PROGRAM SYNTHESIS

Nathanaël Fijalkow

CNRS, LaBRI, Bordeaux, France

& University of Warsaw, Poland

OUTLINE

- The multiple benefits of Domain Specific Language

"a program has structure, it is not a raw piece of text"

- The need for proper interfaces

"LLMs + fact-checking = future"

- The TRUE Programmatic Reinforcement Learning

"Program you are, and to program you will return"

environment — policy

BACK IN 2011: FLASHFILL (Microsoft)

Summit Attendees

File Edit View Insert Format Data Tools Add-ons Help

	A	B	C	D	E	F
1	**Attendees**	**Office Address**	**Hi _____**	**E-mail**	**Zip Code**	
2	Gerald Parker	320 N Morgan St #600, Chicago, IL 60607	Hi Gerald			
3	Phillip Davis	111 8th Ave. New York, NY 10011				
4	Ralph Gonzales	2930 Pearl Street, Boulder, CO. 80301				
5	Shawn Jenkins	1160 Bordeaux Drive, Sunnyvale, CA 94089				
6	Thomas Murphy	1160 Bordeaux Drive, Sunnyvale, CA 94089				
7	Norma Long	2930 Pearl Street, Boulder, CO. 80301				
8	Beverly Harris	320 N Morgan St #600, Chicago, IL 60607				
9	Melissa Torres	1160 Bordeaux Drive, Sunnyvale, CA 94089				
10	Joshua Watson	111 8th Ave. New York, NY 10011				
11	Larry Wright	500 W 2nd St. Austin, TX 78701				
12	Lois Brooks	2930 Pearl Street, Boulder, CO. 80301				
13	Steve James	1160 Bordeaux Drive, Sunnyvale, CA 94089				
14	Helen Rivera	2930 Pearl Street, Boulder, CO. 80301				
15	Samuel Bailey	111 8th Ave. New York, NY 10011				
16	Kenneth Gonzalez	2930 Pearl Street, Boulder, CO. 80301				
17	Sharon Washington	1160 Bordeaux Drive, Sunnyvale, CA 94089				
18	Mildred Evans	1160 Bordeaux Drive, Sunnyvale, CA 94089				

(this picture is actually Smartfill, Google's version)

Limitations: • it doesn't know that Paris is in France!
• it doesn't output the program so we cannot trust the results

BACK IN 2021: CODEX (OpenAI / Github)

```python
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, value, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2016-01-02 -34.01 USD
        2016-01-03 2.59 DKK
        2016-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                         float(value),
                         currency))
    return expenses
```

Copilot

Limitations: treats code as raw text: a lot of syntactic errors!

A FEW MONTHS AGO: BARD (Google)

What new discoveries from the James Space Webb Telescope can I tell my 9 year old about?

Your 9-year old might like these recent discoveries made by The James Webb Space Telescope (JWST):

- **In 2023, The JWST discovered a number of galaxies nicknamed "green peas."** They were given this name because they are small and round, like peas.
- **The telescope captured images of galaxies that are over 13 billion years old.** This means that the light from these galaxies has been traveling for over 13 billion years to reach us.
- **JWST took the very first pictures of a planet outside of our own solar system.** These distant worlds are called "exoplanets". Exo means "from outside".

These discoveries can spark a child's imagination about the infinite wonders of the universe

Check it

Limitations: it doesn't know, it learns (from biassed data)

Program Synthesis

SPECIFICATION ⟹ PROGRAM

logical formula
examples
natural language description
demonstration
⋮

code
System
Chip
macro
⋮

- One of the oldest dream of AI
- For decades only for delusional researchers
- The upcoming revolution for | developpers
  end users ?

RESEARCH PERSPECTIVE ON SYNTHESIS

- HUGE practical success and impact
- Theory is lagging behind but highly needed
- at the intersection of
  - Programming Languages
  - Software Engineering
  - Machine learning
  - Algorithms

PART I — SYNTHESIS IN A DSL

All Large Language Models (LLM) tools I know

write code in Python / C / Java / ...

