



HAL
open science

ACES: generating diverse programming puzzles with autotelic language models and semantic descriptors

Julien Pourcel, Cédric Colas, Pierre-Yves Oudeyer, Laetitia Teodorescu

► To cite this version:

Julien Pourcel, Cédric Colas, Pierre-Yves Oudeyer, Laetitia Teodorescu. ACES: generating diverse programming puzzles with autotelic language models and semantic descriptors. Neurips 2023 - The 37th Annual Conference on Neural Information Processing Systems, Dec 2023, Nouvelle Orleans, United States. hal-04366773

HAL Id: hal-04366773

<https://inria.hal.science/hal-04366773>

Submitted on 29 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

ACES: Generating Diverse Programming Puzzles With Autotelic Language Models And Semantic Descriptors



Julien Pourcel, Cédric Colas, Pierre-Yves Oudeyer, Laetitia Teodorescu
 Contact: julien.pourcel@inria.fr

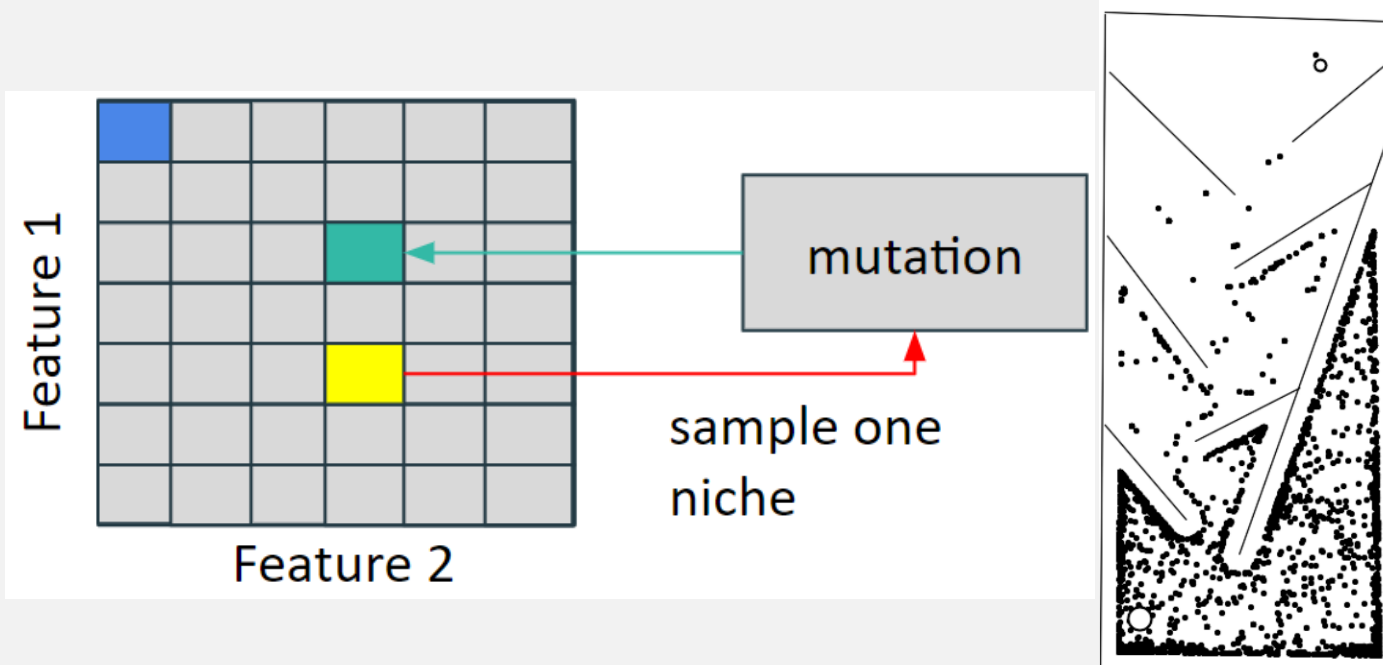


Background

- Generative models usually aim to replicate a reference distribution.
- They generally do not focus on enhancing diversity.
- Methods that do target diversity are limited to narrow or uninterpretable space (embedding space).

Diversity algorithm:

- Define a meaningful features space
- Try to maximize coverage of that space



Python Programming Puzzles (P3)

```
def f(x: float, a=1020) -> bool:
    """Find a number that when squared is close to a."""
    return abs(x ** 2 - a) < 10 ** -3
```

Problem to solve and test function (test units) return True/False
 True: correct solution,
 False: incorrect solution

```
def g(a=1020) -> float:
    return a ** 0.5
```

