An Empirical Study of C to Rust Translation using Local Large-Language Models

Nathan Rutherford & Dan O'Keeffe

Email: nathan.rutherford.2019@live.rhul.ac.uk
Web: nrutherford.com

The Third International Workshop on Large Language Models for Code
April 14, 2026

Royal Holloway, University of London

Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Manual Translated C Code (💰)

Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Manual Translated C Code (💰)

Rule-based Translation [5, 4] (☣)

Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Manual Translated C Code (💰)

Rule-based Translation [5, 4] (☣)

LLM Translation [9, 2, 1, 10, 7, 8] (🌀)

Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Manual Translated C Code (💰)

Rule-based Translation [5, 4] (☣)

LLM Translation [9, 2, 1, 10, 7, 8] (🌀)

Hybrid Translation [6, 11, 3] (文A)

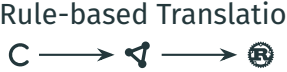
Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Manual Translated C Code (💰)

Rule-based Translation [5, 4] (☣)



LLM Translation [9, 2, 1, 10, 7, 8] (🌀)

Hybrid Translation [6, 11, 3] (文A)

Existing approaches leverage proprietary LLMs (ChatGPT, Claude)

- Larger context widows
- Cloud based (non-private)

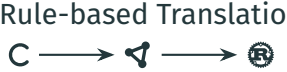
Motivation

Desire to rewrite (legacy) code in Rust for improved security guarantees.

- Rust has two variants: *Safe* and *Unsafe* Rust

Manual Translated C Code (💰)

Rule-based Translation [5, 4] (☣)



LLM Translation [9, 2, 1, 10, 7, 8] (🌀)

Hybrid Translation [6, 11, 3] (文A)

Existing approaches leverage proprietary LLMs (ChatGPT, Claude)

- Larger context widows
- Cloud based (non-private)

What happens in a **resource constrained** setting?

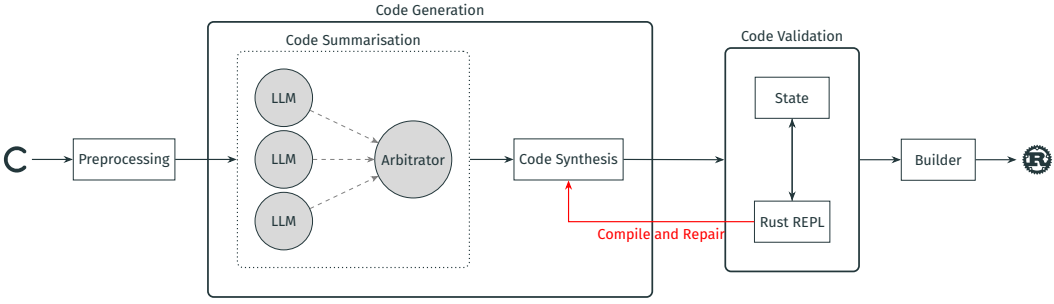
Contributions

OXIDATION - a C2Rust translation framework for resource constrained settings

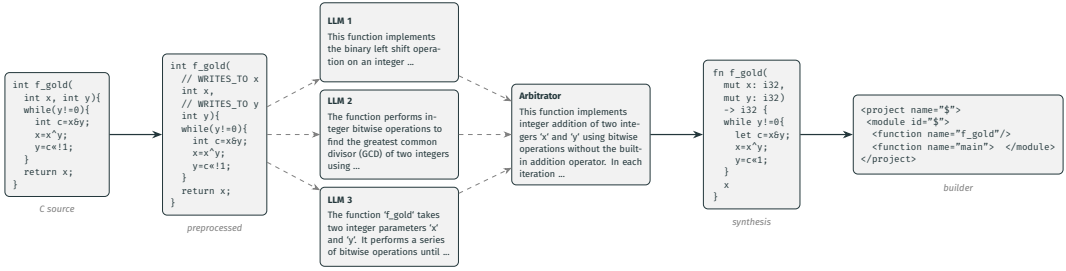
Two new techniques for context management:

1. Arbitration for structuring code summarisation and generation
2. Incremental composition and validation of generated code fragments

Oxidation Overview



Code Arbitration



Goal: Allow the Arbitrator to compensate for what other models miss

- Specialise summarisations on different properties

Reduced context size in comparison to thinking / planning modes.

Evaluation

Evaluation Goals:

1. Evaluate the runtime and accuracy of the OXIDATION pipeline for different local LLM configurations.
2. Analyse translation failures to better understand problematic code patterns for local LLMs.

