Orel Adivi and Daniel Noor

This is final project in the course "Software Synthesis and Automated Reasoning" (236347)

Technion - Israel Institute of Technology

18 October 2022

Download CorSys

CorSys

- The implementation is available in GitHub: https://github.com/orel-adivi/CorSys
- The project also has a website: https://orel-adivi.github.io/CorSys/
- The README.md contains all the documentation: https://github.com/orel-adivi/CorSys/blob/main/README.md

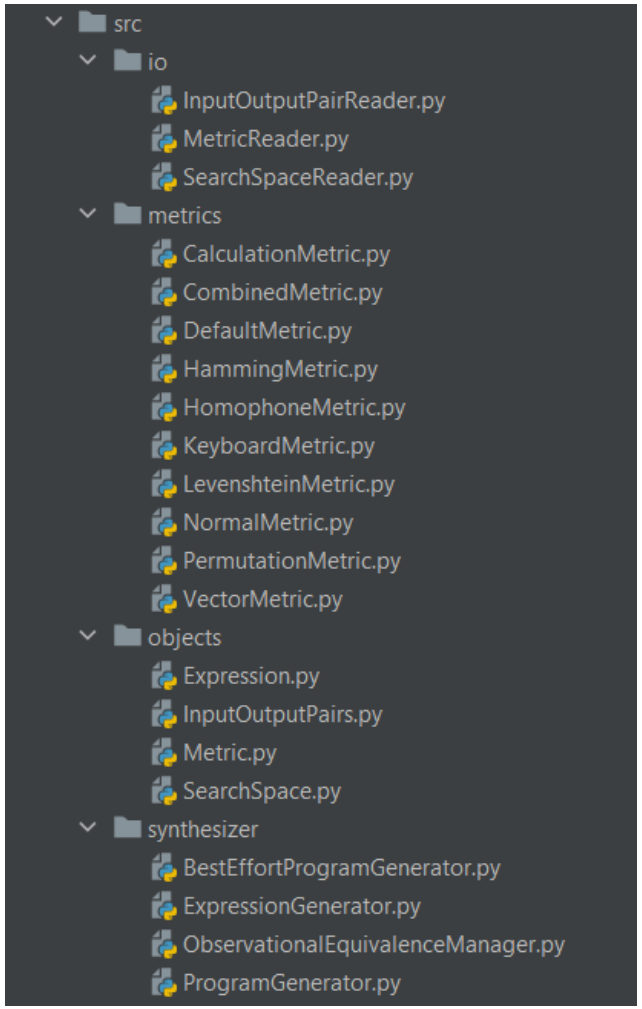
The problem

CorSys

- Programmers make mistakes.
- Input-output specifications given by the user may be incorrect.
- We would like to generalize a concept for a solution, previously presented by Peleg and Polikarpova (2020).

CorSys is a program synthesizer, which synthesizes best-effort Python expressions while weighting the chance for mistakes in given user outputs, using various metrics for mistake probability evaluation.

Usage




```
usage: Synthesizer.py [-h] -io INPUT_OUTPUT_FILE -s SEARCH_SPACE_FILE
                      [-m {DefaultMetric,NormalMetric,CalculationMetric,VectorMetric,HammingMetric,LevenshteinMetric,PermutationMetri
                      [-mp METRIC_PARAMETER]
                      [-t {match,accuracy,height,top,best_by_height,penalized_height,interrupt}]
                      [-tp TACTIC_PARAMETER] [-mh MAX_HEIGHT] [--statistics]

CorSys - Synthesizing best-effort python expressions while weighting the
chance for mistakes in given user outputs.

options:
  -h, --help            show this help message and exit
  -io INPUT_OUTPUT_FILE, --input-output INPUT_OUTPUT_FILE
                        the root for the input-output file
  -s SEARCH_SPACE_FILE, --search-space SEARCH_SPACE_FILE
                        the root for the search space file
  -m {DefaultMetric,NormalMetric,CalculationMetric,VectorMetric,HammingMetric,LevenshteinMetric,PermutationMetric,KeyboardMetric,Homo
                        the metric for the synthesizer (default =
                        'DefaultMetric')
  -mp METRIC_PARAMETER, --metric-parameter METRIC_PARAMETER
                        the parameter for the metric
  -t {match,accuracy,height,top,best_by_height,penalized_height,interrupt}, --tactic {match,accuracy,height,top,best_by_height,penali
                        the tactic for the synthesizer (default = 'height')
  -tp TACTIC_PARAMETER, --tactic-parameter TACTIC_PARAMETER
                        the parameter for the tactic
  -mh MAX_HEIGHT, --max-height MAX_HEIGHT
                        the max height for the synthesizer to search (default
                        = 2)
  --statistics          whether to present statistics

For help with the synthesizer please read SUPPORT.md .
```

