Verifiable Structured Generation in LLMs via Tokenizer-Aware Constraints, Proper Masked Training, and Bounded Repair

David Xu[1]

[1]China Mobile Research Institute

Abstract

We present OIP-CAD, a verification-first framework for structured generation in large language models (LLMs). OIP-CAD integrates: (i) tokenizer-aware safety using a formal boundary model, deterministic BPE/WordPiece/Unigram contracts with proofs and detectors that include Unicode/normalization edge cases and a versioned self-test manifest with conservative fallbacks; (ii) masked-logit training with a complete proof of strict properness as $B \to \infty$, explicit finite-B leakage bounds and sensitivity, and an adaptive masking schedule that preserves calibration under mixture training; (iii) deterministic, bounded-latency minimal-edit repair via anytime A* (ARA*) with a max-combination of consistent admissible heuristics and an ILP fallback with stated total unimodularity (TU) conditions and diagnostics; and (iv) ontology-initialized prefixes (OIP) trained with InfoNCE that are provably safe under mask dominance and constrained by a KL trust region, with ablations and a process reward model (PRM) integration.

We provide formal results and a fully reproducible, controlled micro-benchmark that validates our finite-B leakage bounds and demonstrates invariant-preserving decoding under a simple grammar. We release a tokenizer self-test manifest and minimal scripts to reproduce figures and tables in this paper. Large-scale results on task suites are an important direction, but are out of scope for this artifact-light submission; our focus is on proofs, safety contracts, and correctness-first micro-benchmarks.

Symbols and Notation

- $\mathcal{U}$: byte alphabet; Σ: tokenizer tokens; Γ: grammar/output symbols. - T: temperature; η: label smoothing; α, B: mask parameters. - $K = |\Sigma|$, $k = |\mathcal{S}_t|$: vocabulary and legal-set sizes. - Δ: max logit margin between best legal vs best illegal. - L: locality horizon; M: number of BPE merges; G_τ: Merge-DAG. - Types: a finite set T of semantic classes (e.g., number, string, identifier). We write $P : \Sigma \to 2^{\mathsf{T}}$ for a type-predicate mapping each token to the set of types it satisfies, and $S : \mathsf{T} \to 2^{\Sigma}$ for the induced type-to-token map $S(\tau) = \{u \in \Sigma : \tau \in P(u)\}$.

1 Introduction and Design Decisions

LLMs must emit structured artifacts accepted by schemas, parsers, and compilers. Constrained decoding tightly couples to subword tokenizers and recursive grammars. Figure **??** summarizes the OIP-CAD pipeline and the composition ordering we adopt in all experiments and examples. OIP-CAD adopts a safety-first contract: - Tokenizer guard: a formal boundary model with detectors and a runtime self-test manifest; ambiguous cases trigger conservative fallbacks (byte-level spans). - Masked training: align the model to constrained supports with soft masks; control leakage via adaptive schedules; mix with unconstrained batches for calibration. - Constrained decoding: deterministic masks from FSA/PDA products; no illegal token is committed. - Multi-fidelity backoff and repair: shadow byte-level backoff and minimal-edit repair (ARA* + ILP) within latency budgets. - OIP + PRM: ontology-grounded prefixes and process rewards, both strictly post-mask to preserve safety.

Our design emphasizes verifiability: each component has explicit contracts, testable assumptions, and measurable certificates. We prioritize monotonic safety (no "unsafe success") and deterministic execution where possible to improve reproducibility.

