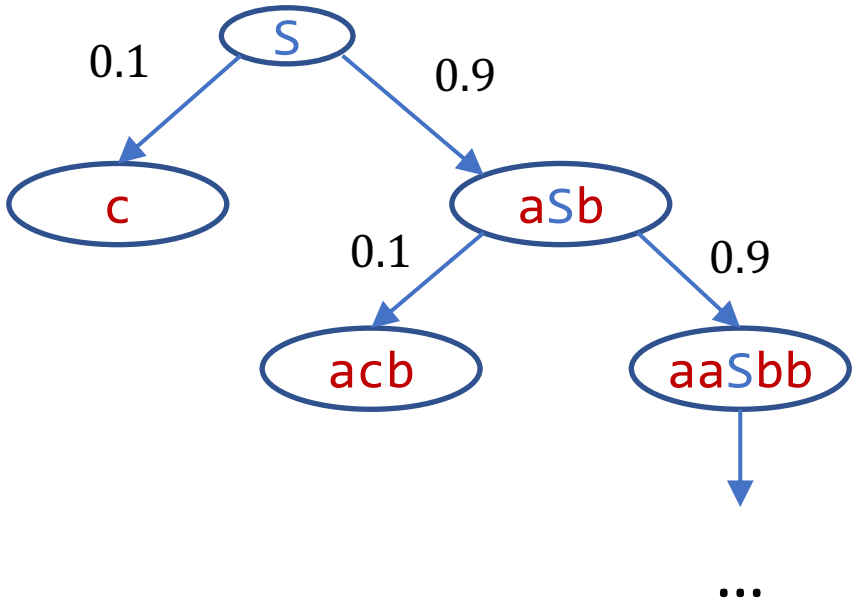
```
S[x,“-”] ->  “.”     0.72
S[x,“-”] ->  “-”     0.001
S[x,“-”] ->  x       0.12
S[x,“-”] ->  S + S   0.02
…
```

- Do you think these other probabilistic models would work as well as a PHOG?

Euphony

- **Q4:** What does h(S) = 0.1 mean? Why is it the case?

```
S ->  a S b  0.9
S ->  c      0.1
```



Euphony

- **Q5:** Give an example of sentential forms n_i, n_j and set of points *pts* such that n_i and n_j are equivalent on *pts* but not weakly equivalent

```
n1 = x + “.”   n2 = “.” + x     pts = [“..”]
```

```
n1 = Rep(“-” + x,”-”, ”.”) + S      n2 = “.” + Rep(x,”-”, ”.”) + S
```

```
n1 = “-” + S   n2 = “-” + “.”     pts = [“-.”]
```

```
n1 = Rep(“-”,“-”,S)    n2 = S
```

```
n1 = Rep(“.”,“-”,S)    n2 = “.”
```

```
 n1 = N + 2     n2 = 1 + N + 1
```

A program for partitioning

```
for i in range(P):
  if i < K:
    sz[i] = n/P + 1
  else:
    sz[i] = n/P
```

How do I ask *the solver* to pick the expression *K*?