Evaluation

Evaluation Goals:

1. Evaluate the runtime and accuracy of the OXIDATION pipeline for different local LLM configurations.
2. Analyse translation failures to better understand problematic code patterns for local LLMs.

Evaluation Dataset: Transcoder-IR

- Samples contain a fgold() baseline, LLM generates ffilled()
- Preprocessed to remove invalid C and any CPP leaving 425/698 samples
- Models Used: Llama3.2:3b, Mistral:7b, Gemma3:4b, Qwen3:4b

Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66	280	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389	4439	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66 ≈ -3×	280	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389	4439	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66	280 -29%	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389	4439	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66	280	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389	4439	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66	280	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389 +31%	4439	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

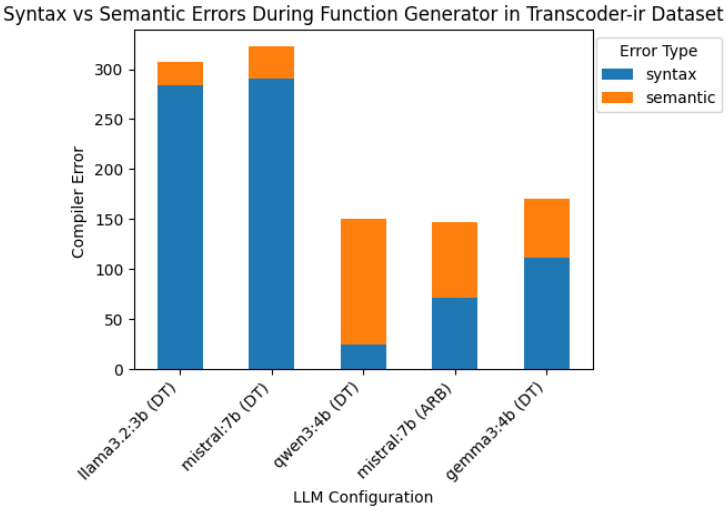
Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66	280	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389	4439 +1021%	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

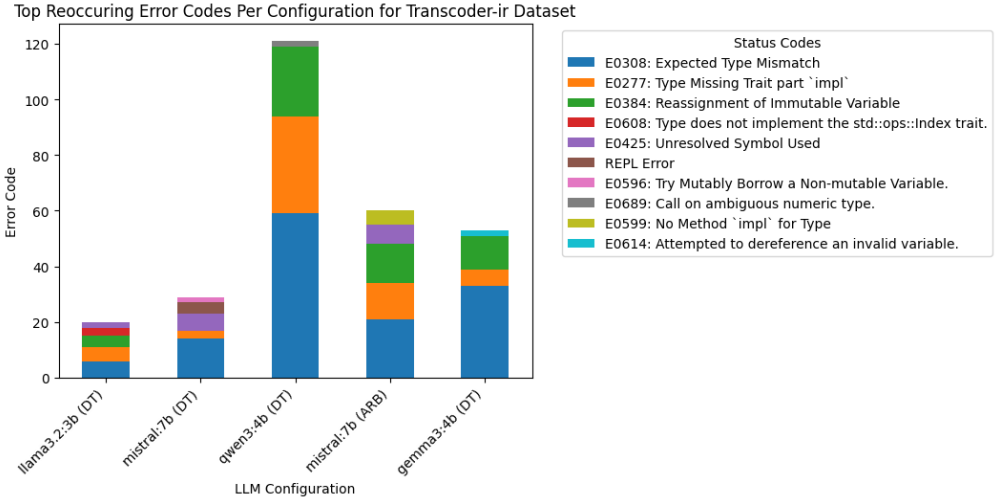
Results: Average Pipeline Runtime

Condition	Correct Translations	Tokens In	Tokens Out	Time Per Prompt
llama3.2:3b (DT)	92	261	63	2.00s
mistral:7b (DT)	66	280	157	5.00s
mistral:7b (ARB)	267	396	222	8.00s
qwen3:4b (DT)	389	4439	128	1m53.00s
gemma3:4b (DT)	250	274	145	3.00s

Results: Syntax vs Semantic Errors



Results: Reoccuring compilation errors



Conclusions

We present OXIDATION, a translation pipeline for resource constrained LLMs.

Paper & Code

Conclusions

We present OXIDATION, a translation pipeline for resource constrained LLMs.

Function level results over the Transcoder-IR dataset suggest:

- 3× increase in translation success for non-thinking models using arbitration
- 31% drop in translation success, but a 5× perf gain for thinking models.

