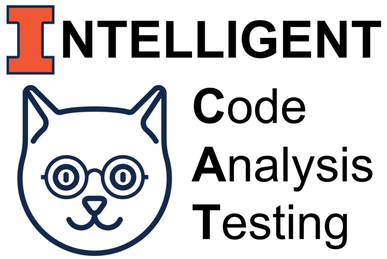
International Conference on the Foundations of Software Engineering (FSE)

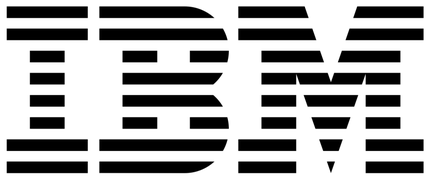
AlphaTrans: A Neuro-Symbolic Compositional Approach for Repository-Level Code Translation and Validation

Ali Reza Ibrahimzada[1], Kaiyao Ke[1], Mrigank Pawagi[3], Muhammad Salman Abid[4], Rangeet Pan[2], Saurabh Sinha[2], Reyhaneh Jabbarvand[1]

[1]University of Illinois Urbana-Champaign [2]IBM Research [3]Indian Institute of Science [4]Cornell University

June 23rd, 2025





Code Translation

- Code Translation converts source code from one programming language (PL) to another

Java Code

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

Python Code

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

- Code translation is very challenging because it requires understanding syntax and semantics of the code in both source and target language

Research Gap

Translation

Validation

- Demand for idiomatic, repository-level automated code translation
  - Transpilation scales to repository-level, but produces unreadable, non-idiomatic code
  - LLM-based techniques produce readable and idiomatic code, but do not scale

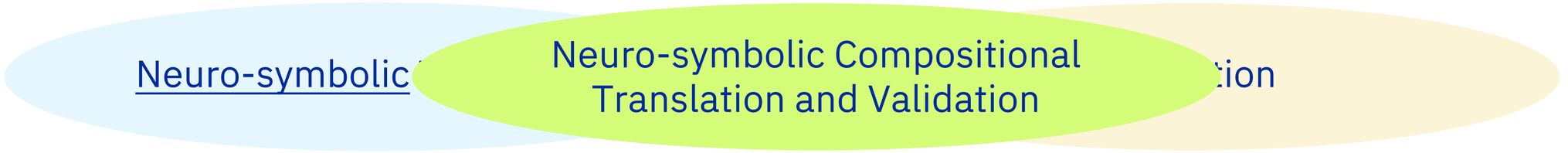
- Translation and validation as separate processes
  - Translation first, validation next: costly and may involve redoing entire translations
  - Hard to debug (root-causing where the inconsistency come from and patch the translation approach)

Bridging the Research Gap

Neuro-symbolic Translation

- Demand for idiomatic, repository-level automated code translation
  - Combining the powers of program analysis and generative AI