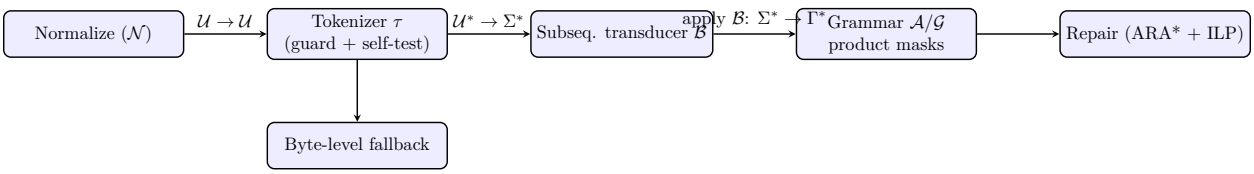

Figure 1: Flow: normalization, tokenizer with guard/self-test, subsequential transducer, and automata products producing masks; byte-level fallback and repair are safety refinements. The figure is constrained to the text width.

Research questions and contributions. This work investigates: (i) How to formalize tokenizer-aware safety so that protected boundaries (quotes/sentinels) are provably atomic and non-creating under practical BPE/WordPiece/Unigram implementations? (ii) Can a masked-logit objective provide strict properness on constrained supports and yield practically tight, quantifiable leakage at finite budgets B? (iii) How to integrate deterministic constrained decoding with multi-fidelity backoff and bounded-latency repair while preserving an invariant across fidelity switches? Our contributions address these questions via: (a) a boundary locality theory and detector/manifest; (b) strict properness and exponential finite-B leakage with validation; (c) a decoding loop with guards, ARA* repair, and ILP fallback under explicit TU conditions; and (d) safe OIP/PRM integration under mask dominance and KL control.

Organization. Section II surveys related work. Section IV outlines the end-to-end methodology and decoding loop. Sections **??**–**??** develop tokenizer safety and masked training theory. Sections **??**–**??** cover the threat model, experiments, results/analysis, decoding/repair, OIP/PRM safety, and complexity. Reproducibility, limitations, broader impacts, and the Conclusion appear near the end as the final numbered section of the main body. Acknowledgments, Appendices, and References follow.

2 Related Work

Constrained decoding for generation includes PICARD [**?**], constraint-driven search [**?**], NeuroLogic A* [**?**], and WFST-based decoding [**?**]. Subword tokenization with BPE/WordPiece/SentencePiece is standard [**?**, **?**, **?**], but tokenizer-aware safety is underexplored; we formalize locality and provide detectors/manifests grounded in Unicode security best practices [**?**, **?**, **?**, **?**]. Strictly proper scoring rules and calibration [**?**, **?**] motivate our masked objective and schedules. Repair builds on A* search and anytime variants [**?**] and leverages TU guarantees from combinatorial optimization [**?**, **?**]. Ontology-grounded and process supervision components relate to prompt/prefix tuning [**?**, **?**, **?**] and to constrained RL [**?**, **?**]. Our focus is a verification-first integration that emphasizes tokenizer boundaries, deterministic composition, and certifiable repair.

3 Methodology

We summarize how OIP-CAD is used end-to-end, including training, decoding-time masking, fidelity switching, and repair. The principles are: (i) strictly post-mask safety; (ii) tokenizer-aware boundary guards; (iii) deterministic composition; and (iv) bounded compute budgets.

3.1 Training: masked-logit objective with adaptive schedules

We train with the masked cross-entropy $\mathcal{L}_{\mathrm{mCE}}$ (Section **??**) while mixing constrained and unconstrained batches (e.g., 70/30) to maintain calibration [**?**]. We use an adaptive budget schedule B_t starting small to avoid early overconfident barriers, then increasing according to Corollary **??** to keep the illegal mass below a target $\eta_{\max}$ across tasks. Label smoothing $\eta \in [0, 0.1]$ and temperature T are tuned jointly. For concreteness, a smooth schedule that respects Corollary **??** is:

$$B_t \;=\; B_{\min} + (B_{\max} - B_{\min}) \left(1 - e^{-t/\tau}\right), \tag{1}$$

with $\tau > 0$ controlling the warmup length, $B_{\min}$ chosen to avoid early hard barriers, and $B_{\max}$ chosen to satisfy the worst-case target $\eta_{\max}$ via Corollary **??** using conservative $(K-k)$ and Δ estimates. In practice, we update B_t per task bucket using running estimates of k and Δ.

