1 Method

The syntax of most programming languages is context-free. Our proposed method is simple. We construct a context-free grammar representing the intersection between the langauge syntax and an automaton recognizing the Levenshtein edit ball up to a fixed distance. Since CFLs are closed under intersection with regular languages, this is admissible. Three outcomes are possible:

1. $\mathcal{G}_\cap$ is empty, in which case there is no repair within the given radius. In this case, we simply increase the radius and try again.
2. $\mathcal{G}_\cap$ is small, in which case we simply enumerate all possible repairs. This is the case for 80% of the Python dataset.
3. $\mathcal{G}_\cap$ is too large to exhaustively sample, in which case we use a PCFG to sample $\mathcal{L}(\mathcal{G})$ without replacement, then reject duplicates. This is the case for 20% of the Python dataset.

After sampling, we use an n-gram model to rank the samples, then return the top-k results by likelihood. We depict the flowchart below: