Théo Matricon

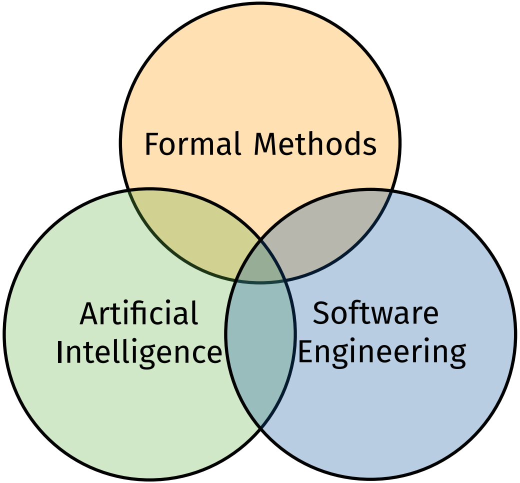
2025-current:

Postdoc in LLM4Code with Mathieu Acher
Code Generation and LLMs
DiverSE, Software Engineering Team
IRISA, Rennes → medical condition (getting RQTH)

2021-2024:

PhD supervised by Nathanaël Fijalkow
Scaling domain agnostic techniques for program Synthesis
M2F, Formal Methods Team
LaBRI, Bordeaux

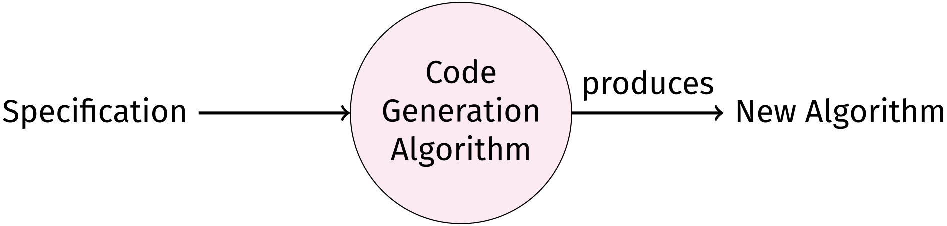
My Research Contributions so far



Selected Publications:

- 2025, submitted to CAV
- 2025, AAAI (+oral: top 20%)
- 2025, IST
- 2022, AAAI (+oral: top 20%)
- 2021, CP

Motivation



Specification

Code Generation Algorithm

produces

New Algorithm

In practice

Specifications:

Logic:

$$\forall a, b$$
$$f(a, b) \geq a$$
$$f(a, b) \geq b$$
$$f(a, b) \in \{a, b\}$$

Examples:

$$f(1, 5) = 5$$
$$f(2, 1) = 2$$
$$f(-3, -9) = -3$$

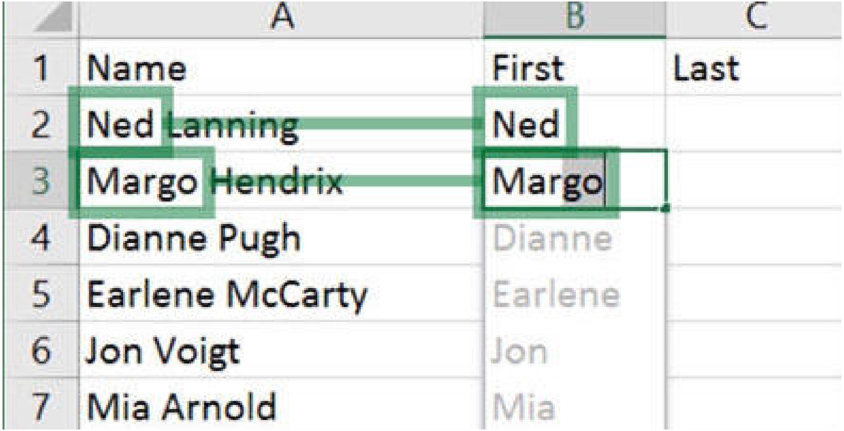
Natural language:

'Write a function that takes the maximum of its two arguments.'