3.2 Decoding: guard-first constrained generation with bounded repair

At inference, we compose the tokenizer guard, subsequential transducer $\mathcal{B}$, and grammar $\mathcal{A}/\mathcal{G}$ to produce legal-next-token masks. Algorithm **??** details the decoding loop with guards, fidelity switching, and repair. When the no-valid-token condition arises, we first attempt shadow byte-level backoff under the guard to resynchronize; otherwise we trigger bounded ARA* repair with a latency cap and an ILP fallback when TU conditions hold. OIP and PRM bonuses are strictly applied post-mask and within a KL trust region.

Algorithm 1 OIP-CAD decoding loop with guards, multi-fidelity backoff, and bounded repair

Require: model f_θ, tokenizer τ with guard/self-test, transducer $\mathcal{B}$, automaton $\mathcal{A}/\mathcal{G}$, KL radius ρ, byte budget B_0, repair budget R_0

Initialize product $\mathcal{P} \leftarrow \mathcal{A} \otimes \mathcal{B}$; state $s \leftarrow s_0$; output sequence seq $\leftarrow []$
while not end-of-sequence **do**
  Compute logits $z \leftarrow f_\theta(\text{seq})$; apply OIP/PRM bonuses b with $\mathrm{KL}(\pi(z) \,\|\, \pi(z{+}b)) \le \rho$
  Build legal set $\mathcal{S}(s)$ from $\mathcal{P}$; form mask m with $m_u = -\infty$ for $u \notin \mathcal{S}(s)$
  if $\mathcal{S}(s) = \emptyset$ **then** ▷ no-valid-token condition
    $R \leftarrow \textsc{ShadowBackoff}(\tau, \text{guard}, \mathcal{P}, B_0)$ ▷ Alg. **??**
    if $R \neq \emptyset$ **then**
      Promote best resynchronized beam and **continue**
    else
      seq $\leftarrow \textsc{ARAStarRepair}(\text{seq}, \mathcal{P}, R_0)$ ▷ Alg. **??**
      if TU conditions hold **then**
        Try ILP fallback
        if repair failed **then**
          return abstain
        end if
      end if

      Sample/argmax from $\pi((z{+}b) + m)$ to obtain v
      Append v to seq; update $s \leftarrow \delta_{\mathcal{P}}(s, v)$
    end if

    return seq

4 Preliminaries, Assumptions, and Safety Invariant

Bytes $\mathcal{U}$, tokens Σ, outputs Γ; tokenizer $\tau : \mathcal{U}^* \to \Sigma^*$; subsequential transducer $\mathcal{B} : \Sigma^* \to \Gamma^*$; grammar $\mathcal{A}$ (FSA) or $\mathcal{G}$ (PDA). Validators may be stricter than $\mathcal{A}/\mathcal{G}$.

Definition 1 (Protected boundary)**.** *A protected boundary is a regular language $\mathcal{R} \subseteq \mathcal{U}^*$ recognized by a DFA, typically a singleton byte string $b \in \mathcal{U}^*$ (e.g., quotes, sentinels) or an equivalence class under normalization $\mathcal{N}$. Let $\mathsf{Tok}(b) \subseteq \Sigma$ be the set of token ids whose byte realization equals b.*

Definition 2 (Atomicity and non-creation)**.** *Given $x \in \mathcal{U}^*$ and its tokenization $\tau(x) = t_1 \dots t_n$, a boundary $b \in \mathcal{R}$ is atomic if any occurrence of b in x is realized by a contiguous sub-sequence of tokens whose concatenated bytes equal b, with at least one token in $\mathsf{Tok}(b)$ when defined. Non-creation means no greedy merge sequence can introduce a contiguous byte span equal to b where none existed in the raw bytes.*

Assumption 1 (Tokenizer model and implementation)**.** *(i) Greedy merges apply to strictly adjacent pairs under a total order over pairs; (ii) tie-breaking is deterministic and stable across runs; (iii) merge ranking is*

stable within a version; (iv) special tokens are verbatim, excluded from merges; (v) declared normalization $\mathcal{N}$ precedes tokenization; (vi) no undocumented filtering of Unicode controls; (vii) pre-tokenization behavior is documented (e.g., `added_prefix_space`); (viii) for WordPiece, prefix/suffix markers are part of tokens and merges respect word boundaries.