I believe this is **WRONG** : complicated syntax / semantics

- program aliasing
- no partial execution
- hard to reason with
- basis errors
- no lifting to other languages

DSL : Domain Specific Language

- Designed for a specific class of tasks
- Defined by a family of primitives

$\Longrightarrow$ FUNCTIONAL PROGRAMMING

EXAMPLE DSL

list of primitives:

- sort : $list(int) \longrightarrow list(int)$
- map : $(t_0 \to t_1) \longrightarrow list(t_0) \longrightarrow list(t_1)$
- succ : $int \longrightarrow int$
- ⋮

(their types)

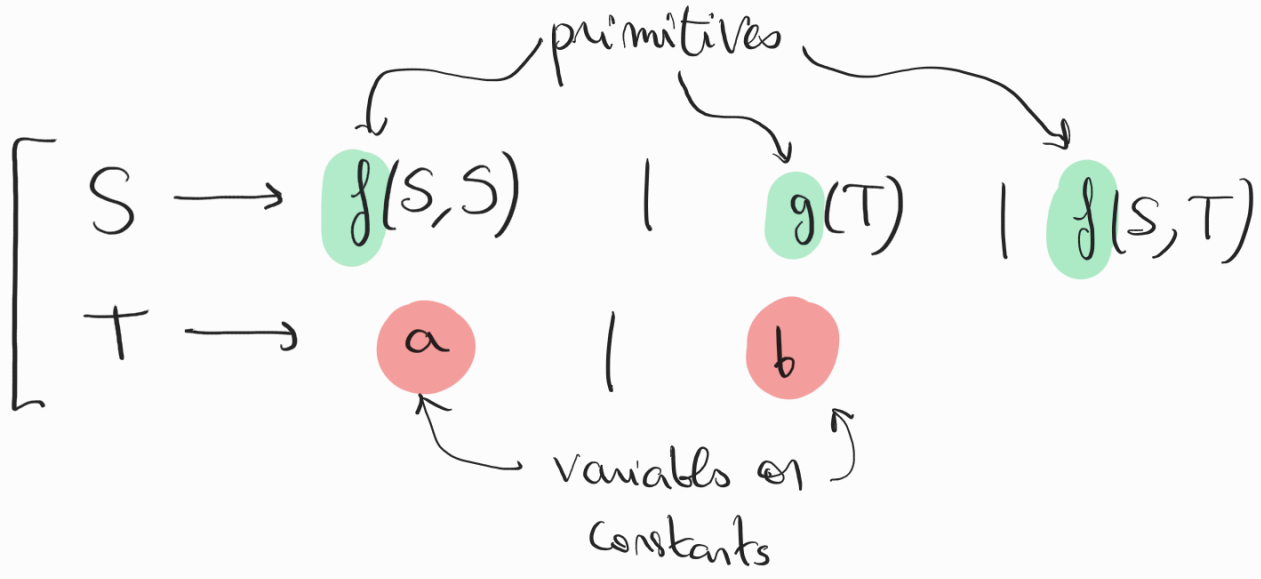
TYPES

atomic types : int, bool, string, list(int), ...

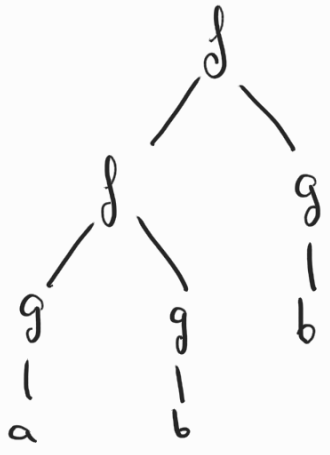
generic types : $t_0, t_1, \ldots$

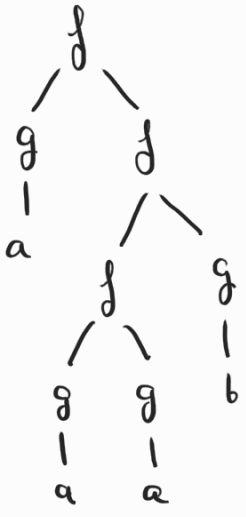
constructor : $\tau_1 \longrightarrow \tau_2$: functions

CONTEXT FREE GRAMMARS



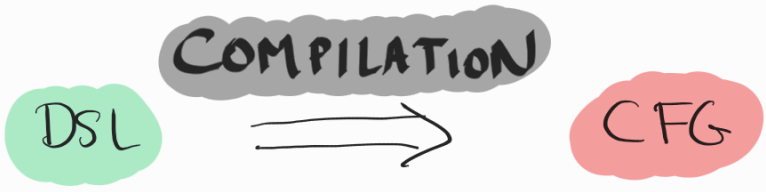
Example trees generated by CFG:





