LapsPython

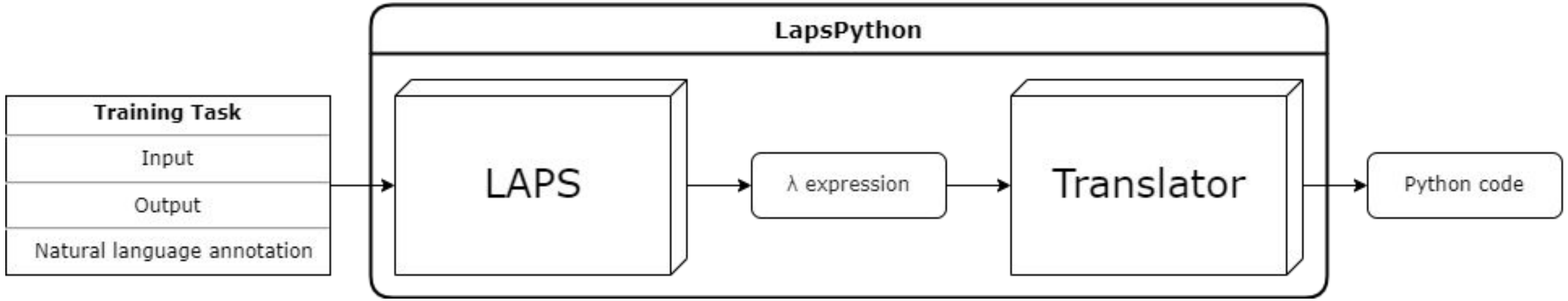
Extend LAPS to synthesize Python/R

Christopher Brückner & Enisa Sabo

30.05.2022

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: Sprint 1

Extraction of programs   Deadline: 06.06.

- Extract implementations of primitives as strings for translation
- Extract synthesized λ expressions to be translated
- Extract λ expressions from learned library to be translated
- Parse λ expressions to construct Abstract Syntax Tree

Example: Extract primitives

```
def _rnot(s): return f"[^{s}]"
def _rconcat(s1): return lambda s2: s1 + s2
```

$$\Downarrow$$

```
{'_rnot':    ('return f"[^{s}]"', ['s']),
 '_rconcat': ('return lambda s2: s1 + s2', ['s1'])}
```

⇒ `{name: (body, [args])}`

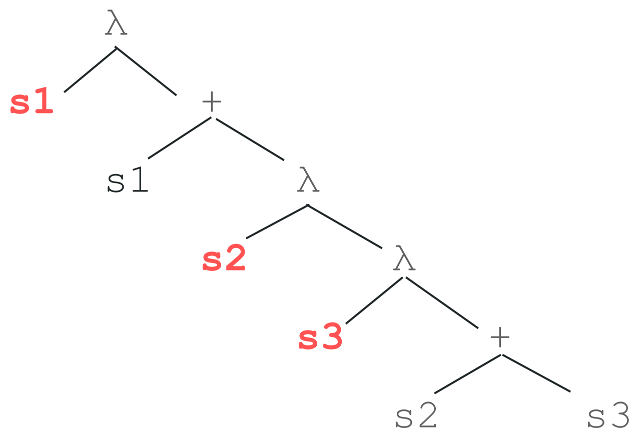
- Approach: regex substitute args in body to resolve variables
- Problem:
    - All primitives have 1 parameter
    - We don’t resolve lambda yet

Next: Extract λ expressions

- 2 locations:
  - Synthesized programs
  - Learned primitives
- LAPS provides a “human readable parser”
- Parsed programs easy to extract, but:
  - Lisp: (λ (x) (f x))
  - LAPS: (λ (f x))

Next: Parse extracted λ expressions

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



```
concat0 = s2 + s3

concat1 = s1 + concat0
```

Further Issue (Sprint 3)

- Creating new domains:
  - Looks very simple for DreamCoder
    - Not at all for LAPS
  - Well documented for DreamCoder
    - Not at all for LAPS
- Main problems:
  - Language annotations
    - LapsTrans will figure it out, we will steal their results
  - Data generation
    - Different domains use different approaches
- Original plan might be too ambitious
  - 2 custom domains ➔ 1 custom domain