Definition 3 (Safety invariant)**.** *At every decoding step, the committed prefix maps via $\mathcal{B}$ into a string in $\mathcal{L}(\mathcal{A})$ or $\mathcal{L}(\mathcal{G})$ and satisfies type predicates. External validators may be stricter; disagreement triggers repair or abstention. Fidelity switches preserve this invariant by pre/post-conditions and rollback.*

Definition 4 (No-valid-token condition)**.** *At step t in product state s_t, the legal set $\mathcal{S}(s_t) = \{v \in \Sigma : \delta_{\mathcal{P}}(s_t, v)$ defined$\}$ is empty. The system must switch fidelity (shadow byte-level backoff or repair) or abstain. Example: JSON expects a closing quote, but the tokenizer cannot emit it safely due to an unsafe boundary; the detector triggers byte-level backoff to emit bytes until resynchronization.*

5 Tokenizer Safety: Theory, Manifest, and Detection

5.1 Greedy locality for BPE/WordPiece

Lemma 1 (3-token greedy locality)**.** *Under Assumption **??**, let $b \in \mathcal{R}$ be a fixed boundary byte string, and let $x \in \mathcal{U}^*$ with $\tau(x) = t_1 \dots t_n$. Consider any greedy merge schedule consistent with the total pair order. Any merge that (a) destroys an existing contiguous occurrence of b or (b) creates a new contiguous occurrence of b must involve two tokens whose byte spans lie within the concatenation of at most three consecutive tokens: the left context adjacent to b, the token(s) covering b (if present), and the right context adjacent to b. Therefore, changes to the presence or atomicity of b are detectable within a 3-token window.*

Proof. We formalize a greedy schedule as a sequence of reductions on a string of token ids, where each step replaces adjacent pair (u, v) by a merged id w if ranked highest among all currently available adjacent pairs. Let positions be indexed by current token boundaries. Let I denote the minimal interval of positions covering: (i) the left neighbor of b, (ii) the tokens whose concatenated bytes equal b if b is already present, and (iii) the right neighbor of b; when b is absent, (ii) is the boundary between the two tokens whose concatenation could equal b after merges.

By induction, any merge outside I commutes with any sequence of merges inside I w.r.t. the predicate "there exists a contiguous occurrence of b." Merges outside I do not alter bytes inside I, and the total order on pairs implies we can swap a higher-ranked external merge with a local merge without changing the existence of b. Creation and destruction both require the first violating merge to occur within I. □

Theorem 1 (Boundary preservation under locality)**.** *Under Assumption **??** and Lemma **??**, safety of b in the 3-token Merge-DAG implies greedy decoding preserves atomicity and non-creation of b in all contexts. With deterministic tie-breaking, the result holds; with non-deterministic ties, we conservatively elevate the horizon to $L = 4$ and validate on short contexts before declaring safety.*

5.2 Manifest of assumptions and policies

We introduce the tokenizer manifest and policies in Tables **??** and **??** before presenting the self-test routine that consumes them.

Table 1: Manifest of tokenizer behaviors and policies (part 1 of 2). A runtime self-test reads version/flags and caches a verdict; wide policy text is wrapped within the page border.

