Program Decomposition and Translation with Static Analysis

Ali Reza Ibrahimzada (alirezai@illinois.edu)

University of Illinois Urbana-Champaign, IL, USA

Code Translation

Code translation converts source code from one programming language to another (Java → Python).

Java Code

```java
public class Main {
    public static void main(String[] args) {
        int max = 5;
        for (int i = 0; i < max; i++) {
            System.out.println(i);
        }
    }
}
```

Translation

Python Code

```python
max = 5
for i in range(max):
    print(i)
```

Figure 1. Translation of a code snippet from Java to Python.

Problem Significance & Research Gap

Use cases of code translation in industry include:

- **Application Modernization**: Upgrading the underlying languages of legacy applications (COBOL).
- **Architecture Migration**: Migrating monolithic software architecture to cloud-native ones.
- **Performance Upgrade**: Translation can be used to improve the performance of existing software.

Existing techniques for code translation broadly fall into the following categories:

- **Transpilers**: Tools like C2Rust [1] lack native target language features (e.g., memory safety).
- **Learning-based Techniques**: These techniques use parallel training data to learn different code features (e.g., mppSMT [2], TransCoder [4]).
- **LLMs**: LLMs have excelled in generative software engineering tasks, ***generalizing*** well and generating more ***natural code***. In this work, we aim at answering the question ***"Can we effectively use LLMs for translating repository-level projects?".***

Empirical Evaluation of LLMs [3]

We systematically evaluated state-of-the-art LLMs on multiple benchmarks and real-world projects.

Dataset	Source Language	% Successful Translations				
		CodeGen	CodeGeeX	StarCoder	GPT-4	Llama 2
CodeNet	C	23.4%	14.9%	42.0%	83.0%	14.9%
	C++	14.0%	3.6%	39.1%	80.0%	9.5%
	Go	14.3%	5.9%	42.0%	85.5%	16.9%
	Java	21.3%	10.3%	30.3%	81.3%	13.9%
	Python	17.5%	7.3%	33.3%	79.9%	11.0%
AVATAR	Java	8.1%	1.8%	11.9%	70.8%	1.8%
	Python	3.8%	1.6%	14.2%	52.2%	4.7%
Evalplus	Python	16.5%	3.7%	22.0%	79.3%	1.2%
Commons-CLI	Java	0.0%	0.0%	0.0%	13.6%	0.0%
Click	Python	0.0%	0.0%	0.0%	0.0%	0.0%
Average	-	8.1%	2.8%	14.5%	47.3%	3.5%

Table 1. Empirical evaluation of LLMs on crafted and real-world benchmarks.

What Causes Unsuccessful Translations?

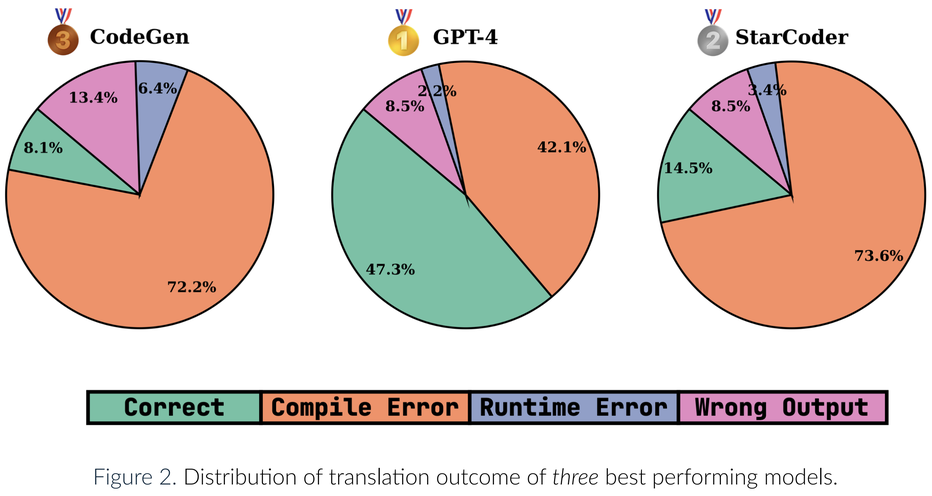


Figure 2. Distribution of translation outcome of *three* best performing models.

Why LLMs Struggle with Real-world Projects?

Real-world projects pose more complex challenges for code translation, such as handling method overloading, exceptions, inheritance relations, etc. LLMs also suffer capturing the proper context due to inter- and intra-procedural dependencies.