Solution function
 return a solution

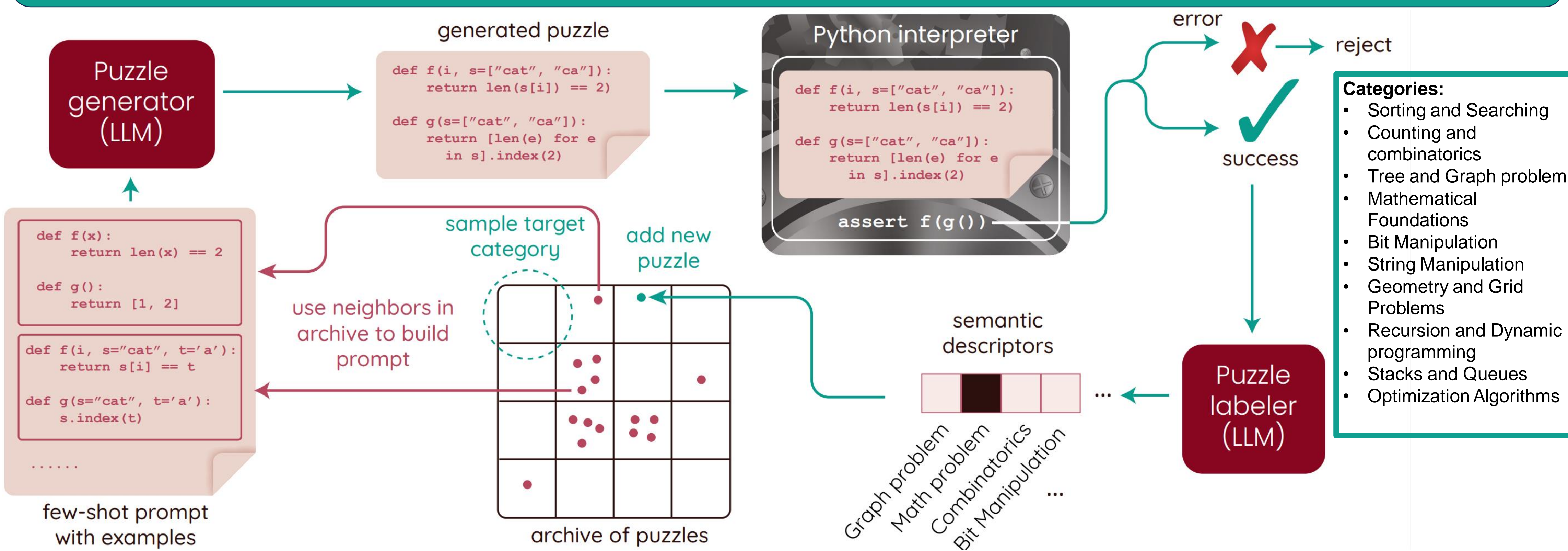
```
assert f(g()) == True
```

Check syntax of the puzzle and correctness

- What Behavioral Characterization space can we use for P3 ?
- How to maximize diversity given limited resources ?

Methods

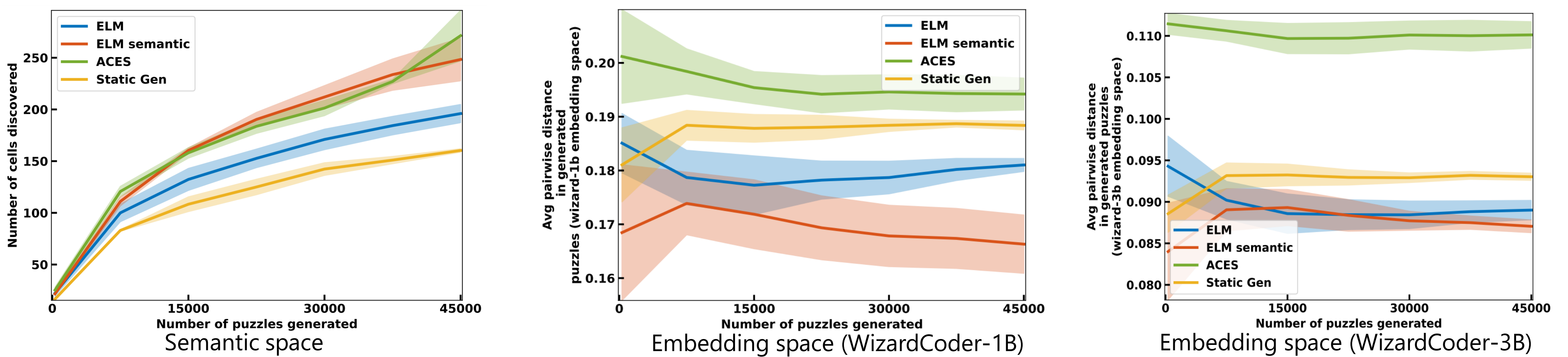
Use language model feedback to create rich human-relevant and interpretable Behavioral Characterization space



- Categories:**
- Sorting and Searching
 - Counting and combinatorics
 - Tree and Graph problem
 - Mathematical Foundations
 - Bit Manipulation
 - String Manipulation
 - Geometry and Grid Problems
 - Recursion and Dynamic programming
 - Stacks and Queues
 - Optimization Algorithms

