Inspectable Control for Structure-Preserving Software Regeneration

Control should happen where structure lives: above tokens.

Alexey Gavrilov, Alan-Barsag Gazzaev, Mikhail Mozikov, Ilya Makarov, Sergey Muravyov

ITMO University · AXXX · FSE Companion '26 · Montréal, QC, Canada

FSE COMPANION '26
Montréal, QC, Canada

doi 10.1145/3803437.3807386

Telegram discussion
@aogavrilov

Problem

Keep structure stable.

Implementation details should change; API and skeleton should not.

Idea

Lock latent positions.

Developers choose positions to lock, not embedding vectors.

Above tokens, structure becomes a control surface.

Result

Parse validity **+13.8 pp**

45.3% → **59.1%**

unconditional → prefix lock k=4

variation stays high

93.6% edit freedom

99.8% diversity

Lock structure. Regenerate the rest.

Encode into top-level structure handles and lower-level detail handles; freeze selected positions, regenerate the rest.

CODE STRUCTURE

API / signature

control flow

return shape

surface tokens vary

LATENT CONTROL SURFACE

Hierarchical VQ-VAE codes

top level = structure handles

lower level = detail handles

SELECT LOCKS

developer selects structural positions to keep fixed

REGENERATE UNLOCKED

locked regenerated

system changes only unlocked latent positions

CONTROLLED VARIANT

preserved structure region

editable detail region

conceptual scope

DIAGNOSTICS

- Parse
- Skeleton
- Signature
- Edit freedom
- Diversity

Pipeline checks

Exact lock preservation

Locked positions remain byte-exact in latent space.

Full lock sanity check

Locking all positions reduces to codec reconstruction.

Hierarchy reading

Top-level codes are the stronger structural control point.

Evidence: coarse locks raise validity

Main comparison: lock the first four top-level latent positions, then regenerate the rest.

main validity test

Baseline 45.3% parse validity → **MAIN RESULT +13.8 pp** Prefix lock k=4 **59.1%** parse validity

stress-test check

Signature span **60.0%** still not API-exact

Interpretation

k=4 already captures coarse program structure; locking those positions improves validity.

Boundary

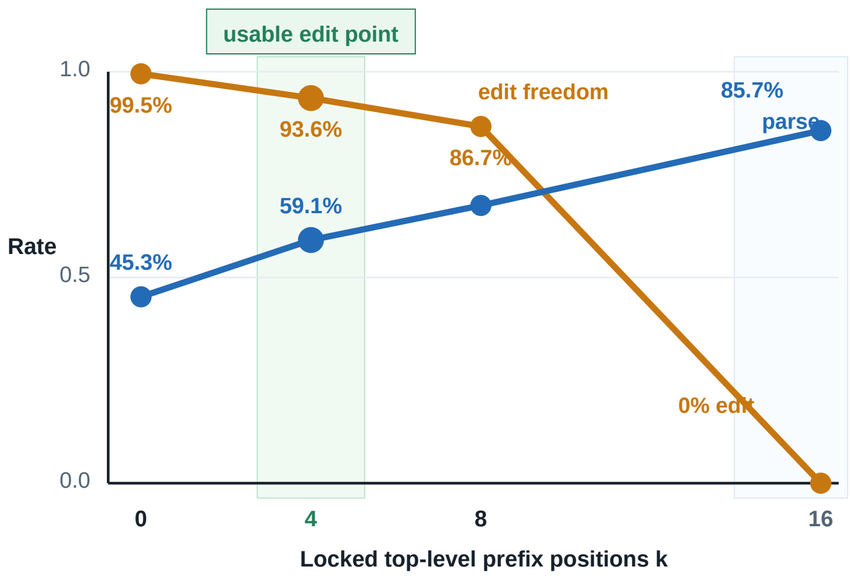
Exact API/interface preservation remains open.

Ceiling check: codec is not the bottleneck

Full lock reconstruction reaches 85.7% parse, 84.8% skeleton, 49.3% signature.

Controllability-freedom trade-off

More locking improves validity, but k=16 removes edit freedom; k=4 is the usable setting.



Recommended **k=4** 59.1% parse 93.6% freedom

+13.8 pp parse over baseline

Boundary k=16 is reconstruction only, not editing.

Use k=4 for editable regeneration; k=16 only tests the codec ceiling.

How it works in code

Why change it? Prevent divide-by-zero while keeping the contract fixed.

Green = same API/return shape. Orange = denominator logic selected for regeneration. Blue = diagnostics.

1 BEFORE: SOURCE

```
def normalize(xs):
    total = sum(xs)
    return [x / total for x in xs]
```

2 CONTROL STEP

top-level latent handles

keep 0..3 | resample denominator logic

3 AFTER: CONTROLLED VARIANT

```
def normalize(xs):
    total = sum(xs) or 1.0
    return [x / total for x in xs]
```

Preserved: function name, args, return shape

Changed: denominator gains zero-sum guard

Verified: parse, skeleton, signature, freedom

Why not tokens?

Token masks freeze surface text. Structure control needs handles above raw tokens.

Contribution

Developers lock positions, not embedding vectors: freeze structural handles, regenerate the rest, and measure validity/freedom.

Current boundary

Validity and skeleton improve; exact API preservation, semantic tests, and downstream repair remain open.

Metric key

Parse Python accepts output

Skeleton coarse program form

Signature interface span

Freedom useful variation

Study: 2,000 CodeParrot Clean functions; 64-token inputs; argmax decoding.

References

[1] Razavi et al. VQ-VAE-2. NeurIPS 2019.

[2] Austin et al. Structured D3PM. NeurIPS 2021.

[3] Sahoo et al. Masked Diffusion LMs. NeurIPS 2024.

[4] Barke et al. Grounded Copilot. PACM HCI CSCW 2023.

[5] Zhang et al. RepoCoder. EMNLP 2023.

Takeaway: above tokens, structure becomes controllable.

Lock latent positions, regenerate implementation details, then verify validity and freedom.

Acknowledgment: Foundation for National Technology Initiative's Projects Support

Contact: alexgavrilov28@gmail.com