University of Alberta
Computing Science Department
Program Synthesis in XAI - W21

Assignment 2
20 Marks
Due date: 05/03 at 23:59

Overview

In this assignment you will implement Probe, the guided bottom-up search algorithm we discussed in class.[1] It is fine to discuss the assignment with your classmates, but do not share your code with them. We will use the forum on eClass and our Discord server to discuss issues related to the assignment.

Description

As in the first assignment, I have attached a Python file with a starter code containing an implementation of the abstract syntax tree (AST) and an interpreter for the language you will use to evaluate Probe.

We will use the following DSL, where "arg" is an input string and "" (empty string), "<", and ">" are constant values.

$$S \rightarrow \text{“”} \mid < \mid > \mid \text{arg} \mid \text{Replace}(S, S, S) \mid \text{Concat}(S, S)$$

We will consider the following test case in our experiments, which was extracted from Probe's paper.

Input (arg)	Output
a < 4 and a > 0	a 4 and a 0
<open and <close>	open and close
<change> <string> to <a> number	change string to a number

In addition to a zip file containing the implementation of the algorithms, described below, you will also submit a report with the empirical results and answers to the questions that are asked.

1. **(6 Marks)** Implement a version of Probe that accepts a probabilistic context-free grammar (PCFG) as input. In this version of Probe you won't update the PCFG with data collected during search. Similarly to the first assignment, your implementation should detect weakly equivalent programs and maintain in the list of programs only one program for each set of outputs. Report the number of programs evaluated by your implementation for solving the test case with the following PCFG.

Production Rule	Probability
arg, "", "<", and ">"	0.188
Replace	0.188
Concat	0.059

[1] See the paper "Just-in-Time Learning for Bottom-Up Enumerative Synthesis" available on eClass.

As a base of comparison, also report the number of programs evaluated if a PCFG with a uniform probability distribution is used. Feel free to set a limit on the number of programs evaluated if the uniformly distributed PCFG requires a very large number of programs to be evaluated. Then, report this limit as a lower bound on the number of programs evaluated by the search procedure.

2. **(7 Marks)** Implement a version of Probe that starts with a uniform distribution over the production rules and adjusts the distribution with data gathered during search, as described in Probe's paper. In this version you will implement the "First Cheapest" (FC) heuristic for selecting which instance is used for adjusting the PCFG. Moreover, the search will be interrupted as soon as program satisfying the FC heuristic is encountered. This means that the PCFG will always be updated with a single program. The search should be restarted once the probability distribution of the PCFG is adjusted.

3. **(7 Marks)** The authors of Probe also described other heuristics for selecting which instances are used for training the model. Answer the following questions about the use of heuristics for selecting the subset of programs used in learning.

   (a) **(3 Marks)** Give an example of a heuristic that performs worse than the FC heuristic in our test case. Explain why your heuristic performs worse. Note that you don't have to implement such a heuristic, but simply describe the heuristic and explain why it performs worse.

   (b) **(4 Marks)** Instead of using heuristics for selecting the programs used for learning, which approach the authors could have used to possibly learn an effective probability distribution while using all programs encountered during search that are able to solve at least one of the input-output pairs? Justify your answer.