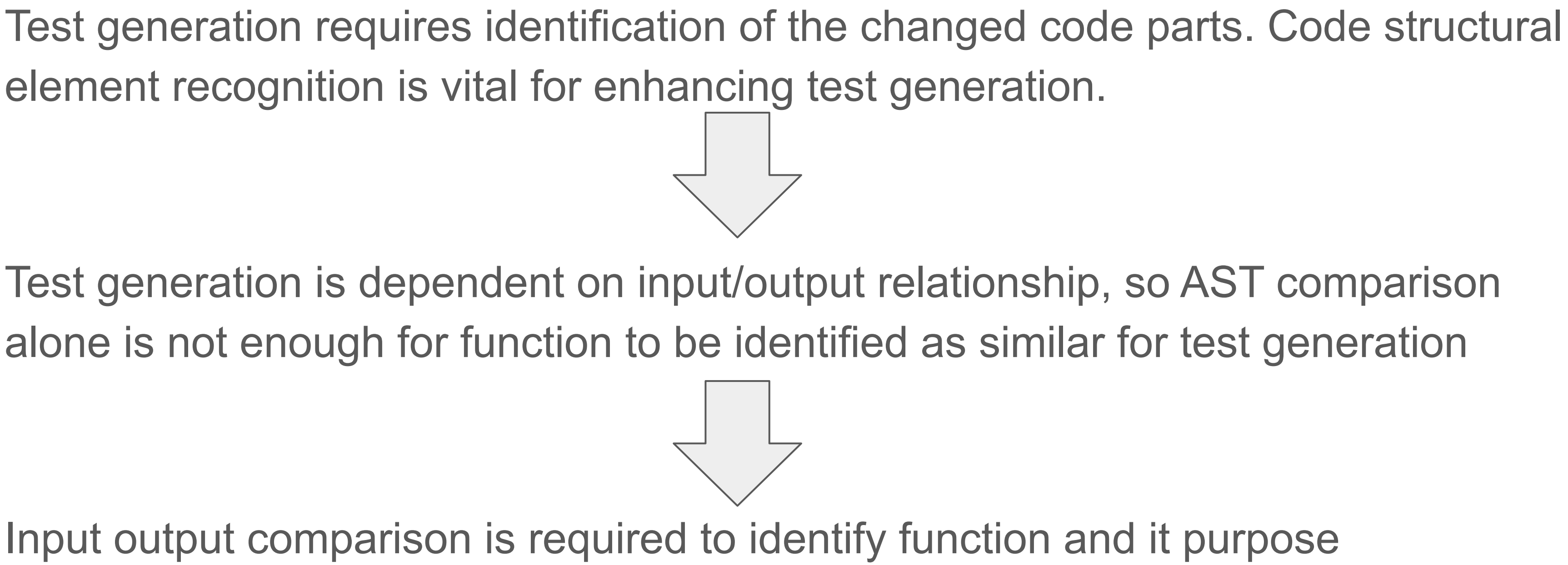
Classification of software structural elements in test generation task

PhD student - Andrii Franko
Supervisor - Roman Bazylevych

Agenda

1. Introduction
2. Methodology
3. Experiments results
4. Conclusion

Introduction



Test generation requires identification of the changed code parts. Code structural element recognition is vital for enhancing test generation.