Produced Algorithms:

```
def max(a: int, b: int) ->int:
    if a <=b:
        return b
    else:
        return a
```

Program Synthesis: From Examples



	A	B	C
1	Name	First	Last
2	Ned Lanning	Ned	
3	Margo Hendrix	Margo	
4	Dianne Pugh	Dianne	
5	Earlene McCarty	Earlene	
6	Jon Voigt	Jon	
7	Mia Arnold	Mia	

Copyright Microsoft for syntactic manipulation,
based on papers by *Gulwani et al.* and extensions by ***Matricon*** *et al.*

Program Synthesis: Problem

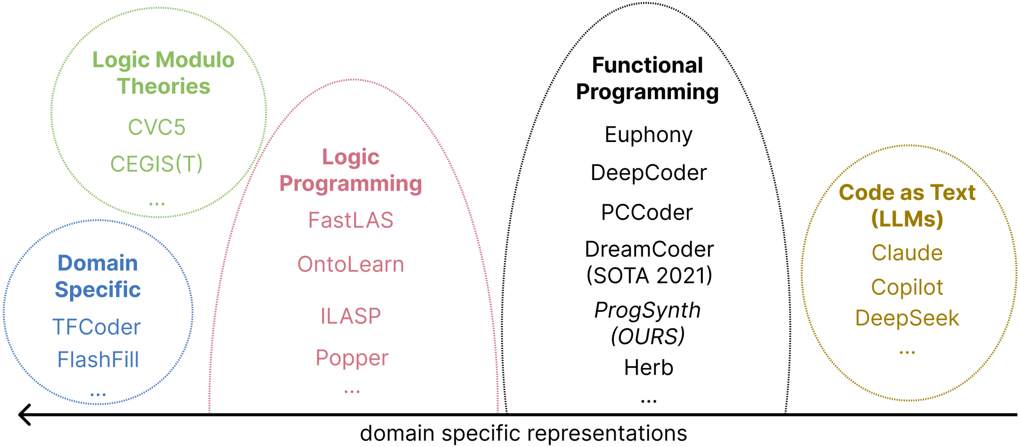
Input:

- the search space G:
  a deterministic tree grammar
- a specification $\mathcal{C}$:
  it checks if a program $p \in \mathcal{L}(G)$ matches the specification

Output:

- a program in the search space that matches the specification:
  a $p \in \mathcal{L}(G)$ such that $\mathcal{C}(p) = \checkmark$

Frameworks



Logic Modulo Theories

CVC5

CEGIS(T)

...

Domain Specific

TFCoder

FlashFill

...

Logic Programming

FastLAS

OntoLearn

ILASP

Popper

...

Functional Programming

Euphony

DeepCoder

PCCoder

DreamCoder (SOTA 2021)

ProgSynth (OURS)

Herb

...

Code as Text (LLMs)

Claude

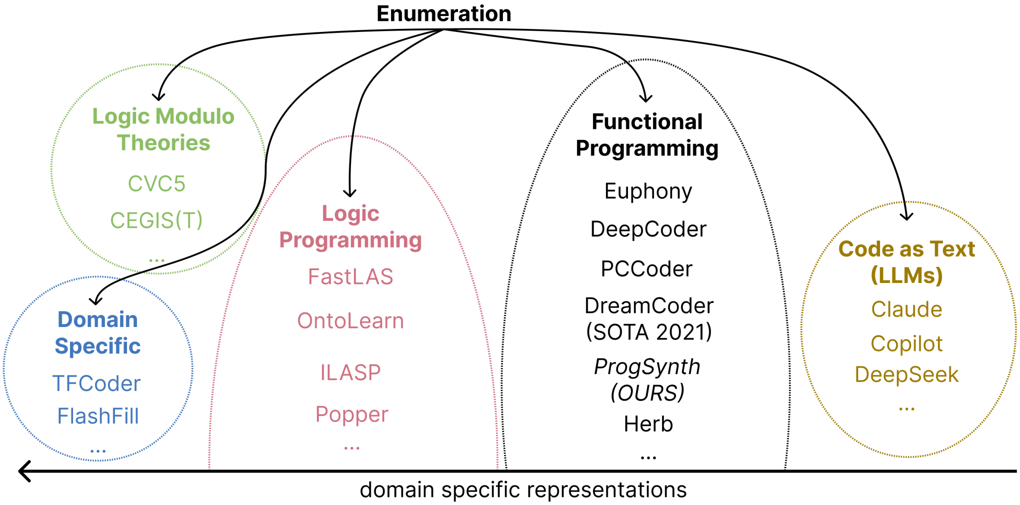
Copilot

DeepSeek

...

