An Organic Diet for Python: devouring a Logic-based Language

Paul Tarau

University of North Texas

December 13, 2022

LPOP'2022



Motivation

- there are deep family resemblances between Prolog and Python
- they enable a smooth embedding in Python of a lightweight Prolog dialect: Natlog¹
- the resulting symbiosis is mutually beneficial:
 - Prolog benefits from the much wider Python deep learning ecosystem
 - Prolog enables neuro-symbolic inference and deep learning system configuration
 - Natlog's simplified syntax brings an easy to learn logic programming language to the ML practitioners

https://github.com/ptarau/natlog,install: "pip3 install natlog"



2/9

An illustration of the Python and Natlog symbiosis :-)

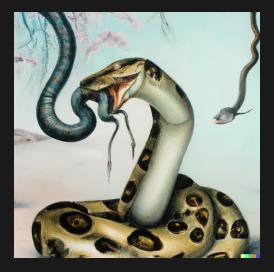


Figure: "Dali painting of big Python devouring small Natlog" (as seen by DALL.E)

Natlog: a Prolog with a lightweight syntax, embedded in Python

```
sibling of X S: parent of X P, parent of S P, distinct S X.
grand parent of X GP: parent of X P, parent of P GP.
```

```
ancestor of X A: parent of X P, parent or ancestor P A.
parent or ancestor P P.
parent or ancestor P A: ancestor of P A.
```

- terms are represented as nested tuples, all Python datatypes are directly reflected
- except variables: a lightweight class Var with a single value slot
- Natlog benefits from Python's memory management and no data conversion is needed
- Natlog is not slow: 227K LIPS when running under pypy3

High-level, intuitive data exchanges

"callables" (functions, classes, instances defining a call method in Python) are invoked from Natlog as in:

```
?- 'len (a b c) L.
ANSWER: {'L': 3}
```

generators are reflected in Natlog as alternative answers on backtracking.

```
?- ``range 1 4 X.
ANSWER: {'X': 1}
ANSWER: {'X': 2}
ANSWER: {'X': 3}
```

when Natlog is called from Python, variable assignments are yielded as Python dict objects

Reflecting metaprogramming constructs

• to conveniently access object and class attributes, Natlog implements setprop and getprop

```
setprop O K V: #setattr O K V.
getprop O K V : `getattr O K V.
```

elegant metaprogramming constructs on the two sides make language interoperation unusually easy

```
def meth call(o, f, xs):
   m = qetattr(o, f)
    return m(*xs)
```

method calls are supported via the Python function meth_call as in the following stack manipulation API:

```
stack S: 'list S. % note the use of the callable empty list constructor
push S X : \#meth \ call \ S \ append \ (X).
pop S X: `meth_call S pop () X.
```

Coroutining with yield and first-class logic engines

A first class logic engine is a language processor reflected through an API that allows its computations to be controlled interactively from another logic engine.

- this is very much the same thing as a programmer controlling Prolog's interactive toplevel loop: launch a new goal, ask for a new answer, interpret it, react to it
- the exception is that it is not the programmer, but it is the program that does it!
- first class logic engines ensure the full meta-level reflection of the execution algorithm
- in Natlog, we implement first class logic engines by exposing the interpreter to itself as a Python coroutine that transfers its answers one at a time via Python's yield operation

A few Examples of Natlog + Python apps

- Natlog can be used remotely as a Streamlit-based Web app
- Natlog is used as an orchestrator for JAX and Pytorch deep learning frameworks
- 3D Animations are easily built with Natlog and Vpython
- Natlog's DCGs are usable for prompt engineering



Figure: Natlog's DCGs as DALL.E prompt generators

Conclusion

- Natlog directly connects:
 - generators and backtracking,
 - nested tuples and terms
 - reflection and meta-interpretation
 - coroutines and first-class logic engines
- it enables logic-based language constructs to access the full power of the Python ecosystem
- two papers describing the details of Natlog:
 - https://github.com/ptarau/natlog/blob/main/ natlog/doc/natlog.pdf
 - * https://github.com/ptarau/natlog/blob/main/
 natlog/doc/natlog_deep.pdf
- next in line: GPT3 and ChatGPT² prompt engineering with Natlog's DGC grammars and its Neural Net orchestrator

