Generating Customised Control-Flow Graphs for Legacy Languages with Semi-Parsing

Céline Deknop
UCLouvain, ICTEAM institute, Belgium
Raincode Labs, Brussels, Belgium

Johan Fabry
Raincode Labs, Brussels, Belgium

Kim Mens
UCLouvain, ICTEAM institute, Belgium

Vadim Zaytsev
Formal Methods & Tools, UTwente, The Netherlands

Meta Data & Stats

Conference:	ICSME
Track:	Software Maintenance and Evolution
Year:	2022
Number of Authors:	4
Citations:	0
Pages (PDF):	10
Figures:	10
References:	42
Formals:	6 definitions

What is the Study About?

To propose a tool and underlying technique that uses semi-parsing to extract control flow graphs from legacy source code (i.e., COBOL).

Industrial goals:

- The effort necessary to create a full parser from scratch is rather large (a famous expert quote puts it at 2–3 years)
- Legacy code modernization
- Support embedded languages

Table of Content

Approach: Parser configuration

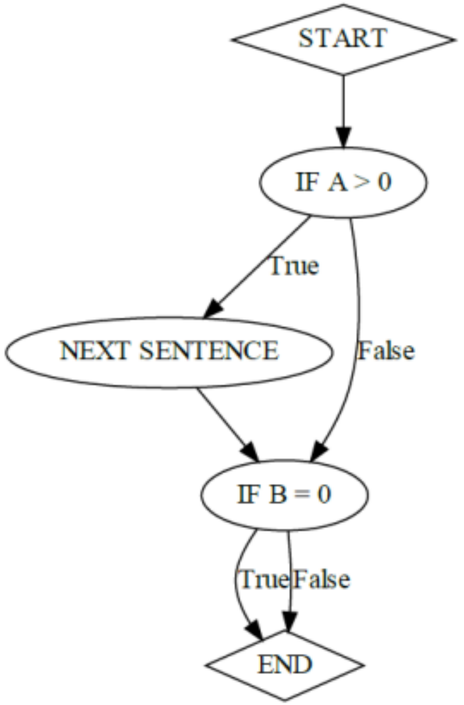
Listing 1: Extract of our parser configuration file for COBOL.

```
1 conditions = [
2     {"start": r"\sIF(\s)+",
3      "condition_delimiter": r"\sTHEN(\s)+",
4      "mandatory_delimiter": False,
5      "single_branch": r"ELSE\s", "end": r"\sEND-IF"},
6     ...
7     ]
```

Approach: Example

Listing 2: A small handcrafted COBOL code example

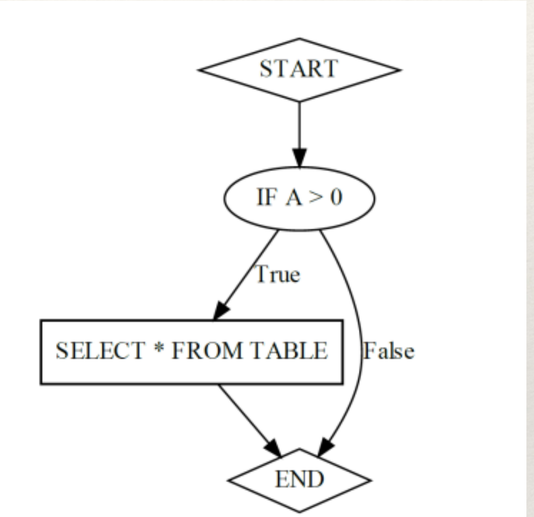
```
1        IDENTIFICATION DIVISION.
2        PROGRAM-ID. Example1.
3        ENVIRONMENT DIVISION.
4        DATA DIVISION.
5        PROCEDURE DIVISION.
6            IF A > 0
7              NEXT SENTENCE
8            ELSE
9              DISPLAY "First if branch"
10           END-IF.
11           IF B = 0
12             DISPLAY "Second if"
13           END-IF.
```