Results

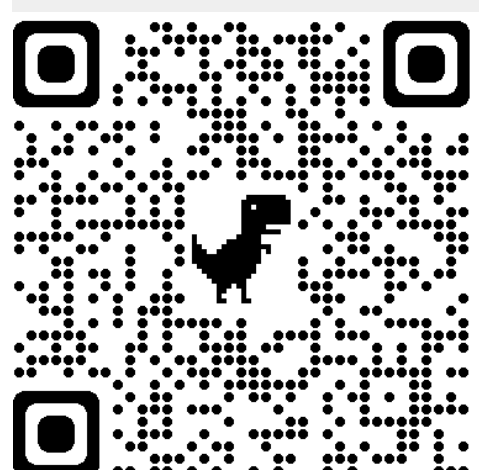
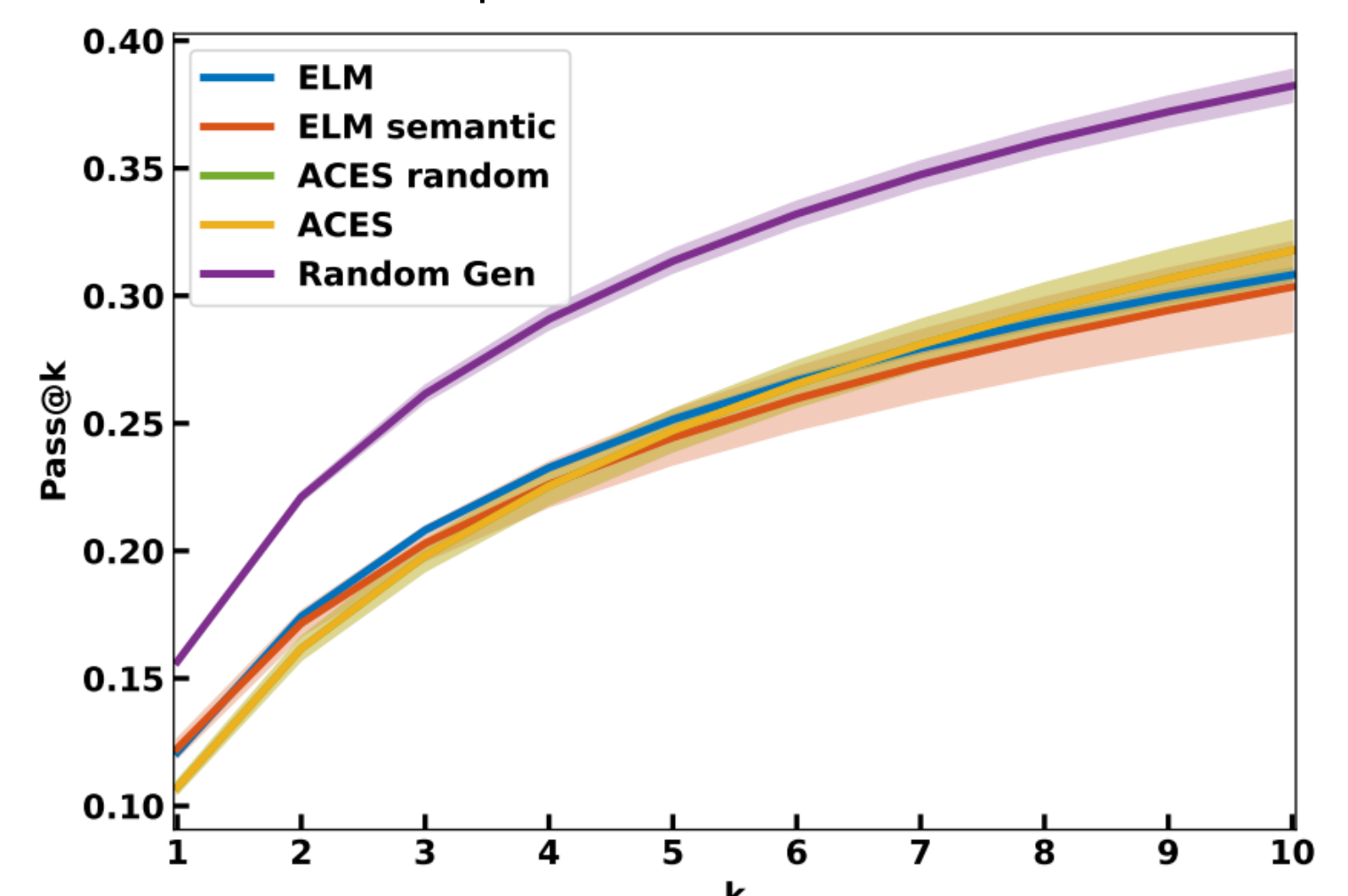
Result 1: Diversity of generated puzzles in semantic and embedding (WizardCoder-1B and 3B) space



Discussion

- Discover richer diversity on a range of metrics
- Semantic space is more interpretable
- Diversity doesn't correlate with downstream performance when fine-tuning language model
 - Quality problem? How can we define quality? (difficulty, ...)
 - Imperfect skills labelling ?

Pass@k on Python Programming Puzzles (P3) test set (number of puzzles solved for k trials)



References:
 Lehman, et al. "Evolution through large models." *Handbook of Evolutionary Machine Learning*. Singapore: Springer Nature Singapore (2023). 331-366.
 Colas, et al. "Language as a cognitive tool to imagine goals in curiosity driven exploration." *Advances in Neural Information Processing Systems* (2020): 3761-3774.