Source Code (Java)

```java
public Options addOption(final Option opt) { ... }
public Options addOption(final String opt, final boolean hasArg,
                         final String desc) { ... }
public Options addOption(final String opt, final String desc) { ... }
```

Translated Code (Python)

```python
def add_option(self, opt): ...
def add_option_arg(self, opt, hasArg, desc): ...
def add_option_desc(self, opt, desc): ...
```

Figure 3. An illustrative example of unsuccessful translation generated by GPT-4 caused by method-overloading.

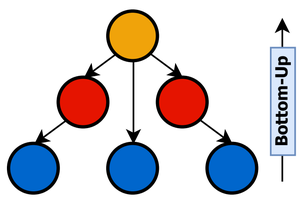
Method-level Program Decomposition

We propose ***method-level program decomposition*** to address the out-of-context issue with LLMs. To support our claim, we use ***static analysis*** to analyze $60K$ methods and study if it is feasible to decompose programs based on methods.

Project	% Files >2K Tokens	# Methods	Avg. Tokens / Method	% Methods >2K Tokens	% 2K Context
bcel	11.29%	4,094	70.42	0.15%	3.44%
beanutils	29.84%	2,675	107.09	0.07%	5.23%
cli	30.77%	582	97.91	0.17%	4.78%
codec	48.30%	1,788	189.29	0.84%	9.24%
collections	19.34%	6,354	74.37	0.02%	3.63%
csv	27.08%	871	102.53	0.11%	5.01%
daemon	27.78%	60	108.63	0.00%	5.30%
dbcp	38.52%	3,622	63.02	0.03%	3.08%
dbutils	13.54%	869	61.44	0.00%	3.00%
fileupload	16.67%	401	77.8	0.00%	3.80%
geometry	39.13%	6,615	124.93	0.03%	6.10%
imaging	14.78%	2530	143.71	0.20%	7.02%
io	22.07%	5,957	77.94	0.07%	3.81%
jexl	25.70%	3,967	109.37	0.20%	5.34%
lang	40.34%	9,134	103.33	0.12%	5.05%
net	23.83%	2,023	98.22	0.15%	4.80%
pool	22.68%	1,377	94.13	0.00%	4.60%
rng	36.60%	3,245	139.69	0.52%	6.82%
text	28.32%	2,712	99.85	0.04%	4.88%
validator	38.00%	1,181	147.42	0.17%	7.20%
Average	**27.73%**	**3002.85**	**104.55**	**0.14%**	**5.11%**

Table 2. The effect of method-level program decomposition on a 2K context window model. The analysis has been done on 20 well-known Apache Commons projects.

Call Graph-based Program Translation



Decomposition Technique	# Source Files	# Out-of-Context Inputs	% Context Occupied
No Decomposition	22	8	36%
Method Decomposition	22	0	3%

Table 3. The effectiveness of method-level decomposition when translating Apache Commons CLI using its Call Graph.

CG-aware Prompt Engineering for Real-World Projects

Can we dynamically craft prompts and enforce dependencies in real-world projects? How effective is providing more contextual information to LLMs?

```
You are an expert $SRC_LANG and $TRG_LANG programmer.

$ICL_EXAMPLE
-----------------------------------------------------------
Translate the following $SRC_LANG method to $TRG_LANG like the
example above:

$SRC_CODE
-----------------------------------------------------------
The following fields(s) are used in the method body and are already
translated:

$DEPENDENT_FIELDS
-----------------------------------------------------------
The following method(s) are called in the method body and are
already translated:

$DEPENDENT_METHODS
-----------------------------------------------------------
$TRG_CODE: // Generated Code
```

Figure 4. Fine-grained prompt template for translating decomposed fragments.

Check our work on Code Translation!

SRC Paper | ICSE Paper | Leaderboard | Code

References

[1] C2rust transpiler. `https://github.com/immunant/c2rust`, 2023.

[2] Anh Tuan Nguyen, Tung Thanh Nguyen, and Tien N Nguyen. Divide-and-conquer approach for multi-phase statistical migration for source code (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 585–596. IEEE, 2015.

[3] Rangeet Pan, Ali Reza Ibrahimzada, Rahul Krishna, Divya Sankar, Lambert Pouguem Wassi, Michele Merler, Boris Sobolev, Raju Pavuluri, Saurabh Sinha, and Reyhaneh Jabbarvand. Lost in translation: A study of bugs introduced by large language models while translating code. In *ACM/IEEE International Conference on Software Engineering*, 2024.

[4] Baptiste Roziere, Marie-Anne Lachaux, Lowik Chanussot, and Guillaume Lample. Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, 33:20601–20611, 2020.