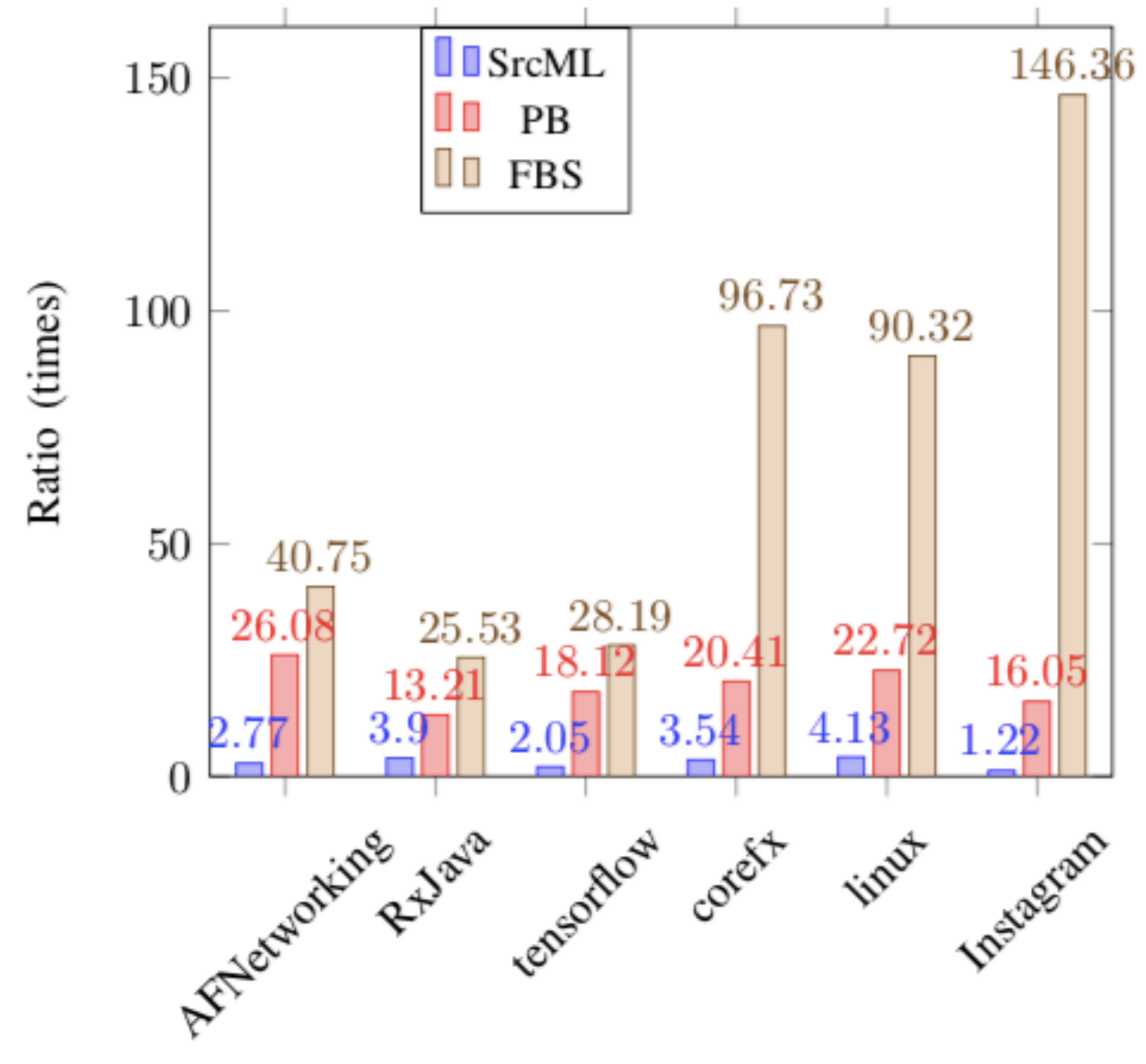



Flattening Abstract Syntax Tree for Efficiency ^[1]

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

Serendipity	Requirements	Design Rationale	Features
<p>Software engineering tools exchange code representations through serialised abstract syntax trees.</p> <p><i>Surprisingly</i>, hierarchical representation is NOT the fastest to exchange tree structures.</p>	<p>Speed up the processing of code whilst preserving the equivalence to hierarchies in an efficient form.</p>	<p>1) Save AST as a <i>flat</i> 1D array by converting tree pointers into integer offsets;</p> <p>2) Flattened AST can be more efficient to access by programming tools through APIs.</p>	<ul style="list-style-type: none"> ✓ <i>fast</i> ✓ <i>language-agnostic</i> <ul style="list-style-type: none"> ✓ <i>ANTLR4 grammar</i> ✓ <i>microservice ready</i> <ul style="list-style-type: none"> ✓ <i>containerised</i> ✓ <i>IDE friendly ?!</i> ✓ <i>human readable ?!</i>
Example Usage	Applications		
<pre># 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</pre>	<ul style="list-style-type: none"> ✓ Parsing: 100x faster for 7 popular projects <ul style="list-style-type: none"> ✓ Big Code Deep Learning [2] (see a demo below) ✓ Slicing <ul style="list-style-type: none"> ✓ 2.5x faster than srcSlicer [3] ✓ Diffing [4]: 20x faster <ul style="list-style-type: none"> ✓ Bug localisation (ConCodeSe) [5] ✓ Extending IDE <ul style="list-style-type: none"> ✓ Visual Studio Code ✓ Browser-based IDE 		

Evaluation Results	Live Demo
 <p>Fig. 1. Speed up parsing by loading respectively SrcML, PB, and FBS instead</p>	<div> <p>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.</p>  </div> <div> <p>← Fig. 1 shows 6 of them, one for each programming language.</p> </div>

References

- 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.
- 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.
- Hakam W. Alomari, Michael L. Collard, Jonathan I. Maletic, Nouh Alhindawi and Omar Meqdadi. **"srcSlice: very efficient and scalable forward static slicing"**. *Software: Evolution and Process*, 26(11):931-961, November 2014.
- 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.
- Tezcan Dilshener, Michel Wermelinger, Yijun Yu: **"Locating bugs without looking back"**. *Automated Software Engineering* 25(3): 383-434 (2018)