Paper & Code

Conclusions

We present OXIDATION, a translation pipeline for resource constrained LLMs.

Function level results over the Transcoder-IR dataset suggest:

- 3× increase in translation success for non-thinking models using arbitration
- 31% drop in translation success, but a 5× perf gain for thinking models.

Future Work:

- Designing code translation for agentic software patterns
- Applying these techniques to different categories of software

Paper & Code

References i

[1] Xuemeng Cai et al. *RustMap: Towards Project-Scale C-to-Rust Migration via Program Analysis and LLM*. Mar. 22, 2025. DOI: 10.48550/arXiv.2503.17741. arXiv: 2503.17741[cs]. URL: http://arxiv.org/abs/2503.17741 (visited on 06/02/2025).

[2] Yufeng Du et al. *Context Length Alone Hurts LLM Performance Despite Perfect Retrieval*. Oct. 6, 2025. DOI: 10.48550/arXiv.2510.05381. arXiv: 2510.05381[cs]. URL: http://arxiv.org/abs/2510.05381 (visited on 10/31/2025).

References ii

[3] Yifei Gao et al. *PR2: Peephole Raw Pointer Rewriting with LLMs for Translating C to Safer Rust*. May 9, 2025. DOI: 10.48550/arXiv.2505.04852. arXiv: 2505.04852[cs]. URL: http://arxiv.org/abs/2505.04852 (visited on 05/12/2025).

References iii

[4] Jaemin Hong and Sukyoung Ryu. **"To Tag, or Not to Tag: Translating C's Unions to Rust's Tagged Unions".** In: *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. ASE '24: 39th IEEE/ACM International Conference on Automated Software Engineering. Sacramento CA USA: ACM, Oct. 27, 2024, pp. 40–52. ISBN: 979-8-4007-1248-7. DOI: 10.1145/3691620.3694985. URL: https://dl.acm.org/doi/10.1145/3691620.3694985 (visited on 05/12/2025).

[5] Immunant and Galois. *C2Rust: Migrate C code to Rust*. 2022. URL: https://github.com/immunant/c2rust.

References iv

[6] Vikram Nitin et al. *C2SaferRust: Transforming C Projects into Safer Rust with NeuroSymbolic Techniques.* Jan. 24, 2025. DOI: 10.48550/arXiv.2501.14257. arXiv: 2501.14257[cs]. URL: http://arxiv.org/abs/2501.14257 (visited on 02/10/2025).

[7] Manish Shetty et al. *Syzygy: Dual Code-Test C to (safe) Rust Translation using LLMs and Dynamic Analysis.* Dec. 21, 2024. DOI: 10.48550/arXiv.2412.14234. arXiv: 2412.14234[cs]. URL: http://arxiv.org/abs/2412.14234 (visited on 01/02/2025).

References v

[8] Chaofan Wang et al. *EvoC2Rust: A Skeleton-guided Framework for Project-Level C-to-Rust Translation*. Oct. 10, 2025. DOI: 10.48550/arXiv.2508.04295. arXiv: 2508.04295[cs]. URL: http://arxiv.org/abs/2508.04295 (visited on 10/31/2025).

[9] Aidan Z. H. Yang et al. *VERT: Verified Equivalent Rust Transpilation with Large Language Models as Few-Shot Learners*. May 25, 2024. DOI: 10.48550/arXiv.2404.18852. arXiv: 2404.18852[cs]. URL: http://arxiv.org/abs/2404.18852 (visited on 11/25/2024).

References vi

[10] Hanliang Zhang et al. **“Scalable, Validated Code Translation of Entire Projects using Large Language Models”.** In: *Proceedings of the ACM on Programming Languages* 9 (PLDI June 10, 2025), pp. 1616–1641. ISSN: 2475-1421. DOI: 10.1145/3729315. URL: https://dl.acm.org/doi/10.1145/3729315 (visited on 10/31/2025).

[11] Tianyang Zhou et al. *LLM-Driven Multi-step Translation from C to Rust using Static Analysis.* Mar. 18, 2025. DOI: 10.48550/arXiv.2503.12511. arXiv: 2503.12511[cs]. URL: http://arxiv.org/abs/2503.12511 (visited on 05/23/2025).