domain specific representations

Frameworks



Enumeration
Logic Modulo Theories
CVC5
CEGIS(T)
...
Domain Specific
TFCoder
FlashFill
...
Logic Programming
FastLAS
OntoLearn
ILASP
Popper
...
Functional Programming
Euphony
DeepCoder
PCCoder
DreamCoder (SOTA 2021)
ProgSynth (OURS)
Herb
...
Code as Text (LLMs)
Claude
Copilot
DeepSeek
...
domain specific representations

Enumeration Problem

Input:

- the search space G:
  a deterministic tree grammar with a cost for each tree
- a specification $\mathcal{C}$:
  it checks if a program $p \in \mathcal{L}(G)$ matches the specification

Goal:

Enumerate all programs in order of non-decreasing costs

Delay:

Time complexity in terms of n: number of programs enumerated

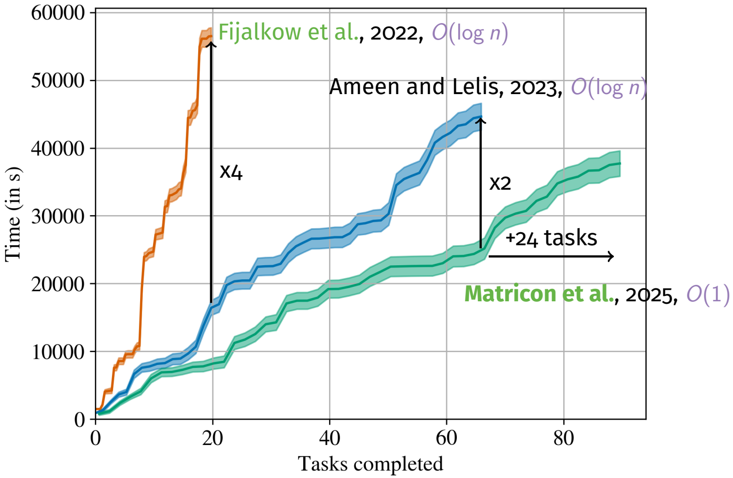
Between enumeration of the n^{th} program and the next

Overview

Major papers:

- 2017, machine learning + enumeration, *Balog et al.*, ICLR
- 2018, $O(\log n)$, *Lee et al.*, PLDI

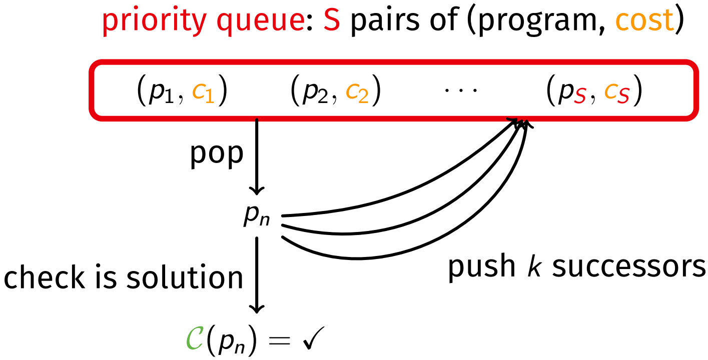
→



Skeleton of an enumeration algorithm

At step n:

$S = O(n)$
$k = O(1)$



priority queue: S pairs of (program, cost)

(p_1, c_1) (p_2, c_2) $\cdots$ (p_S, c_S)