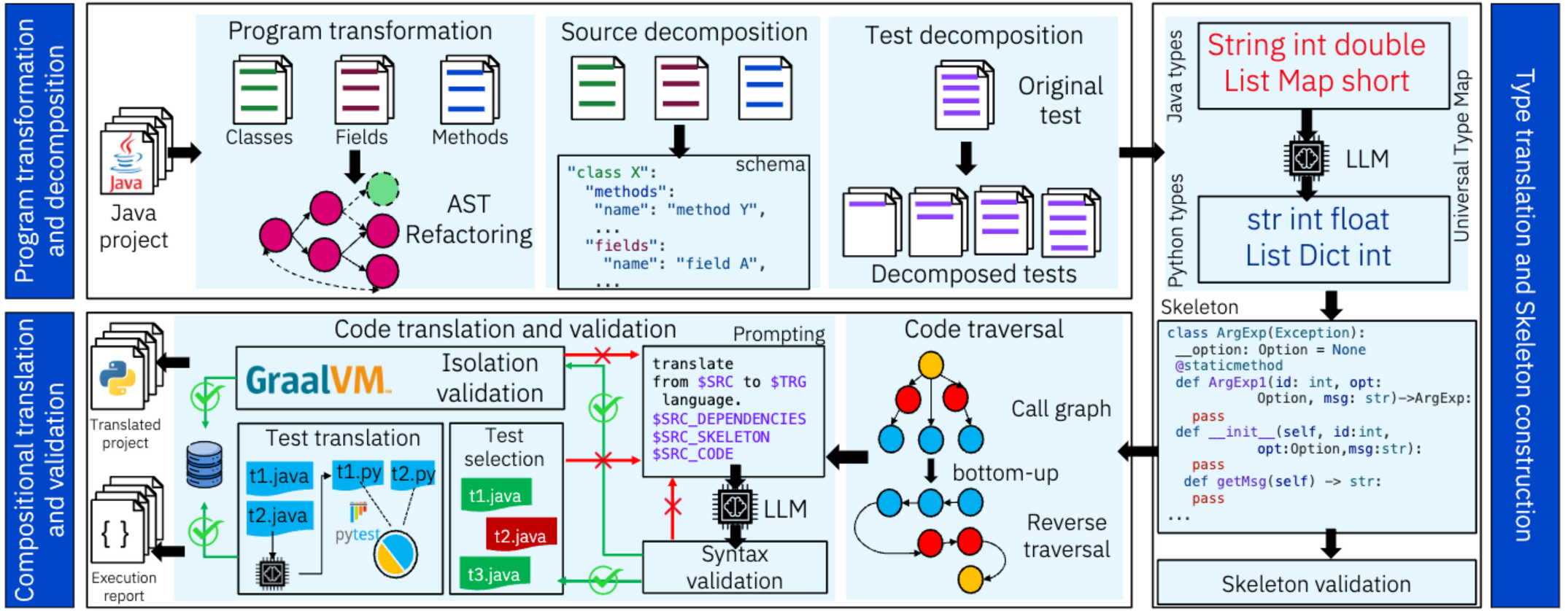
Bridging the Research Gap



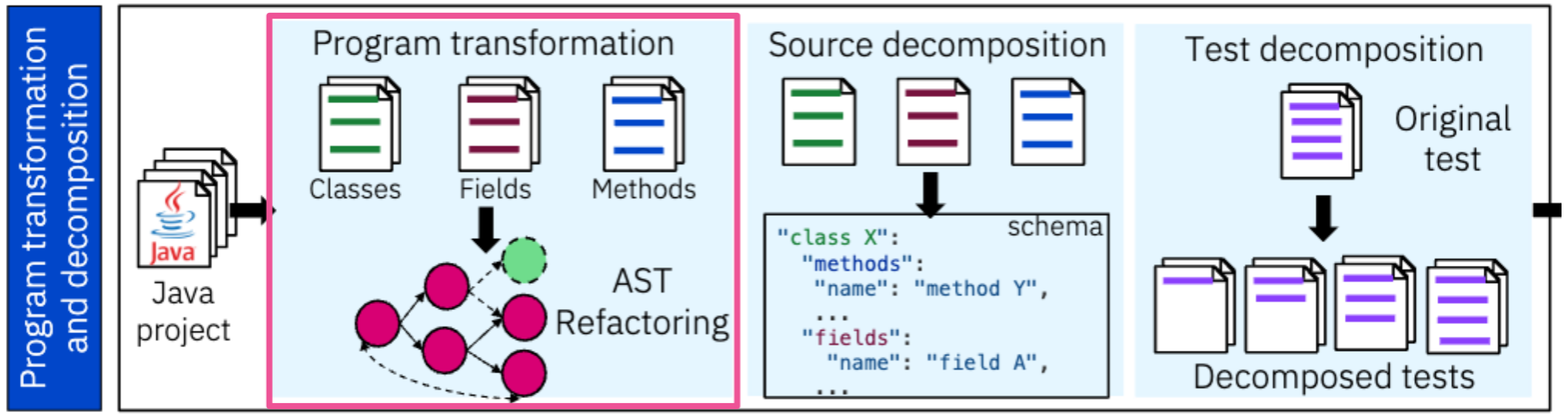
- Demand for idiomatic, repository-level automated code translation
  - Combining the powers of program analysis and generative AI

- Translation and validation as separate processes
  - Unifying translation and validation to promptly check compilation success and functional equivalence amid translation

AlphaTrans

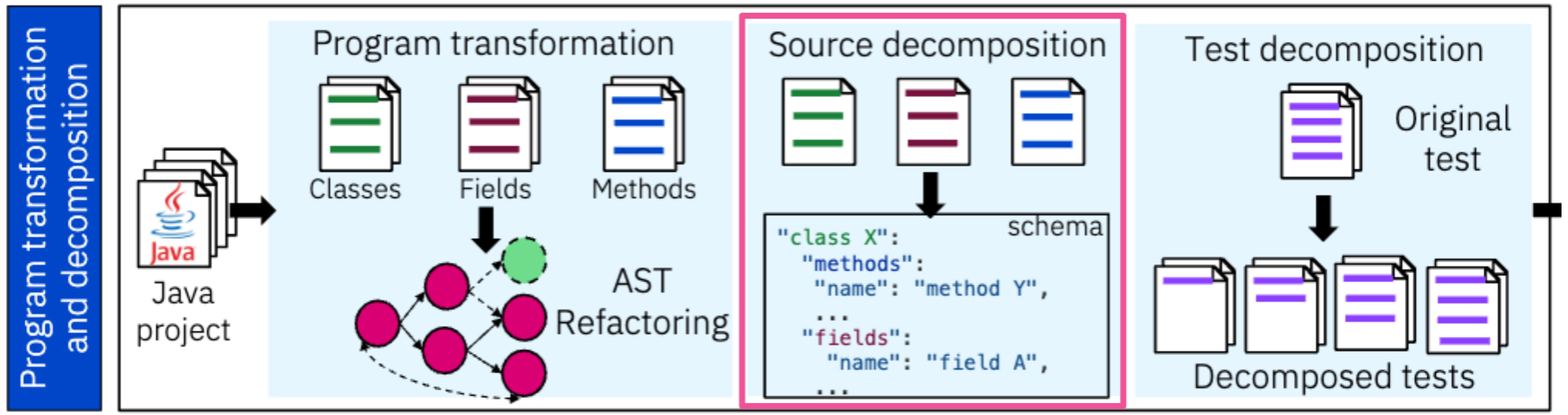

Program transformation and decomposition
Java project
Program transformation
Classes
Fields
Methods
AST Refactoring
Source decomposition
schema
"class X":
"methods":
"name": "method Y",
...
"fields":
"name": "field A",
...
Test decomposition
Original test
Decomposed tests
Java types
Python types
Universal Type Map
String int double
List Map short
LLM
str int float
List Dict int
Skeleton
class ArgExp(Exception):
__option: Option = None
@staticmethod
def ArgExp1(id: int, opt:
Option, msg: str)->ArgExp:
pass
def __init__(self, id:int,
opt:Option,msg:str):
pass
def getMsg(self) -> str:
pass
...
Skeleton validation
Type translation and Skeleton construction
Compositional translation and validation
Translated project
Execution report
Code translation and validation
GraalVM
Isolation validation
Test translation
t1.java
t2.java
t1.py
t2.py
pytest
Test selection
t1.java
t2.java
t3.java
Prompting
translate
from $SRC to $TRG
language.
$SRC_DEPENDENCIES
$SRC_SKELETON
$SRC_CODE
LLM
Syntax validation
Code traversal
Call graph
bottom-up
Reverse traversal