$\mathcal{L}(G)$: set of trees generated by the CFG G

SOME EXAMPLES OF REASONING



List of primitives

Add: int → int → int
Double: int → int
Halve: int → int
IfThenElse: bool → int → int → int
Even: int → bool
Equal: int → int → bool
0: int **False**: bool
1: int **True**: bool

type request:
int → int

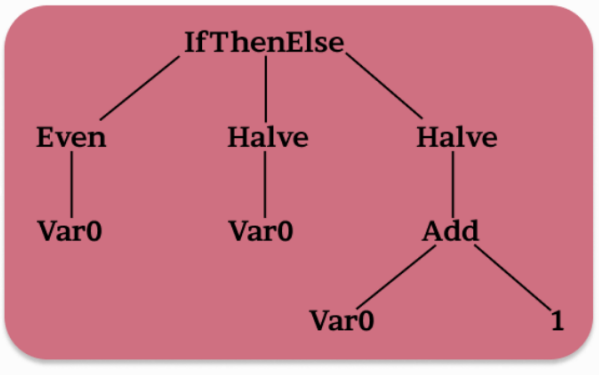
Compilation

Grammar

int → **Add**(int, int)
int → **Double**(int)
int → **Halve**(int)
int → **IfThenElse**(bool, int, int)
int → **0**
int → **1**
int → **Var0**
bool → **Even**(int)
bool → **Equal**(int, int)
bool → **True**
bool → **False**

Usual (Python-style) syntax

```
if Even(var0):
    Halve(var0)
else:
    Halve(Add var0 1)
```



Equivalent AST representation

TYPE-BASED COMPILATION

list of primitives

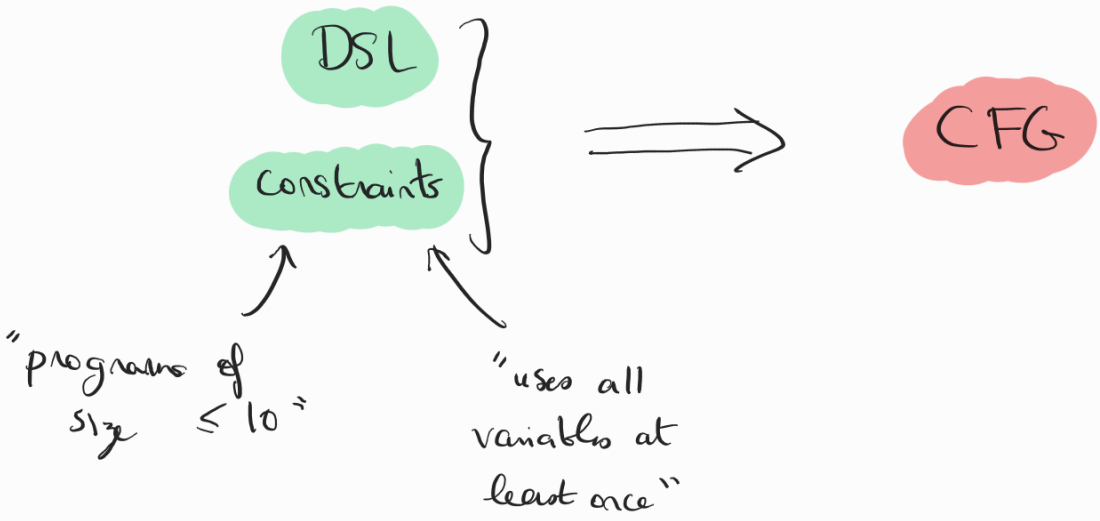
- Sort : list(int) → list(int)
- map : (t0 → t1) → list(t0) → list(t1)
- succ : int → int
- ⋮

CFG G

- list(int) → Sort (list(int))
- list(int) → map (int → int , list(int))
- list(int) → input
- int → int → succ ()
- ⋮

$\mathcal{L}(G)$ = all correctly typed programs

ADVANCED COMPILATION



RECENT WORK: SEMANTICAL PRUNING

(Théo Matricon)

automatically generating equations to reduce the DSL

Example equations:

- Halve(Double(P)) ≡ P
- Add(P, Add(Q, R)) ≡ Add(Add(P, Q), R)
- Add(P, Q) ≡ Add(Q, P)

Objective: construct a CFG representing programs up to equivalence

Unfortunately that's **impossible**: equivalence of programs is very quickly undecidable

Remark: Add(P,Q) ≡ Add(Q,P) is already too hard

But: we can quotient by "regular equivalence"

Key ingredient: well order on programs

PART II KNOWLEDGE MEETS SYNTHESIS

Smartfill doesn't know that Paris is in France

Bard doesn't know what James Web Telescope does

ChatGPT cannot add numbers

BUT Wikipedia and/or Calculators do !

