Information Parity for Code: The Scope of Transfer in Multilingual Code Models

Alexander Tsvetkov Alon Kipnis

School of Computer Science, Reichman University

Paper, code & data
github.com/tsalex1992/information-parity-for-code

Reichman University

Problem

Status quo: Multilingual code benchmarks show uneven model performance across programming languages.

Question: Does a model's lower score in a language reflect broader difficulty with that language, or difficulty specific to the task?

Goal: Use Code IP, a benchmark-free, language-level metric, to investigate when score gaps reflect a model's broader difficulty with a language rather than the task.

Key Contributions

- We define **Code Information Parity (Code IP)**, a metric of how efficiently a model encodes code in each language relative to a reference language (Python), and evaluate it on an audited parallel corpus of 22 algorithms in 20 languages.
- We show that Code IP is **regime-dependent**: it tracks short-horizon generation, weakens on execution prediction, and does not transfer to reverse reasoning, constrained generation, or whole-program synthesis.
- We explain what Code IP captures by separating per-token cost, token count, and public code-volume effects.

Method

Audited parallel code → Negative log-likelihood → Code Information Parity → Task-regime comparison

$$\text{Code IP}(\ell) = \frac{\text{NLL}(\ell)}{\text{NLL}(\text{Python})}$$

Code IP measures how many bits a model needs to encode a program in programming language ℓ, relative to the same program in a reference language (Python). Lower Code IP means greater model-relative efficiency. Code IP extends Information Parity (Tsvetkov & Kipnis, EMNLP Findings 2024) from natural language to code.

Controlled Corpus

22 classic algorithms × 20 languages = 440 implementations ✓ Passed execution checks

Each implementation preserves the same task, algorithmic family, intermediate work, and output semantics as the Python reference. We remove benchmark harnesses, examples, unnecessary imports, boilerplate, and standard-library shortcuts.

Evaluation Setup

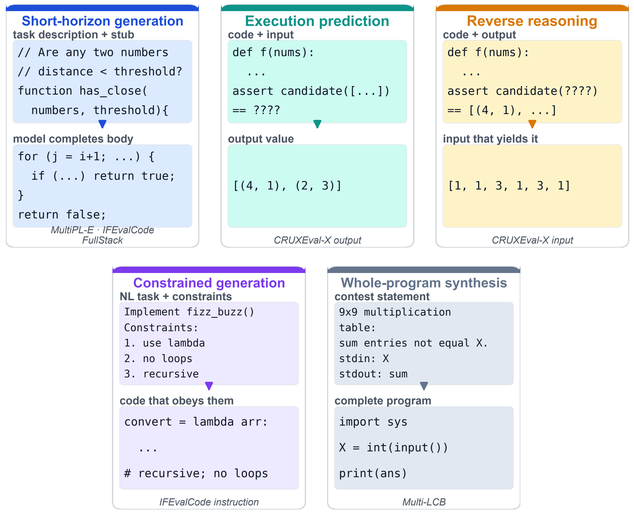
- **Scoring:** raw code only, native tokenizer, total next-token negative log-likelihood (NLL) in bits.
- **Benchmarks:** MultiPL-E, CRUXEval-X, IFEvalCode, FullStack Bench, and Multi-LCB, split into 11 slices (e.g., CRUXEval-X input vs. output) grouped into 5 task regimes.
- **Statistic:** within-model Spearman correlation across non-Python languages, pooled with Fisher-z intervals.
- **Coverage:** Code IP is computed for 80 models; each slice uses the models and languages it shares with Code IP (15–28 models, 5–18 non-Python languages).