Program Transformation



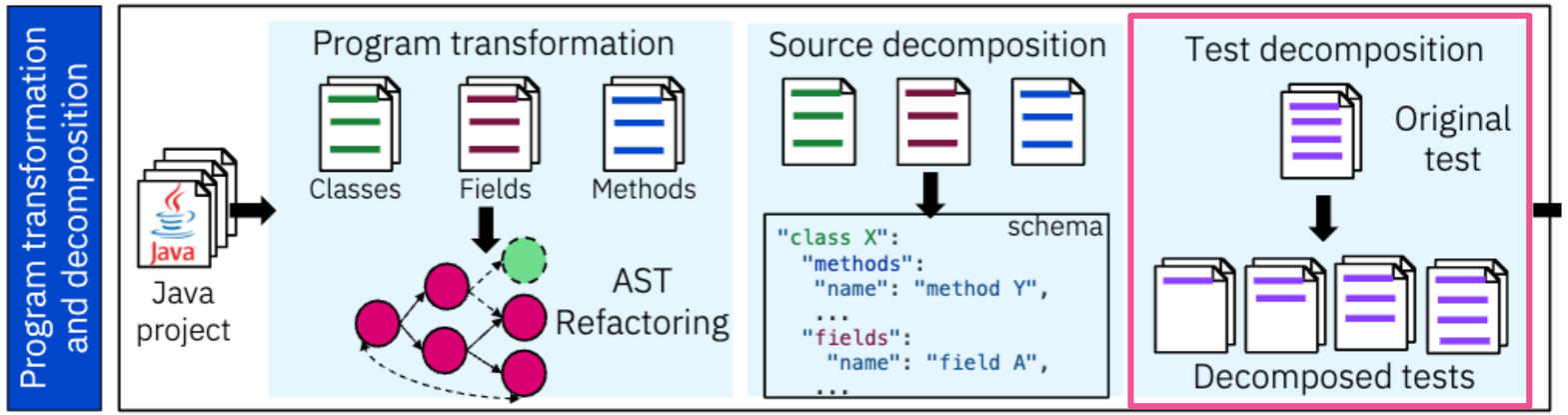
- Automatically refactors the code to resolve programming language-specific features that will be problematic in target language
  - Method/constructor overloading, circular dependencies, local or anonymous inner classes, pointers, pointer arithmetic, etc.

Source Program Decomposition



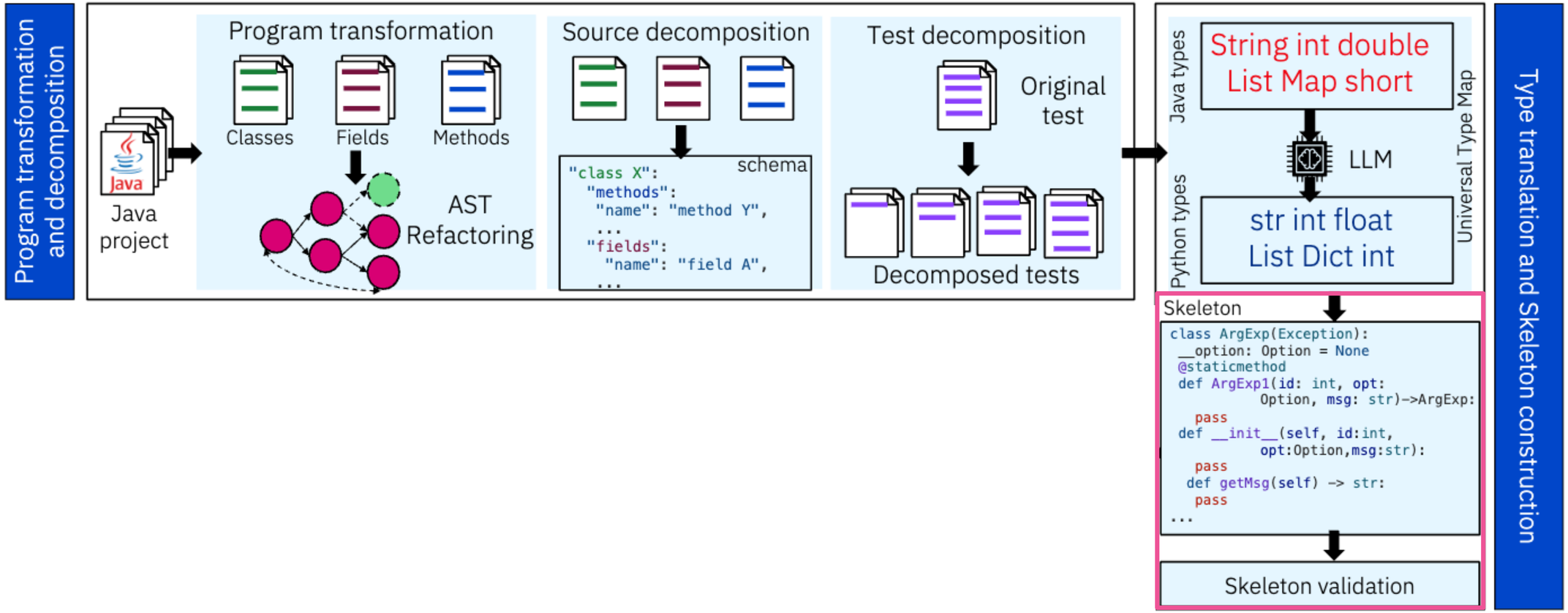
- Breaks the program into small fragments
  - Helps with limited context window and attention span of LLMs