So far, synthesis was either:

- purely syntactic (FlashFill)
- or looks for semantics in training data

RESEARCH QUESTION

How to use extrnal knowledge in Synthesis ?

RECENT WORK: WIKICODER (Theo Matricon)

a program synthesis tool which queries Wikipedia when it's lost

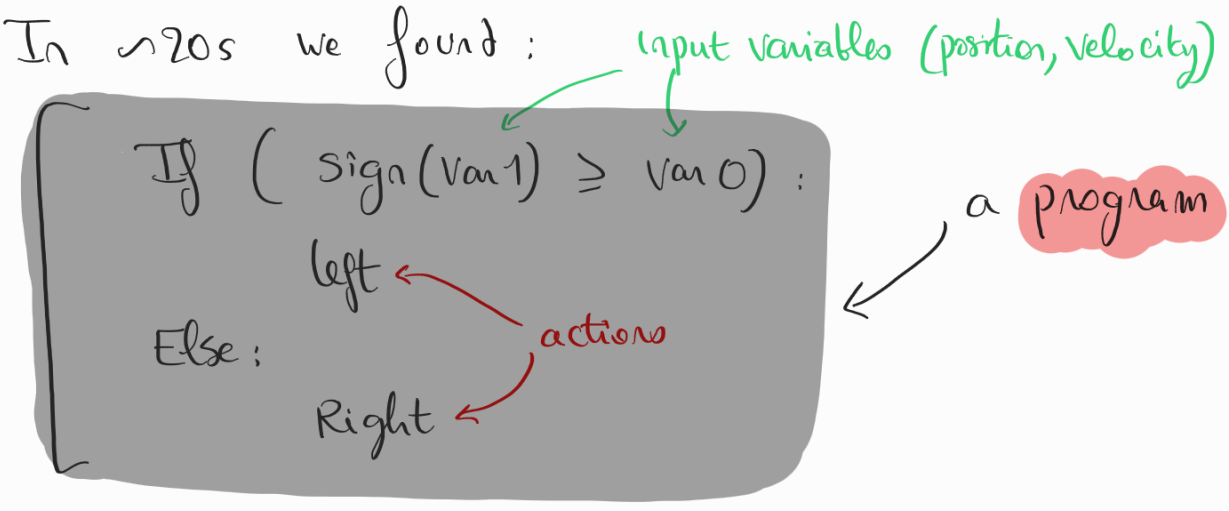
BUT: Wikipedia's structure is hard to use

A LOT to be done

PART III PROGRAMMATIC RL

Experiment on MountainCar

In ~20s we found :



Baseline Deep learning (DQN) : 5 mn for a neural policy ...

QUESTIONS

Q1 what is a good DSL for policies ?

Q2 how do we synthetise policies as programs ?

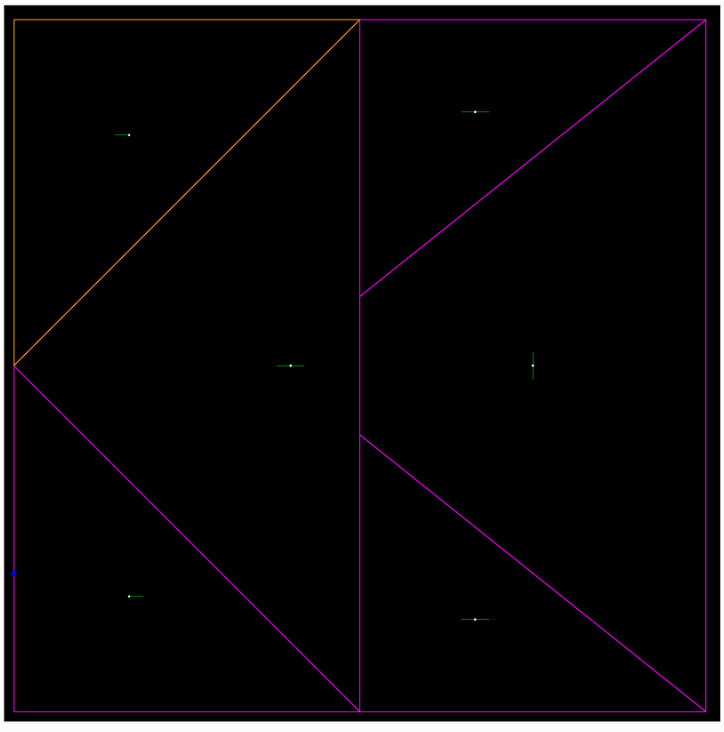
So far :