What Code IP Captures

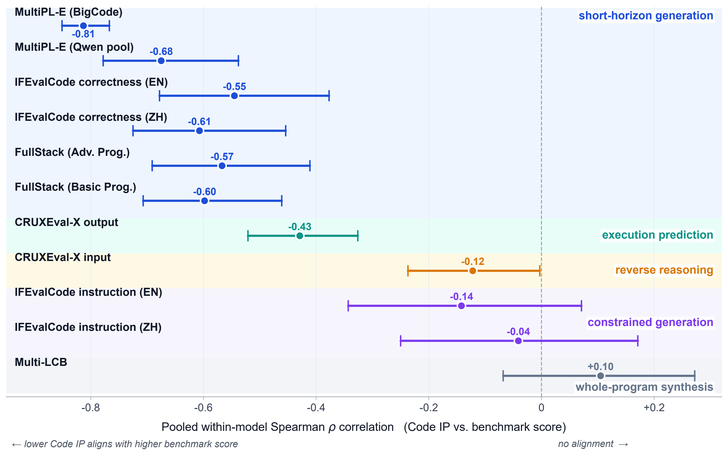
$$\text{Code IP} = \underbrace{\frac{\text{bits/token}(\ell)}{\text{bits/token}(\text{Python})}}_{\text{relative per-token cost}} \times \underbrace{\frac{\#\text{tokens}(\ell)}{\#\text{tokens}(\text{Python})}}_{\text{relative token count}}$$

- **Information-theoretic view:** the number of bits needed to losslessly compress a program with the model, relative to the same program in Python.
- **Combines:** tokenization and verbosity with model-dependent representation cost.
- **Aligns with:** tasks where success depends on continuing or completing code-like context.
- **Weakens when:** tasks move beyond bounded forward completion into reasoning, planning, or synthesis.

5 Task Regimes



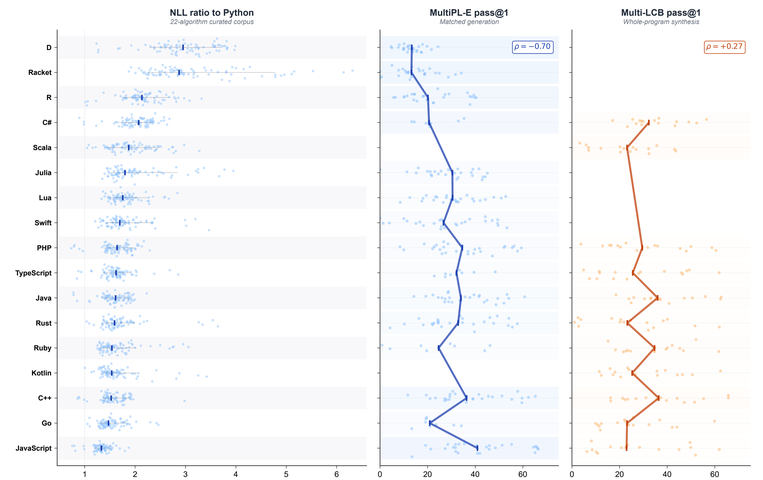
Main Result



Bars show descriptive 95% Fisher-z intervals.

- **Strong** on short-horizon generation (ρ from -0.55 to -0.81).
- **Moderate** on execution prediction ($\rho = -0.43$).
- **Negligible** for reverse reasoning, constrained generation, and whole-program synthesis (ρ from -0.14 to $+0.10$).

Language Ordering Evidence



Languages are ordered by median Code IP. MultiPL-E roughly follows that ordering; Multi-LCB does not.

- The same language ordering is informative for `MultiPL-E` but not `Multi-LCB`.

Robustness: Is It Just Training-Data Volume?

Code IP still predicts short-horizon benchmark performance after controlling for public code volume.

Slice	Code IP ρ after volume control
MultiPL-E BigCode	-0.550
MultiPL-E Qwen	-0.550
CRUXEval-X output	-0.262
Multi-LCB	$+0.214$

Partial Spearman correlations after controlling for $\log_{10}$ language volume in The Stack v2 SWH-full; Stack v2 covers 16/19 non-Python corpus languages.

Takeaway

Code IP measures a model's relative competence across languages without running any benchmark. On short-horizon generation, it identifies which languages the model is weak in. On tasks that require reasoning and planning, this competence alone no longer explains the gaps. Thus, Code IP helps distinguish language weakness from task weakness.