SAS-TO-PYTHON EXPLORATION

FALL 2017

PEOPLE

Di Xu, American Express

Jason Yi
Mingyu Zheng
Richy Chen
Abel Tadesse
Tony Jiang

CMC

Zach Dodds, HMC

OVERALL PROBLEM: TRANSPILER

A **source-to-source compiler**, **transcompiler** or **transpiler** is a type of compiler that takes the source code of a program written in one programming language as its input and produces the equivalent source code in another programming language.

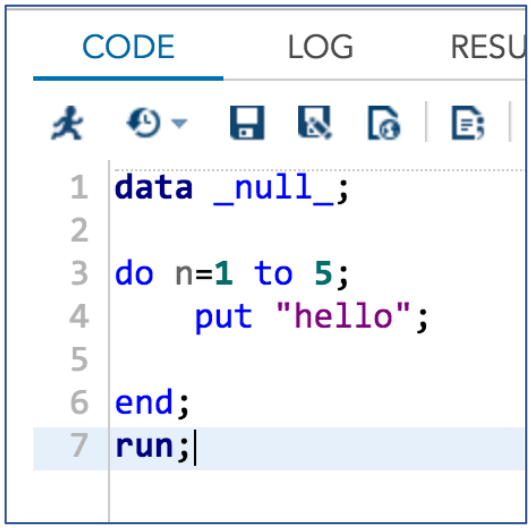
```
data _null_;

do n=1 to 5;
    put "hello";

end;
run;
```

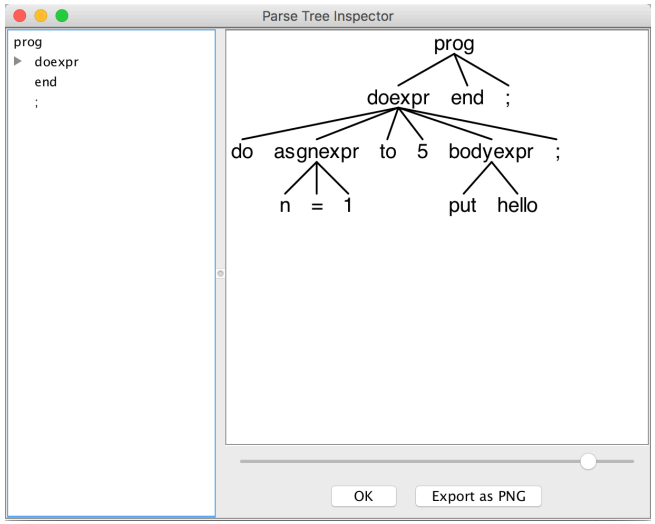
```
def f():
  for n in range(1, 6):
    print( "hello" )
```

SAS

PYTHON

TOKENIZING AND PARSING





SOURCE

PARSE TREE

TOOLS EXPLORED

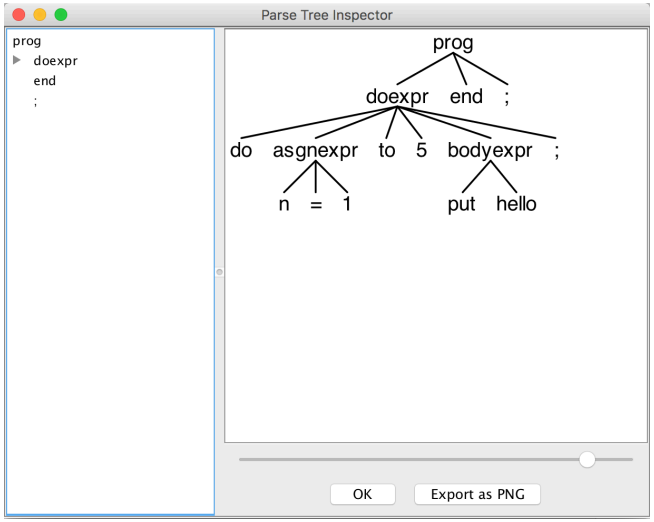
ANTLR v4

build passing | build passing | java 7+ | license BSD

ANTLR (ANother Tool for Language Recognition) is a powerful parser generator for reading, processing, executing, or translating structured text or binary files. It's widely used to build languages, tools, and frameworks. From a grammar, ANTLR generates a parser that can build parse trees and also generates a listener interface (or visitor) that makes it easy to respond to the recognition of phrases of interest.