most answers to Q1 : decision trees or finite state controllers

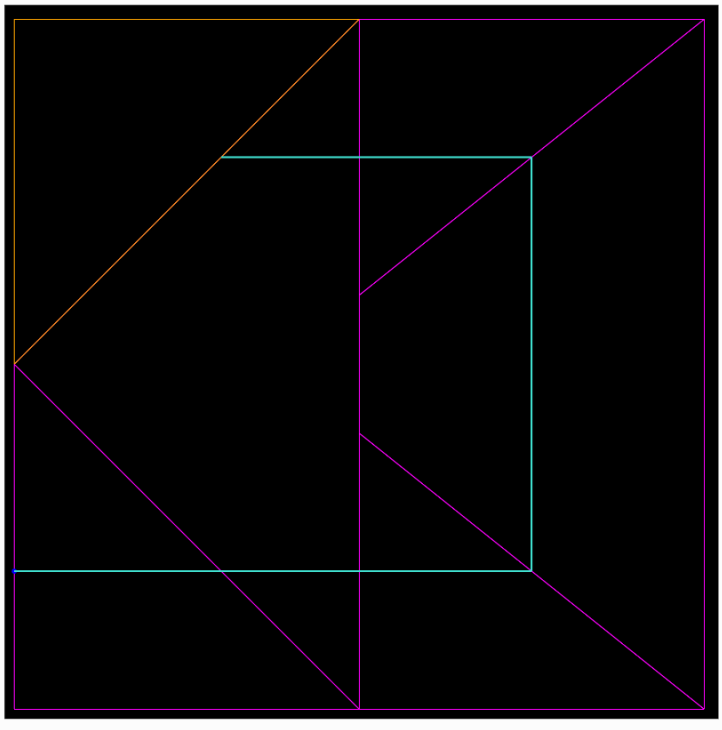
most answers to Q2 : extract from a neural policy

GRIDWORLDS (Guruprerana Shabadi)