pop

p_n

push k successors

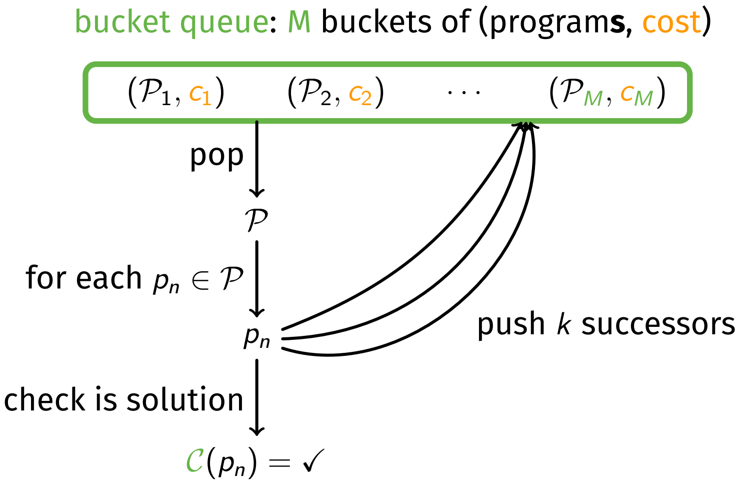
check is solution

$\mathcal{C}(p_n) = \checkmark$

Our enumeration algorithm

At step n:

$M = O(1)$
$k = O(1)$



bucket queue: M buckets of (program**s**, cost)

$(\mathcal{P}_1, c_1)$ $(\mathcal{P}_2, c_2)$ $\cdots$ $(\mathcal{P}_M, c_M)$

pop

$\mathcal{P}$

for each $p_n \in \mathcal{P}$

p_n

push k successors

check is solution

$\mathcal{C}(p_n) = \checkmark$

Our contribution

We prove bounded differences in cost:
$\exists M,\ \forall n,\ \mathit{cost\_next}(p_n) - \mathit{cost}(p_n) \leq M$

This implies: priority queues $O(\log n) \rightarrow$ bucket queues $O(1)$.

Impact

Published in **AAAI 2025** (+oral: 20% of accepted papers).
Fastest ranked enumeration for program synthesis in practice.
First algorithm with $O(1)$ delay → closes open question.

My Research Project

Observations:

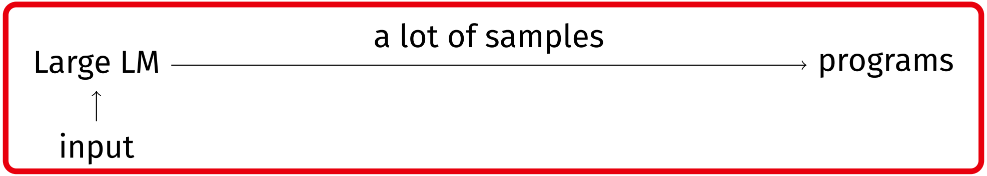
- LLMs are the new state of the art for code generation
- Resource-heavy, expensive and slow
- Exponential increase in data/parameters → linear increase in performance
- Unreliable → only use them as a direction for search

Scaling Code Generation to be reliable and better!

Different code generation paradigm!
Reliable by design, better code with faster generation.

My Research Project

Current Approaches (costly, unreliable, slow)



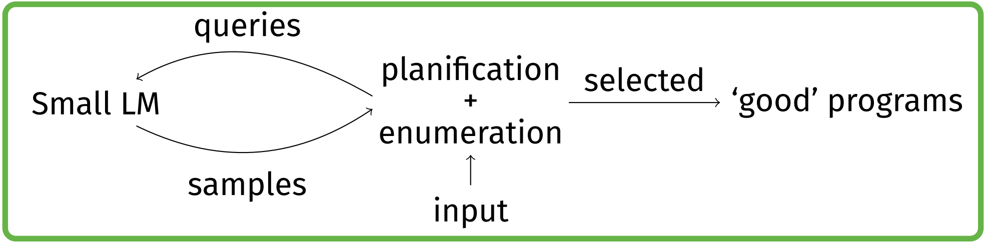
Large LM

a lot of samples

programs

input

My Vision (cheap, reliable, fast)



queries

Small LM

samples

planification + enumeration

input

selected

'good' programs

Axis 1: Hierarchical Code Generation

```python
class DatabaseConnector:
    def open_connection(self, ip: str) ->None:
        do_stuff(self, ip)

    def close_connection(self) ->None:
        if not some_condition():
            raise SomeError()
        do_other_stuff(self)

    def query(self, query: str) ->str:
        return do_thing(self, query)
```

Axis 1: Hierarchical Code Generation

Structure (Easy?)