Test Decomposition



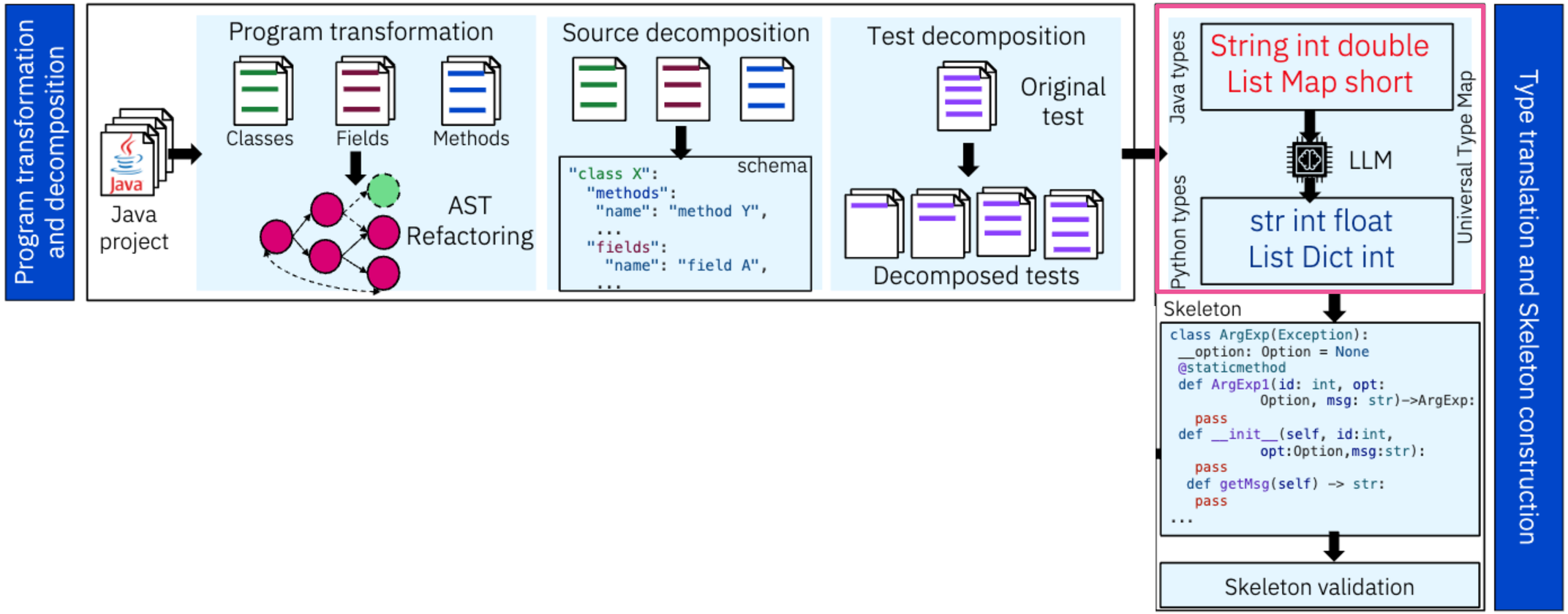
- ❖ Breaks the tests per method invocation
  - Helps with translation bug localization while tests are translated and executed on the translated code

Skeleton Construction and Validation



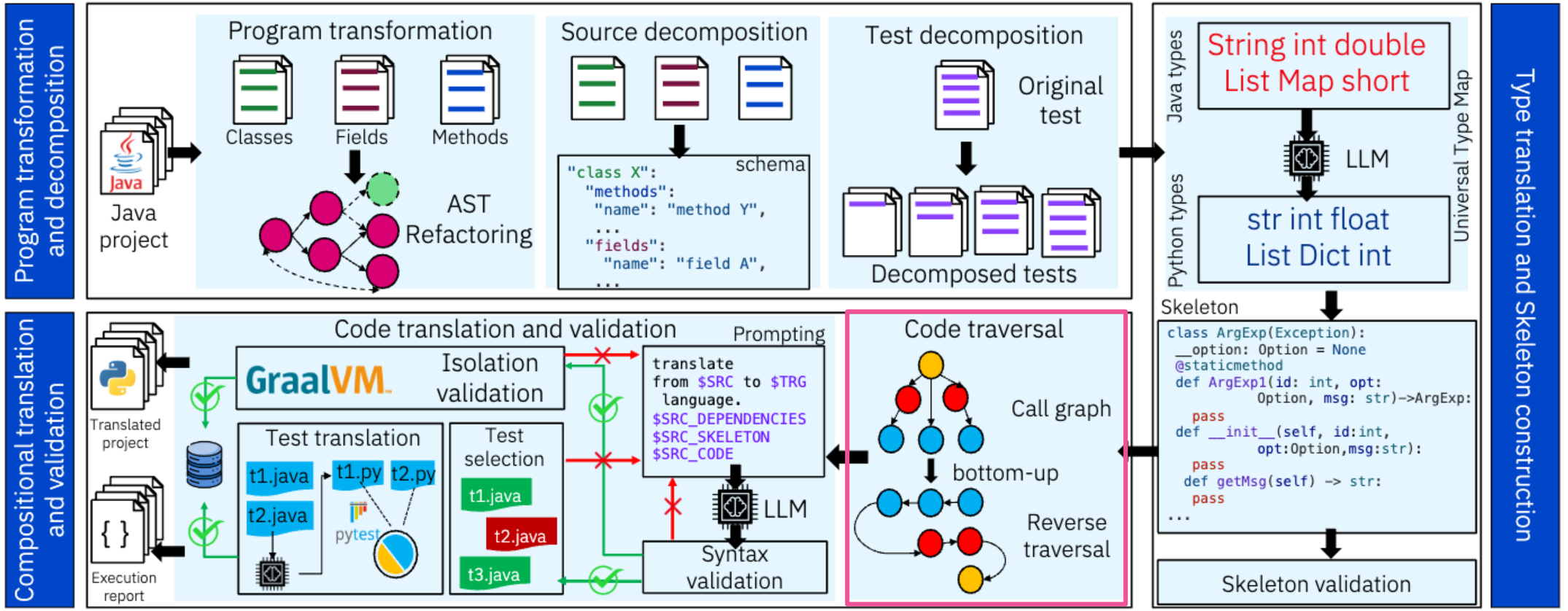
- Constructs the project skeleton in target language and uses it as a guide for compositional (fragment-by-fragment) translation