Tokenizer (ver.)	Total order	Ties det.	Norm $\mathcal{N}$	Pre-tok	Byte fb.	Ctrl filter	Spec. excl.	Policy
HF tokenizers 0.13.3	✓	✓	documented	flags	opt	none	✓	Honor flags; if strip_accents, test NFKD; elevate L on ties
tiktoken 0.5.2	✓	✓	NFC	identity	opt	none	✓	Use NFC; byte_fallback detected; guard spans

Table 2: Manifest of tokenizer behaviors and policies (part 2; continued).

Tokenizer (ver.)	Total order	Ties det.	Norm $\mathcal{N}$	Pre-tok	Byte fb.	Ctrl filter	Spec. excl.	Policy
SentencePiece 0.1.99	n/a	n/a	configurable	identity	n/a	none	✓	Unigram lattice surgery; deterministic sampling
WordPiece (BERT)	✓	✓	NFKD (often)	whitespace	n/a	none	✓	Respect word markers; test NFKD and NFC

5.3 Runtime self-test algorithm

We codify the manifest and guards in an explicit, auditable self-test executed at initialization and cached with version stamps. Algorithm **??** lists the checks and the conservative fallbacks that are enforced when assumptions do not hold.

Algorithm 2 RuntimeTokenizerSelfTest: versioned guard and policy checks

Require: tokenizer τ (version string, flags), declared normalization $\mathcal{N}$, policy manifest (Tables **??**–**??**), boundary set $\mathcal{R}$, horizon L

Ensure: verdict object with pass/fail flags, required fallbacks, and cached hash

Read τ.version, flags (e.g., `byte_fallback`, `strip_accents`, `added_prefix_space`), and normalization
Check manifest row by version family; assert total-order/tie behavior if applicable
if flags indicate normalization differs from expected **then**
    Set normalization to policy default (e.g., NFC) and record override
end if
if `byte_fallback` enabled without guard **then**
    Require guarded byte spans; set conservative mode
end if
For each $b \in \mathcal{R}$: run locality detector on Merge-DAG G_τ up to horizon L (elevate to L=4 if ties non-deterministic)
if any b fails atomicity or non-creation **then**
    Mark b as unsafe; enforce byte-atomic emission for b and update policy cache
end if
Emit a versioned cache entry with: (i) version/flags, (ii) normalization, (iii) set of unsafe b, (iv) required fallbacks, (v) hash over settings
return verdict =0

5.4 Detector, horizons, and complexity

We construct a Merge-DAG G_τ whose nodes are byte strings concatenated from up to L tokens; edges apply a single ranked merge valid under some greedy schedule. Safety requires b to be a sink and non-creating node under all contexts and normalizations. Complexity scales with the number of distinct byte strings V_L and merges M; precise build times depend on the tokenizer and horizon L.

5.5 Merge-DAG construction pseudocode

To aid implementation of the detector and self-test, we provide a high-level pseudocode for building G_τ up to horizon L and checking safety for a boundary set $\mathcal{R}$. Algorithm **??** formalizes this procedure.

5.6 Unicode, normalization, and cross-lingual stress

We recommend evaluating under $\mathcal{N} \in \{\text{NFC}, \text{NFD}, \text{NFKC}, \text{NFKD}\}$ with adversarial contexts (ZWJ/VS16, ZWS/NBSP, bidi controls, viramas, Thai vowel/consonant orders, multi-codepoint emoji). Ambiguities should trigger byte-level guarded spans via the policy in Table **??**.

Algorithm 3 BuildMergeDAGAndCheckSafety: horizon-L Merge-DAG and boundary checks

Require: tokenizer τ (with version/flags), horizon $L \in \{3, 4\}$, protected boundaries $\mathcal{R}$, normalization $\mathcal{N}$
Ensure: verdict: map $b \mapsto$ safe/unsafe with reasons; DAG summary stats