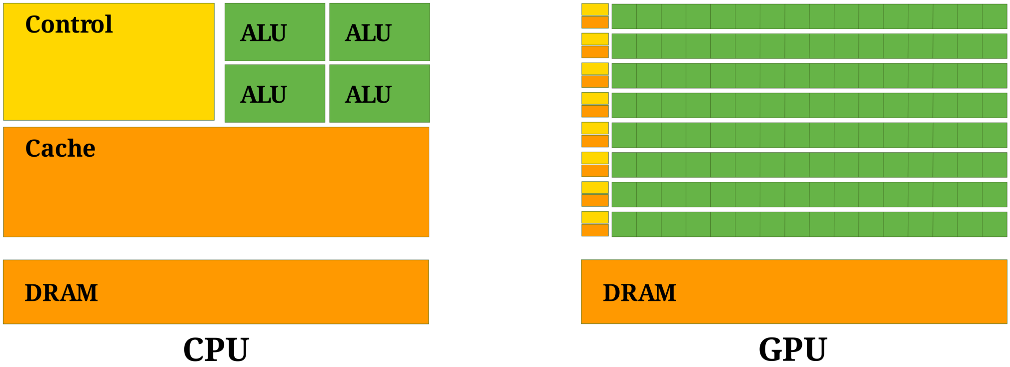
```python
class DatabaseConnector:
    def open_connection(self, ip: str) ->None:
        ???

    def close_connection(self) ->None:
        ???

    def query(self, query: str) ->str:
        ???
```

Filling (Hard?)

Axis 2: Local GPU-friendly Search



CC-by-3.0 NVIDIA

GPUs: massively **parallel** → huge speed-up
But **different paradigm**

Axis 2: Local GPU-friendly Search

Current approaches cannot be adapted efficiently.

We need a **new enumeration paradigm** for GPUs.

It requires a **new theoretical** understanding.

Synergies

Long Term Objective:

Axis 1: provides practical reduction of the complexity of the problem.
Axis 2: improves the search efficiency.

Orthogonal directions that can be coupled together for better results!

Integration

LLM4Code project
Cross-Team interactions with FM and AI

→ CRIStAL, UMR 9189, Lille, **Spirals**

SE and AI: Clément Quinton, Romain Rouvoy
Enumeration: Pierre Bourhis (+ **LINKS**)

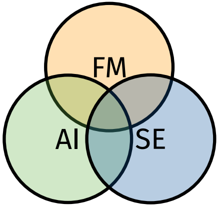
→ IRISA, UMR 6074, Rennes, **DiverSE**

SE and AI: Mathieu Acher, Olivier Barais, Benoît Combemale, Aymeric Blot, Quentin Perez, Djamel E. Khelladi
CodeCommons project

→ LaBRI, UMR 5800, Bordeaux, **Progress**

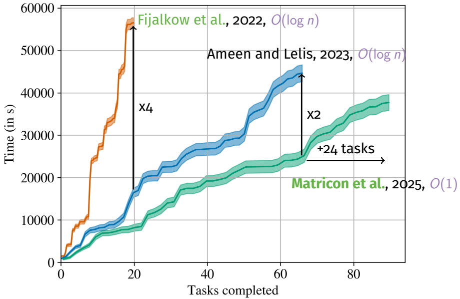
SE and AI: Romain Robbes, Xavier Blanc, Jean-Rémy Falleri and Thomas Degueule
Enumeration: (+ **M2F**)

Théo Matricon



Selected Publications:

- AI: 2 A^* →
- FM: 1 A, A^* submitted
- SE: 1 A



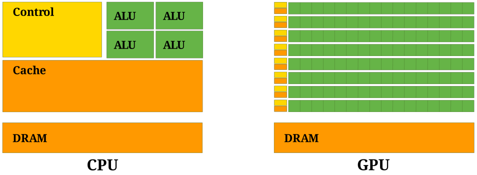
Research Project:

Structure (Easy?)   Filling (Hard?)

```
class DatabaseConnector:
    def open_connection(self, ip: str) ->None:
        ???

    def close_connection(self) ->None:
        ???

    def query(self, query: str) ->str:
        ???
```


Control
ALU
ALU
ALU
ALU
Cache
DRAM
CPU
DRAM
GPU