Skeletons as Translation Guideline



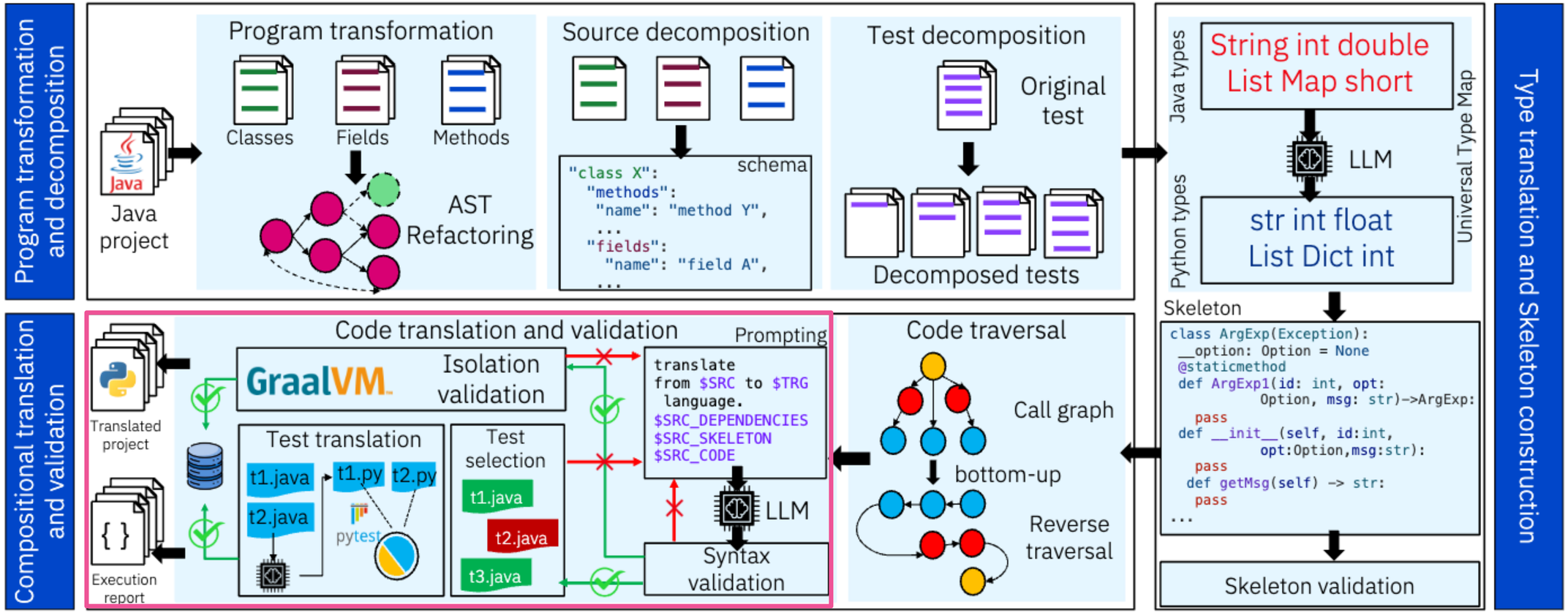
- Constructs the project skeleton in target language and uses it as a guide for compositional (fragment-by-fragment) translation
  - Requires initial type resolution to ensure compilability of the skeletons