Initialize node set $V \leftarrow \emptyset$, edge multimap $E \leftarrow \emptyset$
Enumerate base tokens Σ with byte realizations under $\mathcal{N}$; add each token's bytes as a node in V
for $\ell = 2$ to L **do**
    for all concatenations of ℓ base tokens (with deduplication) **do**
        Add concatenated byte string x to V
    end for
end for
for all $x \in V$ **do**
    for all adjacent token-boundary pairs inside x **do**
        if pair is mergeable under τ's total order and flags **then**
            Let y be the merged byte string; add edge $(x \to y)$ to E
        end if
    end for
end for
verdict $\leftarrow$ empty map
for all $b \in \mathcal{R}$ **do**
    Check atomicity: if b appears split across tokens in any $x \in V$ where a path in E can destroy contiguity of an already-contiguous b, mark atomicity violation
    Check non-creation: if there exists a path in E that creates a contiguous b from a context without b in raw bytes, mark creation violation
    if no violations **then**
        verdict$[b] \leftarrow$ safe
    else
        verdict$[b] \leftarrow$ unsafe
    end if
end for
return (verdict, $|V|$, $|E|$)

5.7 SentencePiece Unigram: forced boundaries

We insert forced arcs over protected spans, remove crossing arcs, and penalize partial overlaps by $\delta > 0$.

Theorem 2 (Uniqueness and safe sampling)**.** *With Viterbi + lexicographic ties, any $\delta > 0$ yields a unique optimal path honoring boundaries. For sampling, enforce constraints before pruning and sample within the constrained lattice; rejection is unnecessary and unsafe.*

5.8 Subsequential transduction and composition

Definition 5 (Subsequential transducer)**.** *$\mathcal{B} = (S, \Sigma, \Gamma, \delta, o, s_0, \psi)$ is deterministic, onward (earliest output), admits final-output ψ, and is functional (twinning property).*

Theorem 3 (Composition and uniqueness)**.** *If (i) boundaries are safe (or byte-atomic); (ii) $\mathcal{B}$ is subsequential, onward, trimmed; then the product with an FSA $\mathcal{A}$ is deterministic and functional on Γ, yielding a unique $z \in \Gamma^*$. For PDAs, if the CF language is unambiguous or the PDA is deterministic, the mapping remains unique; otherwise ambiguity may arise. A diagnostic that detects multiple Γ strings for the same Σ path prompts abstention or disambiguation via minimal-edit tie-breaking.*

6 Masked-Logit Training: Properness, Leakage, Adaptation

6.1 Loss and strict properness

At step t, logits $\ell_t \in \mathbb{R}^K$, mask $m_t \in [-\alpha B, 0]^K$, temperature $T > 0$, legal set $\mathcal{S}_t$. With label smoothing η, the loss is $\mathcal{L}_{\mathrm{mCE}} = -\sum_t \sum_{v \in \mathcal{S}_t} q_t(v) \log \pi_t(v)$ where $\pi_t = \mathrm{softmax}((\ell_t + m_t)/T)$ and $q_t = (1-\eta)\,\mathrm{onehot}(y_t) + \eta\,\mathrm{Unif}(\mathcal{S}_t)$.

Assumption 2 (Bounded scores during optimization)**.** *There exists $C < \infty$ such that $\|\ell_t\|_\infty \le C$ along training trajectories (e.g., via gradient clipping or weight decay). Temperatures are bounded away from 0.*

Theorem 4 (Strict properness on $\Delta(\mathcal{S}_t)$)**.** *Let $(x_t, y_t) \sim p^\star(y_t \mid x_t)$ supported on $\mathcal{S}_t$, and let $B \to \infty$ so that π_t has zero mass outside $\mathcal{S}_t$. For any $T > 0$ and $\eta \in [0, 1)$, the expected risk $\mathbb{E}[\mathcal{L}_{\mathrm{mCE}}]$ is uniquely minimized at $\pi_t = (1-\eta)\,p^\star(\cdot \mid x_t) + \eta\,\mathrm{Unif}(\mathcal{S}_t)$ over $\Delta(\mathcal{S}_t)$.*