problem

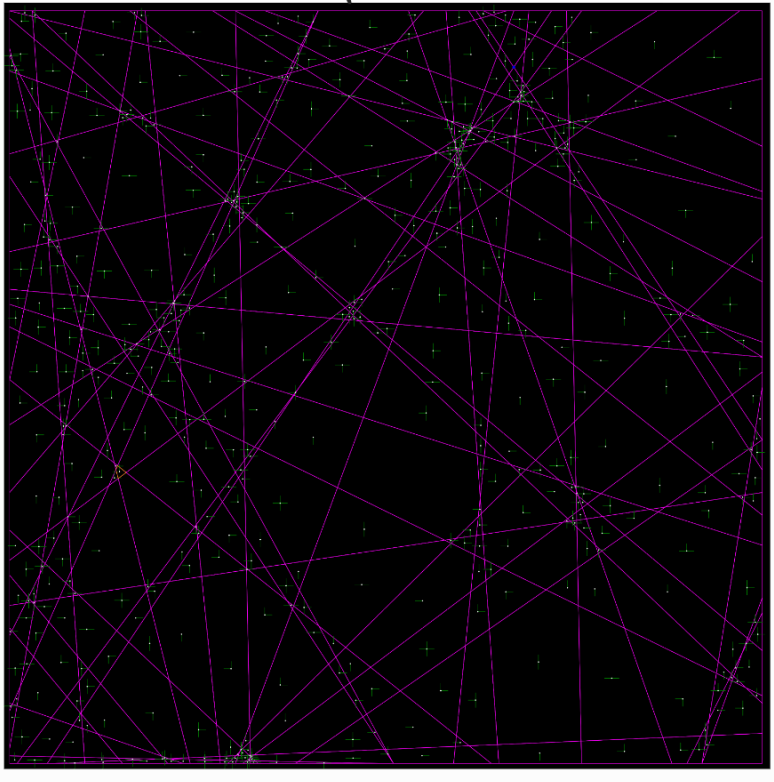


Solution

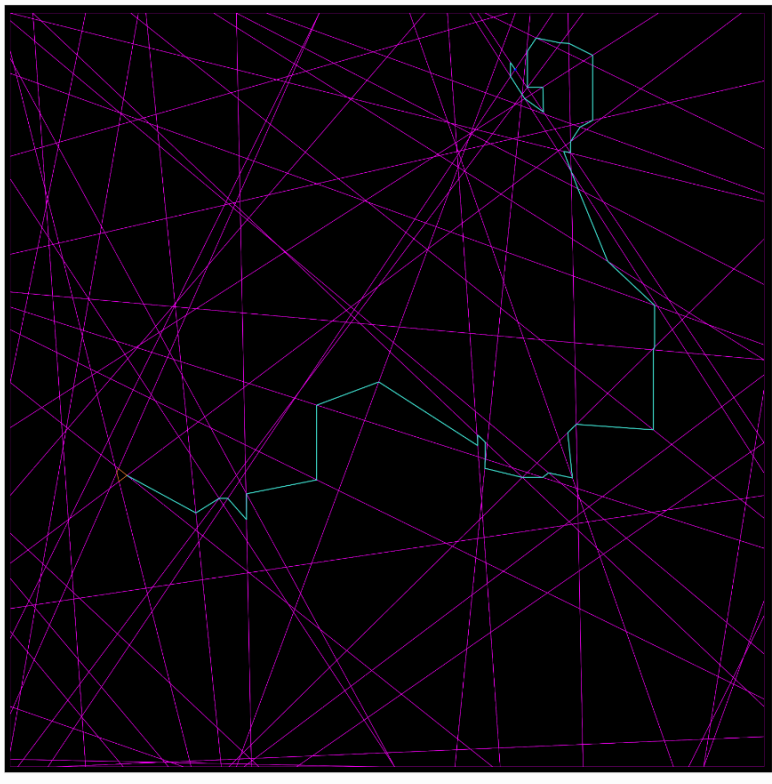


Key feature: the environment is generated as a PROLOG program

problem



Solution

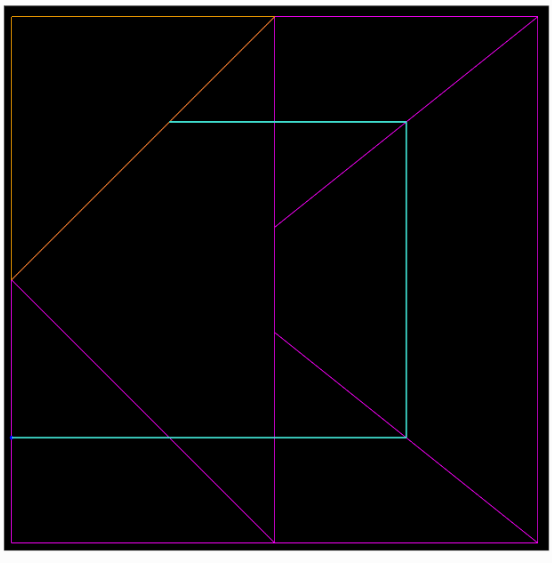


COMPARING TWO APPROACHES

(1) Solve as explicit MDP and return optimal policy as explicit function states → actions

(2) find a policy as program using backward analysis

Example



generated policy

```
Until([(0, 0), (0, 25)]):
  [(0, 25), (0, 0)] ->
    [(5, 20), (25, 0)], Preference: Neutral, Edge: [(0, 25), (25, 0)]

Until([(0, 25), (25, 0)]):
  [(0, 25), (25, 0)] ->
    [(25, 0), (25, 20)], Preference: Neutral, Edge: [(25, 20), (25, 0)]

Until([(25, 0), (25, 20)]):
  [(25, 20), (25, 0)] ->
    [(25, 20), (50, 0)], Preference: Neutral, Edge: [(25, 20), (50, 0)]

Until([(25, 20), (50, 0)]):
  [(25, 20), (50, 0)] ->
    [(25, 30), (50, 50)], Preference: Neutral, Edge: [(25, 30), (50, 50)]

Until([(25, 30), (50, 50)]):
  [(25, 30), (50, 50)] ->
    [(25, 30), (25, 50)], Preference: Neutral, Edge: [(25, 30), (25, 50)]

Until([(25, 30), (25, 50)]):
  [(25, 30), (25, 50)] ->
    [(0, 25), (25, 50)], Preference: Neutral, Edge: [(0, 25), (25, 50)]

Until([(0, 25), (25, 50)]):

  Win!
```

Summary of experimental results:

the program as policies are infinitely more succinct !

and Verifiable, explainable, readable, executable, ...

- environment as program : SMALL
- environment as explicit MDP : LARGE
- policy as program : SMALL
- policy as explicit S → A : LARGE

Motto:

Program you are, and to program you will return

environment

policy

JOIN US

PhD and Postdoc

We have open positions in Bordeaux !

both theory and practice