Compositional Translation



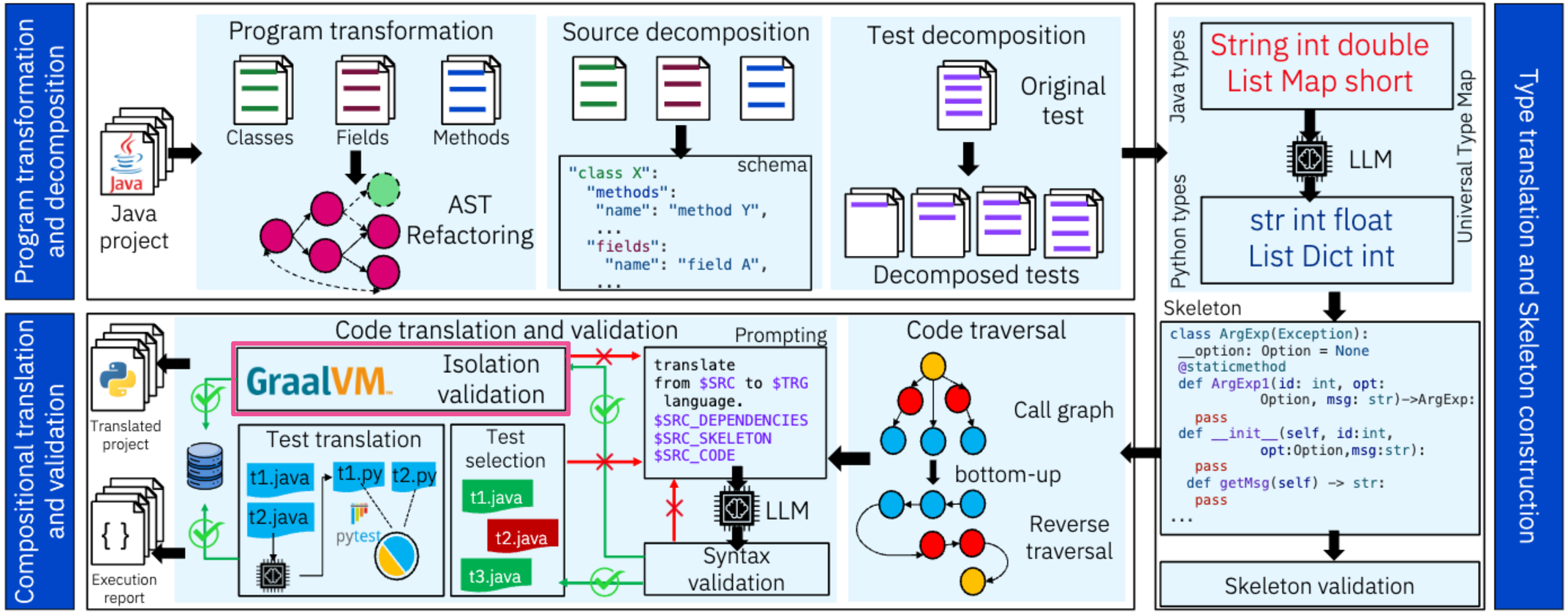
- Translates fragment-by-fragment in reverse topological call order
  - The order ensures all dependencies are previously translated

Compositional Translation and Validation



- Validates whether skeletons are compilable or pass applicable tests per each fragment translation

Language Interoperability



- Tests individual translated fragments in isolation using the tests in source language
  - Minimizes impact of test translation coupling effect (due to high degree of call chain dependency)

Evaluating AlphaTrans to Translate Java Repositories

- Ten large-scale Java projects to be translated to Python
- LLM: DeepSeekCoder-33b-Instruct

Subjects	# Classes	# Methods	# JUnit Tests	Method Coverage (%)	ATR (%)	SV (%)	# Fragments			
							Fields		Application Methods	Test Methods
							Application	Test		
cli [11]	58	664	437	94.14	96.60	100	104	57	273	2180
codec [12]	156	1780	992	91.03	96.01	100	425	140	680	2849
csv [13]	41	694	309	90.64	92.34	100	146	35	235	1272
exec [14]	56	407	70	54.84	78.90	100	104	27	248	327
fast-pfor [34]	82	971	82	54.55	87.40	100	127	14	748	302
fileupload [15]	49	381	39	13.02	98.31	100	121	39	192	194
graph [16]	118	879	146	58.78	97.13	100	216	29	541	975
jansi [49]	48	474	107	23.47	84.83	100	378	0	409	123
pool [17]	98	1097	73	22.29	91.60	100	203	91	682	649
validator [18]	130	1228	464	63.31	95.51	100	421	209	646	1463
Total	836	8575	2719	56.57	91.99	100	2245	641	4654	10334

Type Resolution and Skeleton Validation

- 91.99% automated type resolution (ATR values per project)
- 100% initial skeleton validation (SV values per project)

Subjects	# Classes	# Methods	# JUnit Tests	Method Coverage (%)	ATR (%)	SV (%)	# Fragments			
							Fields		Application Methods	Test Methods
							Application	Test		
cli [11]	58	664	437	94.14	96.60	100	104	57	273	2180
codec [12]	156	1780	992	91.03	96.01	100	425	140	680	2849
csv [13]	41	694	309	90.64	92.34	100	146	35	235	1272
exec [14]	56	407	70	54.84	78.90	100	104	27	248	327
fast-pfor [34]	82	971	82	54.55	87.40	100	127	14	748	302
fileupload [15]	49	381	39	13.02	98.31	100	121	39	192	194
graph [16]	118	879	146	58.78	97.13	100	216	29	541	975
jansi [49]	48	474	107	23.47	84.83	100	378	0	409	123
pool [17]	98	1097	73	22.29	91.60	100	203	91	682	649
validator [18]	130	1228	464	63.31	95.51	100	421	209	646	1463
Total	836	8575	2719	56.57	91.99	100	2245	641	4654	10334

Abbreviations: SV (Skeleton Validation) and ATR (Automated Type Resolution)

Effectiveness in Translating Real-World Java Projects

- 98.8% syntactical correctness
- 25.14% of the translated methods were functionally equivalent (GS+ M1 All)
  - More than half of methods that were covered by tests
- 27.03% runtime behavior validation (GS+M1 Some)