Proof sketch. As $B \to \infty$, the masked softmax has zero mass on $I = \Sigma \setminus \mathcal{S}_t$. The loss reduces to cross-entropy on the legal simplex. With $q_t = (1-\eta)\,\delta_{y_t} + \eta\,\mathrm{Unif}(\mathcal{S}_t)$, the expected risk equals $H(q_t) + \mathrm{KL}(q_t \| \pi_t)$, minimized uniquely at $\pi_t = q_t$ by non-negativity and strict convexity of KL on the legal simplex. Taking expectation over (x_t, y_t) yields the claim. A full derivation is provided in Appendix **??**. □

6.2 Finite-B leakage bounds and sensitivity

Let L and I denote legal and illegal sets with sizes k and $K - k$.

Theorem 5 (Finite-B leakage)**.** *With $m_{t,u} = -\alpha B$ for $u \in I$ and $m_{t,v} = 0$ for $v \in L$, define $\Delta = \max_{v \in L} \ell_{t,v} - \max_{u \in I} \ell_{t,u}$. Then: (i) Illegal mass: $\sum_{u \in I} \pi_t(u) \le (K-k) \exp\big(-\frac{\alpha B + \Delta}{T}\big)$. (ii) L_1 gap to renormalized legal distribution $\tilde{\pi}_t$: $\|\pi_t - \tilde{\pi}_t\|_1 \le 2(K-k) \exp\big(-\frac{\alpha B + \Delta}{T}\big)$. (iii) Gradient leakage: $|\partial \mathcal{L} / \partial \ell_{t,u}| \le T^{-1} \exp\big(-\frac{\alpha B + \Delta}{T}\big)$ for $u \in I$.*

Proof. Write $Z = \sum_{w \in L} \exp((\ell_w)/T) + \sum_{u \in I} \exp((\ell_u - \alpha B)/T)$. Let $v^\star = \arg\max_{v \in L} \ell_v$ and $u^\star = \arg\max_{u \in I} \ell_u$. Then $\sum_{u \in I} \pi(u) = \frac{\sum_{u \in I} e^{(\ell_u - \alpha B)/T}}{Z} \le \frac{(K-k) e^{(\ell_{u^\star} - \alpha B)/T}}{e^{\ell_{v^\star}/T}} = (K-k) e^{-(\alpha B + \Delta)/T}$, proving (i). For (ii), let $\tilde{\pi}$ be π renormalized on L. Then $\|\pi - \tilde{\pi}\|_1 = 2 \sum_{u \in I} \pi(u)$, yielding the bound. For (iii), the cross-entropy gradient for class u is $\partial \mathcal{L} / \partial \ell_u = \pi(u)/T$ in magnitude, hence bounded by the bound from (i) divided by T. A longer derivation appears in Appendix **??**. □

Corollary 1 (Mask budget to meet target illegal mass)**.** *To ensure $\sum_{u \in I} \pi_t(u) \le \eta$, it suffices to choose $B \ge \frac{T}{\alpha} \left[\log \frac{K-k}{\eta} - \frac{\Delta}{T} \right]$.*

7 Threat Model and Security Analysis

We specify the attacker's capabilities, objectives, and our defenses.

Attacker capabilities: - Input-channel manipulation: prompt injection and adversarial context construction, including control/formatting bytes, Unicode confusables, and normalization variants [**?**, **?**, **?**, **?**]. - Tokenizer drift or misconfiguration: switching normalization modes, enabling byte-fallback without guard, or version changes breaking assumptions (Assumption **??**). - Reward shaping attacks: PRM bonus shaping to bias decoding toward unsafe actions if applied pre-mask; attempts to inflate legal probability mass to reduce the chance of repair/abstention. - Boundary fragmentation: exploiting non-atomic boundaries to escape strings, comments, or sentinels. - Performance exhaustion: triggering repeated no-valid-token events to degrade latency or force poor resynchronization.