Approach: Example with Embedded SQL Code

Listing 3: A small COBOL program with embedded SQL code

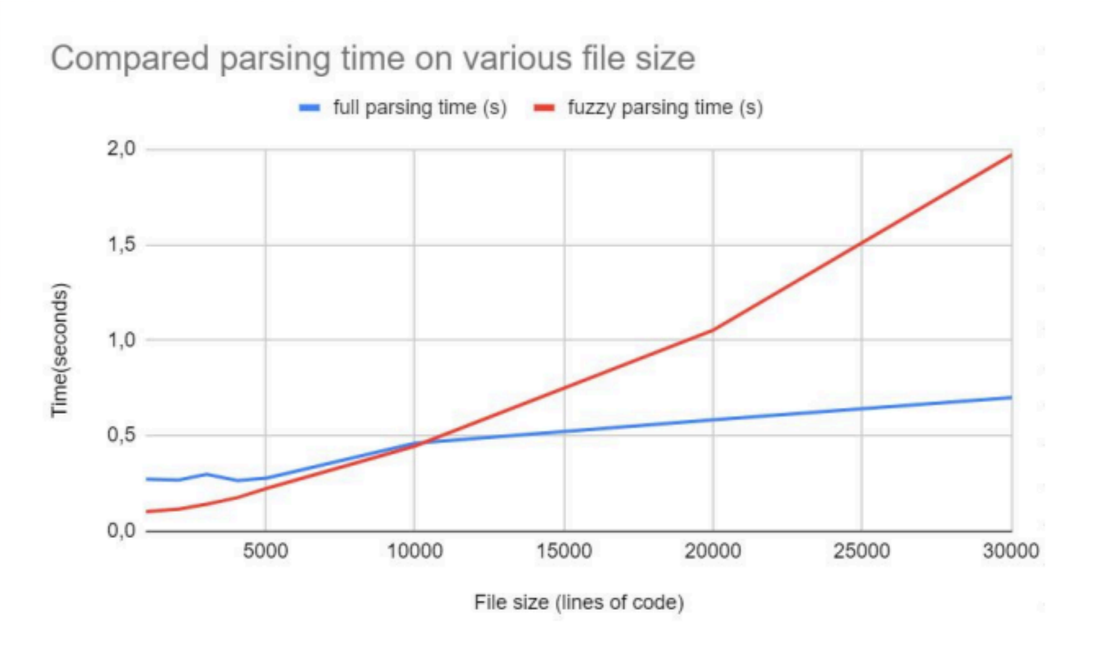
```
1        IDENTIFICATION DIVISION.
2        PROGRAM-ID. Example2.
3        ENVIRONMENT DIVISION.
4        DATA DIVISION.
5        PROCEDURE DIVISION.
6            IF A > 0
7              EXEC SQL SELECT * FROM TABLE END-EXEC
8            END-IF.
```

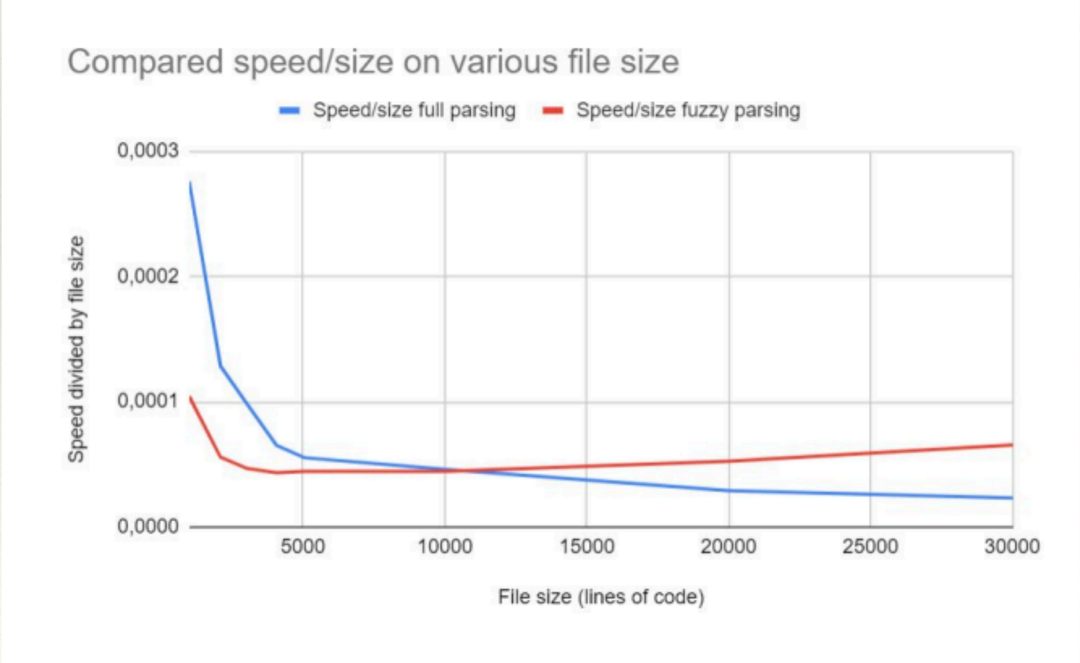


Evaluation

	Av. file size (lines)	Fuzzy total time (s)	Full total time (s)	Speedup fuzzy w.r.t. full
Test files (32)	15	0.03	9.35	312
NIST (410)	781	37,45	127.87	3.4
Use case post-migration (3014)	1715	273.01	959.12	3.5
Use case pre-migration (3014)	2675	1272.29	947.53	0.7

TABLE I: Execution times on all file sets





Discussion

- *Efficiency and Correctness*
- *Modularity*
- *Dialect agnosticism*
- *Ability to handle partial or non-compilable code*

What is good/interesing about the paper

- *Structured and easy to read*
- *Detailed example*

What could be better

- *A lot of attention to simple definition*
- *Too simple examples*
- *Raincode Labs (https://www.raincodelabs.com) mentioned 26 times*
- *Confidentian quantitve research*
- *Missing Background and Related Worκ*

8. Conclusion