LapsPython

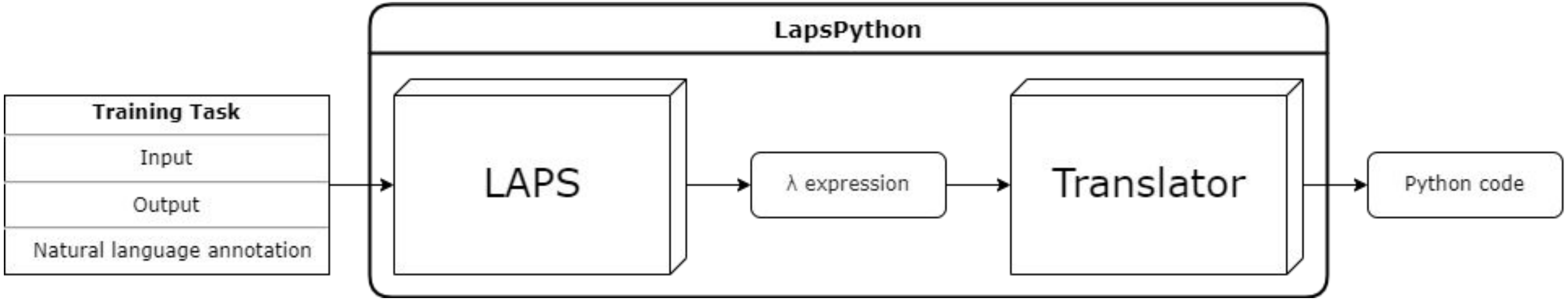
Extend LAPS to synthesize Python/R

Christopher Brückner & Enisa Sabo

23.05.2022

Objective

Extend LAPS to synthesize Python/R code from natural language



- Create rule-based translator from λ-calculus to Python code
- Define sets of primitives and tasks that target useful domains

Project Plan

1. 06.06. Extraction of programs
    - Extract implementations of primitives as strings for translation
    - Extract synthesized lambda expressions to be translated
    - Extract λ expressions from learned library to be translated
    - Parse λ expressions to construct Abstract Syntax Tree

2. 20.06. Translation of programs
    - Implement Python code generation for simple ASTs (arithmetics, procedures)
    - Extend translation to subset of 1 pre-implemented domain (string editing)
    - Extend translation to full domain

3. 04.07. Extension to 2 custom domains
    - Implement primitives for a subset of list processing
    - Implement annotated tasks for a subset of list processing
    - Implement primitives and tasks for a subset of data processing (pandas)

4. 18.07. Extension to 1 domain in R
    - Implement R code generation from ASTs
    - Re-implement the data processing primitives in R (tidyverse)

Example: Extract primitives

```
### Basic regex substring manipulations
def _rnot(s): return f"[^{s}]"
def _ror(s1): return lambda s2: f"(({s1})|({s2}))"
def _rconcat(s1): return lambda s2: s1 + s2
```

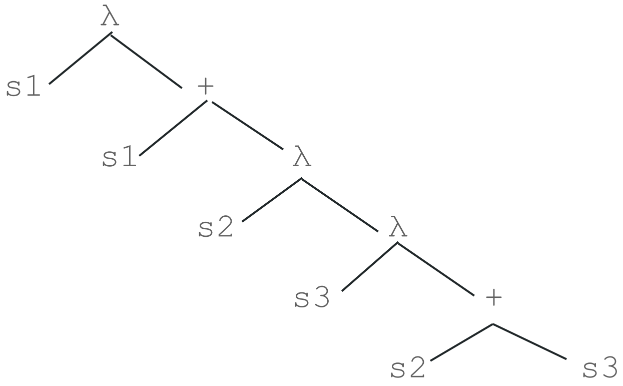
⇓

```
{‘rnot’:    ‘f"[^{s}]"’,
 ‘ror’:     ‘f”(({s1}|{{s2}))’
 ‘rconcat’: ‘s1 + s2’}
```

Example: Parse λ expressions

```
concat_twice = (λ (s1) (s1 + λ (s2) (λ (s3) (s2 + s3))))

concat_twice = lambda s1: s1 + lambda s2: lambda s3: s2 + s3
```



```
concat0 = s2 + s3

concat1 = s1 + concat0
```