

# Flattening Abstract Syntax Tree for Efficiency <sup>[1]</sup>

Yijun Yu, The Open University, U.K. <http://mcs.open.ac.uk/yy66>

## Serendipity

Software engineering tools exchange code representations through serialised **abstract syntax trees**.

*Surprisingly*, hierarchical representation is **NOT** the fastest to exchange tree structures.

## Requirements

Speed up the processing of code whilst preserving the **equivalence** to hierarchies in an **efficient** form.

## Design Rationale

- 1) Save AST as a *flat* 1D array by converting tree pointers into integer offsets;
- 2) Flattened AST can be more efficient to access by programming tools through APIs.

## Features

- ✓ **fast**
- ✓ **language-agnostic**
- ✓ **ANTLR4 grammar**
- ✓ **microservice ready**
- ✓ **containerised**
- ✓ **IDE friendly ?!**
- ✓ **human readable ?!**

## Example Usage

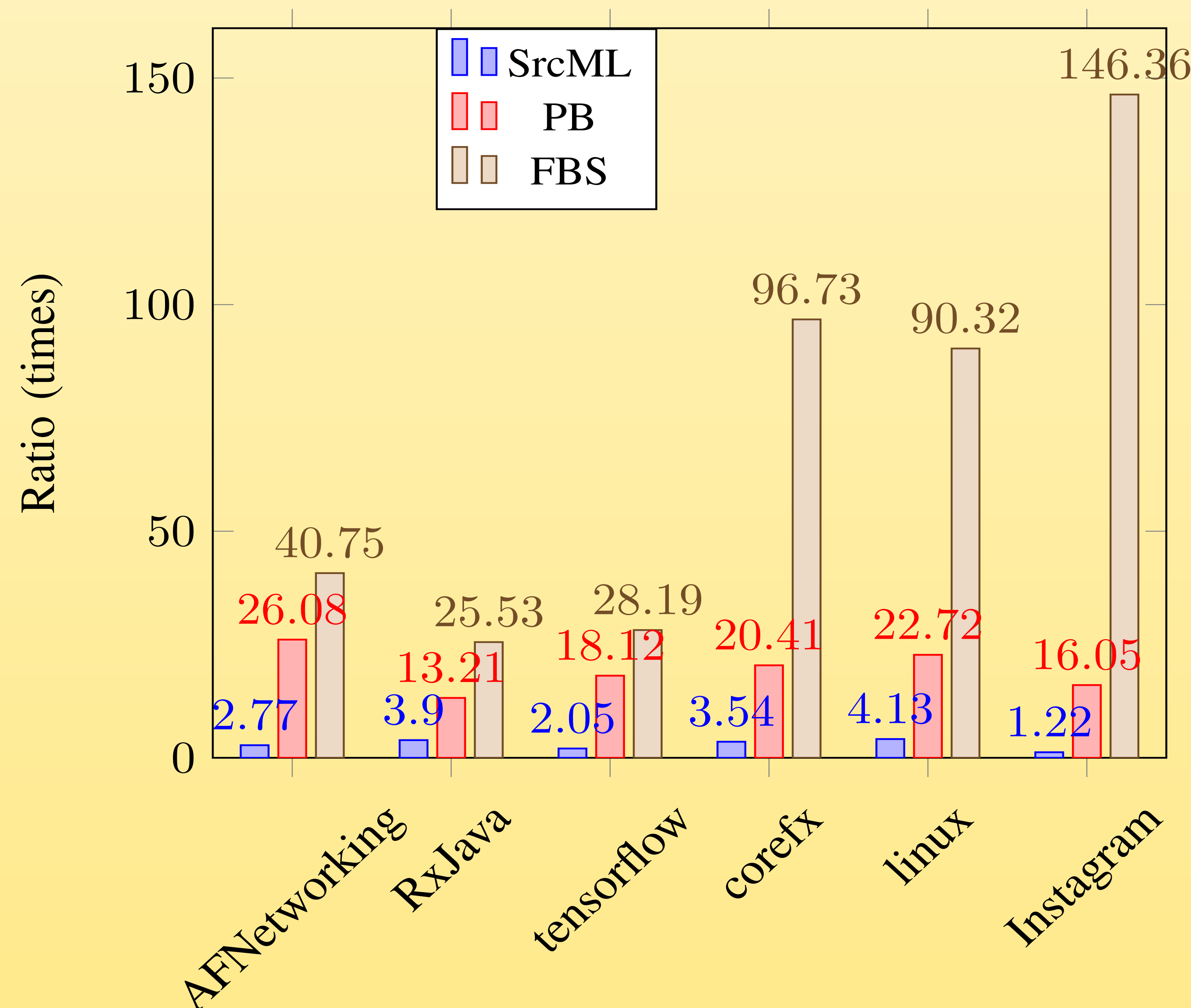
```
# print the command line options and arguments
alias fast="docker run -v $PWD:/e yijun/fast"
# convert a C++ code into protobuf representation
fast foo.cc foo.pb
# convert a Java code into flatbuffers representation
fast bar.java bar.fbs
# convert a flatbuffers representation back to C#
fast moo.fbs moo.cs
# slice a program
fast -S -G foo.java foo.fbs
# diff two programs
fast -D v1.java v2.java
```

## Applications

- ✓ **Parsing: 100x faster for 7 popular projects**
  - ✓ **Big Code Deep Learning [2]** (see a demo below)
- ✓ **Slicing**
  - ✓ **2.5x faster than srcSlicer [3]**
- ✓ **Diffing [4]: 20x faster**
  - ✓ **Bug localisation (ConCodeSe) [5]**
- ✓ **Extending IDE**
  - ✓ **Visual Studio Code**
  - ✓ **Browser-based IDE**
- **Search for gravitational lens**

## Evaluation Results

## Live Demo



Parsing flattened AST is 100x faster on a benchmark of 29 projects of 6 programming languages: ObjectiveC, Java, C++, C#, C, Smali). A total of 298,312,076 LOC.



Fig. 1 shows 6 of them, one for each programming language.

## References

- 1) Yijun Yu. "fAST: Flattening Abstract Syntax Trees for Efficiency". In: 41st ACM/IEEE International Conference on Software Engineering, 25-31 May 2019, Montreal, Canada, ACM and IEEE.
- 2) Bui D. Q. Nghi, Yijun Yu, Lingxiao Jiang: "Bilateral Dependency Neural Networks for Cross-Language Algorithm Classification". In the 26th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Hangzhou, China, February 24-27, 2019: 422-433.
- 3) Hakam W. Alomari, Michael L. Collard, Jonathan I. Maletic, Nohu Alhindawi and Omar Meqdadi. "srcSlice: very efficient and scalable forward static slicing". Software: Evolution and Process, 26(11):931-961, November 2014.
- 4) Yijun Yu, Thein Thun Tun, and Bashar Nuseibeh, "Specifying and detecting meaningful changes in programs," In: Proc. of the 26th IEEE/ACM Conference on Automated Software Engineering, pp. 273-282, 2011.
- 5) Tezcan Dilshener, Michel Wermelinger, Yijun Yu: "Locating bugs without looking back". Automated Software Engineering 25(3): 383-434 (2018)