Verification: I/O Equivilence

Category	Llama3	Mistral 7B	Qwen3 4B	Arb Mistral 7B	Gemma3 4B
No Translation	275	359	30	158	146
ABI Incompatible	7	39	179	32	131
Runtime Timeout	0	1	0	0	0
Missing Rust Import	3	0	1	0	1
C Compilation Error (Clang)	2	3	4	7	7
Function Signature Mismatch	90	0	6	139	29
Test Fail: Pointer Related	14	8	50	18	26
Test Fail: Runtime Exception	1	0	0	1	0
Test Fail: IO Mismatch	12	8	35	33	22
Test Pass	17	4	120	35	63

Results: Impact of Arbitration Summaries

Original C

```c
int f_gold ( int x ) {
  int m = 1;
  while ( x & m ) {
    x = x ^ m;
    m <<= 1;
  }
  x = x ^ m;
  return x;
}
```

3b LLM

```rust
let mut x = 10u32; let
    mut y = 2u32;
while y != 0 {
int carry = x & y;
x = x ^ y;
y = (y - carry) / 2;
}
x;
```

4b & Thinking LLM

```rust
fn f_gold(x: i32, y: i32)
    -> i32 {
    let mut x = x;
    let mut y = y;
    while y != 0 {
        let carry = x & y
            ;
        x = x ^ y;
        y = carry << 1;
    }
    x
}
```

Arb w/out Thinking

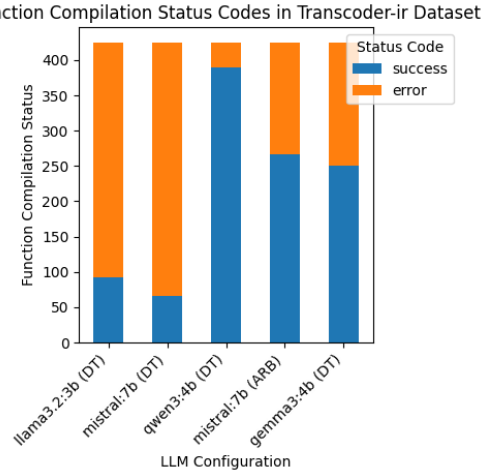
```rust
fn f_gold(mut x: u64, mut
    y: u64) -> u64 {
    loop {
        let carry = x & y
            ;
        x ^= y;
        y <<= 1;
        if y == 0 {
            break;
        }
        y |= carry;
    }
    x
}
```

Arb w/ Thinking

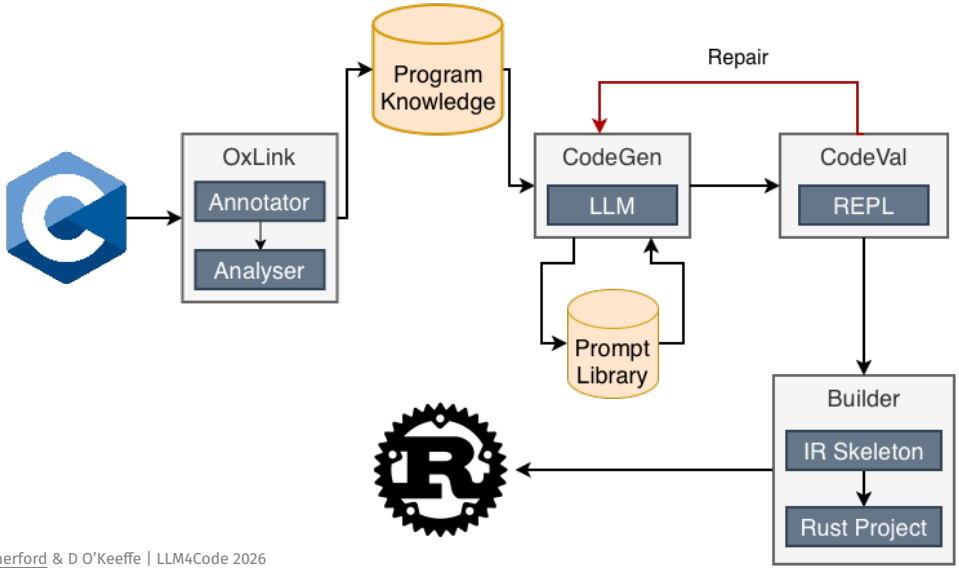
```rust
fn f_gold(mut x: i32, mut
    y: i32) -> i32 {
    while y != 0 {
        let carry = x & y
            ;
        x = x ^ y;
        y = carry << 1;
    }
    x
}
```

Figure 1: Translation for ADD_TWO_NUMBERS_WITHOUT_USING_ARITHMETIC_OPERATORS

Results: Function Compilation Status



System Overview



Arbitration