PARSING THE PARSE TREE

PARSE TREE



S-EXPRESSION

```
In [12]: run parse_tree.py
Input S-expression:
 '(prog (doexpr do (asgnexpr n = 1) to 5
(bodyexpr put hello) ;) end ;)'


Parsed to Python:

['prog',
 ['doexpr',
  'do',
  ['asgnexpr', 'n', '=', 1],
  'to',
  5,
  ['bodyexpr', 'put', 'hello'],
  ';'],
 'end',
 ';']
```

PYTHON-READABLE PARSE TREE

TOOLS EXPLORED

6.2. `re` — Regular expression operations

```
term_regex = r'''(?mx)
    \s*(?:
        (?P<brackl>\()|
        (?P<brackr>\))|
        (?P<num>\-?\d+\.\d+|\-?\d+)|
        (?P<sq>"[^"]*")|
        (?P<s>[^(^)\s]+)
       )'''
```

GENERATING PYTHON

PYTHON PARSE TREE

```
['prog',
 ['doexpr',
  'do',
  ['asgnexpr', 'n', '=', 1],
  'to',
  5,
  ['bodyexpr', 'put', 'hello'],
  ';'],
 'end',
 ';']
```

PYTHON

```python
def f():
    for n in range(1, 6):
        print( "hello" )
```

TOOLS (TO BE) EXPLORED

LLVM Overview

LLVM began as a research project at the University of Illinois, with the goal of providing a modern, SSA-based compilation strategy capable of supporting both static and dynamic compilation of arbitrary programming languages. Since then, LLVM has grown to be an umbrella project consisting of a number of subprojects, many of which are being used in production by a wide variety of commercial and open source projects

RESULTS

SAS

INPUT

```
1          OPTIONS NONOTES
61
62         data _null_;
63
64         do n=1 to 5;
65
66           put "hello";
67
68         end;
69         run;

hello
hello
hello
hello
hello
```

EXECUTION

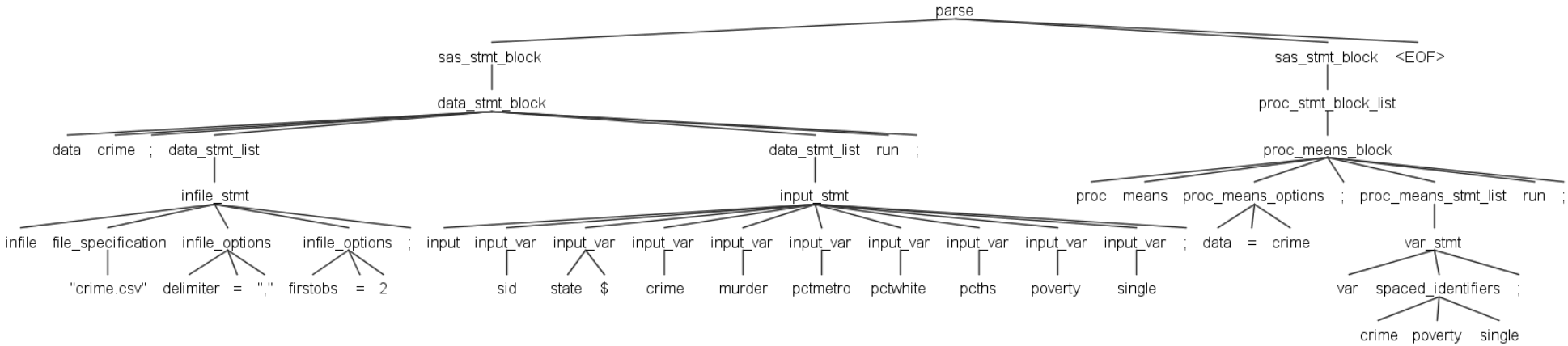
PYTHON

OUTPUT

```
def f():
  for n in range(1, 6):
    print( "hello" )


In [34]: f()
hello
hello
hello
hello
hello
```

EXECUTION

Largest SAS grammar subset found so far:



https://github.com/xueqilsj/sas-grammar

EPILOGUE

- UNDERSTANDING OF THE PROBLEM & APPROACHES
- EXPLORATION OF AVAILABLE TOOLS ANTLR4, LLVM, SAS, Python
- END-TO-END PROTOTYPE
- BIG-PICTURE DISCUSSIONS

  Chris Stone, Ben Wiedermann, Melissa O'Neill (HMC CS faculty)

  ANTLR4 allows add'l code; recursive descent also possible

  80/20 + iterate. Possible exploration of initial scriptset…

- THANK YOU, DI – PERHAPS REVISIT, SPRING '18