Subjects	AMF	Syntax Check (%)	SNEF (%)	GraalVM			Test Translation											M1	
				GS (%)	GF (%)	GE (%)	TNEF (%)	ATP (%)	OTF (%)			MTF (%)			ATF (%)			TPR (%)	All	Some
									O	RE	AF	O	RE	AF	O	RE	AF			
cli	273	100	5.86	70.70	11.72	11.72	35.90	8.42	10.62	51.72	48.28	32.23	91.07	8.93	6.96	100	0	10.08	0	16
codec	680	98.53	8.97	38.38	32.50	20.15	65.59	4.12	4.41	60.00	40.00	12.79	55.11	44.89	4.12	75.21	24.79	9.43	11	27
csv	235	98.72	9.36	38.72	26.81	25.11	74.47	0	13.62	96.88	3.13	0	0	0	2.55	100	0	0	0	3
exec	248	100	45.16	33.47	2.02	19.35	34.27	4.44	2.02	40.00	60.00	7.26	93.36	6.64	6.85	100	0	19.29	6	9
fast-pfor	748	95.32	45.45	12.03	24.20	18.32	41.71	4.28	1.74	84.62	15.38	3.74	79.23	20.77	3.07	85.88	14.12	20.08	6	25
fileupload	192	100	86.98	8.85	1.04	3.13	1.56	3.65	6.77	30.77	69.23	1.04	91.67	8.33	0	0	0	63.44	2	3
graph	541	99.63	41.22	24.77	22.92	11.09	57.12	0	0.92	100	0	0.18	100	0	0.55	100	0	11.04	0	1
jansi	409	99.76	76.53	8.07	11.49	3.91	22.25	0.24	0.98	100	0	0	0	0	0	0	0	1.07	0	1
pool	682	100	77.71	6.01	1.32	14.96	19.35	1.61	1.03	100	0	0.29	100	0	0	0	0	6.62	4	2
validator	646	99.23	36.69	30.50	11.15	21.67	46.44	3.25	5.42	71.43	28.57	4.18	84.50	15.50	4.02	99.12	0.88	11.70	1	31
Total	4654	98.80	43.43	24.50	16.23	15.84	41.92	2.88	3.72	70.52	29.48	5.44	82.77	17.23	2.62	92.86	7.14	9.76	30	118

GS shows percentage of Graal success. SNEF shows the percentage of fragments now covered by any tests in the source program. M1 shows number of fragments that were validated through test translation, while Graal could not validate them

Effectiveness in Translating Real-World Java Projects

- 98.8% syntactical correctness
- 25.
  - N
- 27.0

All the translated code are idiomatic Python, measured by PyLint (score 10 out of 10)

Subjects	AMF	Syntax Check (%)	SNEF (%)	GraalVM			Test Translation											M1	
				GS (%)	GF (%)	GE (%)	TNEF (%)	ATP (%)	OTF (%)			MTF (%)			ATF (%)			TPR (%)		
									O	RE	AF	O	RE	AF	O	RE	AF		All	Some
cli	273	100	5.86	70.70	11.72	11.72	35.90	8.42	10.62	51.72	48.28	32.23	91.07	8.93	6.96	100	0	10.08	0	16
codec	680	98.53	8.97	38.38	32.50	20.15	65.59	4.12	4.41	60.00	40.00	12.79	55.11	44.89	4.12	75.21	24.79	9.43	11	27
csv	235	98.72	9.36	38.72	26.81	25.11	74.47	0	13.62	96.88	3.13	0	0	0	2.55	100	0	0	0	3
exec	248	100	45.16	33.47	2.02	19.35	34.27	4.44	2.02	40.00	60.00	7.26	93.36	6.64	6.85	100	0	19.29	6	9
fast-pfor	748	95.32	45.45	12.03	24.20	18.32	41.71	4.28	1.74	84.62	15.38	3.74	79.23	20.77	3.07	85.88	14.12	20.08	6	25
fileupload	192	100	86.98	8.85	1.04	3.13	1.56	3.65	6.77	30.77	69.23	1.04	91.67	8.33	0	0	0	63.44	2	3
graph	541	99.63	41.22	24.77	22.92	11.09	57.12	0	0.92	100	0	0.18	100	0	0.55	100	0	11.04	0	1
jansi	409	99.76	76.53	8.07	11.49	3.91	22.25	0.24	0.98	100	0	0	0	0	0	0	0	1.07	0	1
pool	682	100	77.71	6.01	1.32	14.96	19.35	1.61	1.03	100	0	0.29	100	0	0	0	0	6.62	4	2
validator	646	99.23	36.69	30.50	11.15	21.67	46.44	3.25	5.42	71.43	28.57	4.18	84.50	15.50	4.02	99.12	0.88	11.70	1	31
Total	4654	98.80	43.43	24.50	16.23	15.84	41.92	2.88	3.72	70.52	29.48	5.44	82.77	17.23	2.62	92.86	7.14	9.76	30	118

GS shows percentage of Graal success. SNEF shows the percentage of fragments now covered by any tests in the source program. M1 shows number of fragments that were validated through test translation, while Graal could not validate them

Fixing Translation Bugs

How much does it take for human developers to fix translation bugs?

- Commons-FileUpload
  - 5.5 hours and required 120 line additions and 114 line deletions
- Commons-CLI
  - 11 hours to fully achieve all passing tests, making 614 and 1253 line additions and deletions
- Commons-CSV
  - 30 hours with 2676 and 999 line additions and deletions
- Commons-Validator
  - 34 hours to fix translation bugs, with 3585 and 2416 line additions and deletions

Fixing Translation Bugs

How much does it take for human developers to fix translation bugs?

Without AlphaTrans, it can take several days, if not weeks and months, to translate a project from scratch

❖ Com

-

❖ Com

- 11 hours to fully achieve all passing tests, making 614 and 1253 line additions and deletions

❖ Commons-CSV

- 30 hours with 2676 and 999 line additions and deletions

❖ Commons-Validator

- 34 hours to fix translation bugs, with 3585 and 2416 line additions and deletions

Impact of Test Suite Quality

- Augmented original test suites with EvoSuite unit tests
  - Translated unit tests individually and executed on the translated code
  - Decrease in call chain length due to more isolated unit tests: 21.84 to 11.41
  - Increase in TPR (+5.85%) and ATP (+2.11%)