From the documentation

Metrics

CorSys

- `DefaultMetric` - this metric uses the default implementation for equality of values.
- `NormalMetric` - this metric uses a normal distribution function to determine the relative distance between two numbers. The `--metric-parameter` defines the standard deviation value for use.
- `CalculationMetric` - this metric considers two values closer if the differences between them can be explained by manual calculation mistakes.
- `VectorMetric` - this metric lets the user choose a vector distance function and then uses it to measure the distance between values. The `--metric-parameter` defines the vector distance function and can be one of `braycurtis`, `canberra`, `correlation`, `cosine`, `jensenshannon`, `hamming`, `jaccard`, `russellrao`, and `yule`.
- `HammingMetric` - this metric computes the Hamming distance between strings and normalizes it according to the string length.
- `LevenshteinMetric` - this metric computes the Levenshtein distance between strings and normalizes it according to the string length. The `--metric-parameter` defines whether to use the recursive implementation (with memoization), in the case the truth value is `True`, or the dynamic programming implementation, in the case the truth value is `False`.
- `PermutationMetric` - this metric considered lists equal if they contain the same elements, regardless of order.
- `KeyboardMetric` - this metric computes the distance between two characters based on the physical distance between their keys on a QWERTY keyboard.
- `HomophoneMetric` - this metric considers two strings closer if they are pronounced similarly.

From the documentation

Tactics

CorSys

- `match` - the first expression whose distance value is equal or less than the defined value is returned. The `--tactic-parameter` defines the threshold distance for returning an expression and should be between 0.0 to the number of examples.
- `accuracy` - the first expression whose distance value, divided by the number of examples, is equal or less than the defined value is returned. The `--tactic-parameter` defines the threshold distance, after normalization, for returning an expression, and should be between 0.0 to 1.0.
- `height` - the best expression, among all possible expressions whose syntax-tree height is up to the defined value, is returned. Please note that the height threshold is defined by `--max-height` parameter, and `--tactic-parameter` is ignored.
- `top` - the best expressions, among all possible expressions whose syntax-tree height is up to the defined value, are returned, one in each line (in descending accuracy). The `--tactic-parameter` defines the number of expressions to return.
- `best_by_height` - the best expressions, among all possible expressions whose syntax-tree height is up to the defined value, are returned, one in each line, so each line represents a different syntax-tree height limit. Please note that the maximal syntax-tree height is defined by `--max-height` parameter, and `--tactic-parameter` is ignored.
- `penalized_height` - the best expression, among all possible expressions whose syntax-tree height is up to the defined, is returned. Each expression is penalized according to its syntax-tree height, so smaller expressions are preferred. The `--tactic-parameter` defines the penalty for each addition of one for the syntax-tree height and should be between 0.0 to 1.0.
- `interrupt` - the best expression, until finishing searching all possible expressions whose syntax-tree height is up to the defined or until keyboard interrupt `(ctrl + c)`, is returned. The `--tactic-parameter` is ignored.

From the documentation

Benchmarks

CorSys

The following benchmarks are available:

- benchmark_1 - this is a sanity benchmark, testing integer expression synthesis with DefaultMetric.
- benchmark_2 - this benchmark tests float expression synthesis with DefaultMetric.
- benchmark_3 - this benchmark tests string-related expression synthesis with DefaultMetric.
- benchmark_4 - this benchmark tests list-related expression synthesis with DefaultMetric.
- benchmark_5 - this is a numerical error benchmark, testing float expression synthesis with NormalMetric.
- benchmark_6 - this is a calculation error benchmark, testing integer expression synthesis with CalculationMetric.
- benchmark_7 - this is a typo benchmark, testing string expression synthesis with LevenshteinMetric.
- benchmark_8 - this is a typo benchmark, testing string expression synthesis with KeyboardMetric.
- benchmark_9 - this is a typo benchmark, testing string expression synthesis with HomophoneMetric.
- benchmark_10 - this is a list-element typo benchmark, testing list expression synthesis with HammingMetric.

From the documentation

Project Engineering

CorSys

Continuous Integration

In order to ensure the correctness of commits sent to the GitHub server, a continuous integration pipeline was set. These checks are run automatically for each pull request and each push. The following actions were set:

1. **Build** - basic tests are run with the updated code, to ensure the lack of syntax errors.
2. **Benchmarks** - all the benchmarks are run with the updated code, to ensure its correctness.
3. **Style check** - the coding style is automatically checked using Flake8, to match the PEP8 coding standard.
4. **Vulnerabilities check** - the updated code is checked to ensure it does not contain any known vulnerability.
5. **Dependency review** - the dependencies are reviewed to check for any security issues.
6. **Website** - the CorSys website is updated with the current information.
7. **Dependabot** - the dependency versions (in requirements.txt) are updated regularly.

For the relevant actions, the checks were run in all the supported Python version (CPython 3.9 and CPython 3.10), and on all supported operating systems - Windows (Windows Server 2022), macOS (macOS Big Sur 11), and Linux (Ubuntu 20.04).

From the documentation

Suggestions for Future Research

CorSys

We could not synthesize expressions whose syntax-tree height is 3 (due to both long runtime and RAM requirements).

- Adding additional metrics
- Analyzing the frequency of user mistakes
- Dealing with incorrect input specifications
- Improving the efficiency with different implementations
- Improving the efficiency with jitting
- Testing the current implementation
- Using the current implementation for different tasks

Demo time!