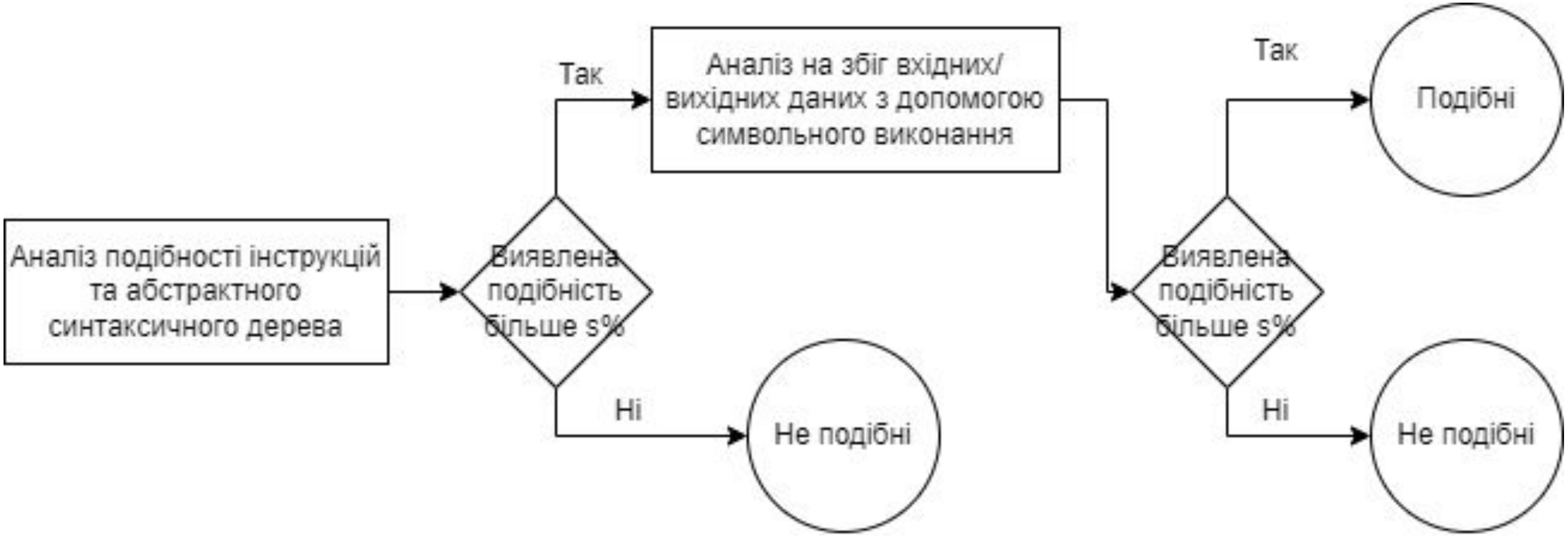
Test generation is dependent on input/output relationship, so AST comparison alone is not enough for function to be identified as similar for test generation

Input output comparison is required to identify function and it purpose

Methodology

- Take into account function characteristic that are vital for test generation
- Design method that will take into account vital properties

Results



Аналіз подібності інструкцій та абстрактного синтаксичного дерева

Виявлена подібність більше s%

Так

Аналіз на збіг вхідних/ вихідних даних з допомогою символьного виконання

Ні

Не подібні

Виявлена подібність більше s%

Так

Подібні

Ні

Не подібні

Results

Algorithm would work in cases:

1) Additional set of condition added to function which provides handling for additional cases (modified switch/case)
2) Distinction between small utility function (getters and setters of different properties)
3) Distinction between object created by using common design patterns but which have different roles

Results

The algorithm will fail to identify

Function which have similar structure but use different set of const data. In this case test should be update because the input/output relationship has been changed

	Function	Function
	Calculate CRC16 with polynom1	Calculate CRC16 with polynom2
AST comparison	100% similarity	100% similarity
Symbolic execution input/output comparison	0% similarity	0% similarity

Results

The algorithm will fail to identify

Function that were fully rewritten. In this case a new test need to be generated because the structure has changed.

	Function	Function
	Calculate Fibonacci numbers iterative implementation	Calculate Fibonacci numbers recursive implementation
AST comparison	0% similarity	0% similarity
Symbolic execution input/output comparison	100% similarity	100% similarity

Conclusion

- Proposed algorithm may potentially help in building advanced test generation systems. It should be further enhanced.
- It may help to balance cases when AST comparison faces problems due to small function size, or common design pattern usage
- It may detect changes in function when the const data was changed without AST changes
- It should be further tested on appropriate data sets