Subjects	Developer-Written Test					EvoSuite Test				
	Method Coverage (%)	# Decomposed Tests	Avg. Methods Executed / Test	TPR (%)	ATP (%)	Method Coverage (%)	# Tests	Avg. Methods Executed / Test	TPR+ (%)	ATP+ (%)
cli	94.14	3036	34.25	10.08	8.42	95.97	569	12.15	2.99	1.47
codec	91.03	3522	10.56	9.43	4.12	80.74	1141	8.02	3.51	0.88
csv	90.64	1219	52.62	0	0	74.04	220	39.16	0	0.00
exec	54.84	311	18.99	19.29	4.44	61.29	245	6.32	3.27	1.21
fast-pfor	54.55	249	41.62	20.08	4.28	39.17	1843	4.31	5.59	1.07
fileupload	13.02	93	3.54	63.44	3.65	70.31	231	5.29	11.26	2.60
graph	58.78	933	25.02	11.04	0.00	76.71	800	9.00	5.13	0.92
jansi	23.47	187	13.57	1.07	0.24	51.83	332	9.08	3.31	0.73
pool	22.29	287	6.52	6.62	1.61	37.24	394	7.36	10.41	2.20
validator	63.31	1479	11.68	11.70	3.25	81.42	1305	13.43	9.73	7.59
Total	56.57	11316	21.84	9.76	2.88	66.87	7080	11.41	5.85	2.11

TPR shows the percentage of test passes and ATP shows the percentage of all decomposed tests pass for a fragment

Impact of Test Suite Quality

- Augmented original test suites with EvoSuite unit tests
  - Translated unit tests individually and executed on the translated code
  -
  -

High coverage test suites with focused unit tests are the key for test-based translation validation

Subjects	Developer-Written Test					EvoSuite Test				
	Method Coverage (%)	# Decomposed Tests	Avg. Methods Executed / Test	TPR (%)	ATP (%)	Method Coverage (%)	# Tests	Avg. Methods Executed / Test	TPR+ (%)	ATP+ (%)
cli	94.14	3036	34.25	10.08	8.42	95.97	569	12.15	2.99	1.47
codec	91.03	3522	10.56	9.43	4.12	80.74	1141	8.02	3.51	0.88
csv	90.64	1219	52.62	0	0	74.04	220	39.16	0	0.00
exec	54.84	311	18.99	19.29	4.44	61.29	245	6.32	3.27	1.21
fast-pfor	54.55	249	41.62	20.08	4.28	39.17	1843	4.31	5.59	1.07
fileupload	13.02	93	3.54	63.44	3.65	70.31	231	5.29	11.26	2.60
graph	58.78	933	25.02	11.04	0.00	76.71	800	9.00	5.13	0.92
jansi	23.47	187	13.57	1.07	0.24	51.83	332	9.08	3.31	0.73
pool	22.29	287	6.52	6.62	1.61	37.24	394	7.36	10.41	2.20
validator	63.31	1479	11.68	11.70	3.25	81.42	1305	13.43	9.73	7.59
Total	56.57	11316	21.84	9.76	2.88	66.87	7080	11.41	5.85	2.11

TPR shows the percentage of test passes and ATP shows the percentage of all decomposed tests pass for a fragment

AlphaTrans Is Just The Beginning

1. Translating library APIs

- Major source of LLM hallucination and translation failure

```
1          ----------- JAVA SOURCE CODE -----------
2 Calendar calendar = Calendar.getInstance()
3 calendar.set(Calendar.MONTH, 0);
```

```
1          ---------- PYTHON TRANSLATION ----------
2   calendar = datetime.datetime.now()
3 - calendar = calendar.replace(month=0)
4 + calendar = calendar.replace(month=1)
```

2. Implementing non-existent logic

```
1          ----------- JAVA SOURCE CODE -----------
2 Iterator<String> headers = ls.keySet().iterator();
3
4 assertEquals("content", headers.next());
5
6 assertFalse(headers.hasNext());
```

```
1          ---------- PYTHON TRANSLATION ----------
2 - headers = iter(ls.keys())
3 + headers = PeekableIterator(ls.keys())
4   self.assertEqual("content", next(headers))
5 - self.assertFalse(next(headers, None) is not None)
6 + self.assertFalse(headers.hasNext())
```

AlphaTrans Is Just The Beginning

3. Teaching LLMs to deal with corner cases (e.g., type casting)

```
1          ----------- JAVA SOURCE CODE -----------
2 qChar = "'";
3 nullStr = null;
4
5 this.qNullStr = qChar + nullStr + qChar;
```

```
1          ---------- PYTHON TRANSLATION ----------
2   qChar = "'"
3   nullStr = None
4 - self.qNullStr = qChar + nullStr + qChar
5 + self.qNullStr = qChar + str(nullStr) + qChar
```

4. Automated type resolution

- Removing the need for manual validation and contextual type resolution

5. A better in-isolation testing

- GraalVM has its limitation, i.e., state sharing
- GraalVM is limited to handful of PLs

AlphaTrans Is Just The Beginning

6. Other forms of translation validation
    - Runtime verification
    - Formal verification of specific properties, e.g., concurrency

7. Translating arbitrary pairs of PLs
    - Far from being a PL-agnostic pipeline
    - PL-specific properties are unique to programing languages

AlphaTrans: A Neuro-Symbolic Compositional Approach for Repository-Level Code Translation and Validation

Ali Reza Ibrahimzada[1], Kaiyao Ke[1], Mrigank Pawagi[3], Muhammad Salman Abid[4], Rangeet Pan[2], Saurabh Sinha[2], Reyhaneh Jabbarvand[1]

[1]University of Illinois Urbana-Champaign [2]IBM Research [3]Indian Institute of Science [4]Cornell University

- Try it on new projects!
- Come talk to me if interested in repository-level code translation

Tool: https://github.com/Intelligent-CAT-Lab/AlphaTrans