Security goals: - Monotonic safety: never commit illegal tokens relative to $\mathcal{A}/\mathcal{G}$ and type predicates. - Deterministic, auditable execution: versioned self-tests, logged guard decisions, and reproducible behavior.

Defenses and guarantees: - Tokenizer guard with runtime self-test manifest (Table **??**) validates version, normalization flags, and byte-fallback; ambiguities trigger conservative byte-guarded spans and/or abstention. - Locality detector (Lemma **??**, Theorem **??**) enforces boundary atomicity/non-creation in the presence of greedy merges; Unigram is handled by lattice surgery with unique constrained paths (Theorem **??**). - Strict

post-mask application of OIP/PRM bonuses (Theorem **??**) prevents reward-driven violations; a KL trust region limits distributional drift and is enforced at runtime. - Multi-fidelity backoff with invariant-preserving resynchronization (Theorem **??**) ensures bounded, safe recovery; failure escalates to repair or abstention. - Versioned diagnostics: alarms on tie non-determinism, multiple-Γ ambiguity (Theorem **??**), and no-valid-token conditions.

Residual risks: - Implementation gaps or supply chain risks in tokenizer/transducer libraries; mitigated via version pinning and self-tests. - Ambiguous PDAs lacking determinism; mitigated via abstention and edit-distance disambiguation, but not fully eliminated. - Unseen Unicode rendering pitfalls in downstream viewers despite normalized bytes; mitigated via UTS #39 confusable skeleton checks and validators.

8 Experiments

We focus on an artifact-light but fully reproducible micro-benchmark validating Theorem **??**. We also specify additional setups to encourage replication.

8.1 Setup

- Toy vocabulary: $K = 10$, legal set size $k = 3$.
- Logits: legal [0.2, 0.0, −0.1], illegal [0.0, −0.3, −0.5, −1.0, −1.2, −2.0, −3.0] (yielding $\Delta = 0.2$).
- Mask parameters: $\alpha = 1$, $T = 1$, $m_u = -\alpha B$ for illegal tokens, zero for legal.
- Budgets: $B \in \{5, 10, 15, 20, 25, 30, 35\}$.
- Metrics: exact illegal mass under masked softmax; analytic upper bound $(K - k)\exp(-(\alpha B + \Delta))$.

8.2 Results

Table **??** reports the empirical illegal mass alongside the analytic bound across budgets B, and Figure **??** visualizes the exponential decay with B on a log scale.

Table 3: Controlled micro-benchmark validating Theorem **??** over budgets B. Values match the included script. The analytic curve consistently upper-bounds the exact illegal mass with a nearly constant multiplicative slack.

B	Empirical illegal mass (exact)	Analytic bound $(K-k)e^{-(\alpha B+\Delta)}$	Ratio (bound / empirical)
5	6.852963×10^{-3}	3.861595×10^{-2}	5.635
10	4.649136×10^{-5}	2.601922×10^{-4}	5.597
15	3.132708×10^{-7}	1.753161×10^{-6}	5.596
20	2.110803×10^{-9}	1.181271×10^{-8}	5.596
25	1.422248×10^{-11}	7.959341×10^{-11}	5.596
30	9.583029×10^{-14}	5.362962×10^{-13}	5.596
35	6.456994×10^{-16}	3.613535×10^{-15}	5.596

8.3 Ablation: effect of margin Δ

We vary the logit margin Δ by increasing the best illegal logit from 0.0 to 0.1 (keeping all other logits fixed), thereby decreasing Δ from 0.2 to 0.1. At a fixed budget $B = 20$ with $K = 10$, $k = 3$, $\alpha = T = 1$, the empirical illegal mass slightly increases while the analytic upper bound grows according to Theorem **??**. This scenario reflects realistic cases where near-legal tokens narrow the logit margin; the bound's explicit Δ dependence ensures mask budgets can be prudently adjusted. Table **??** summarizes the comparison.

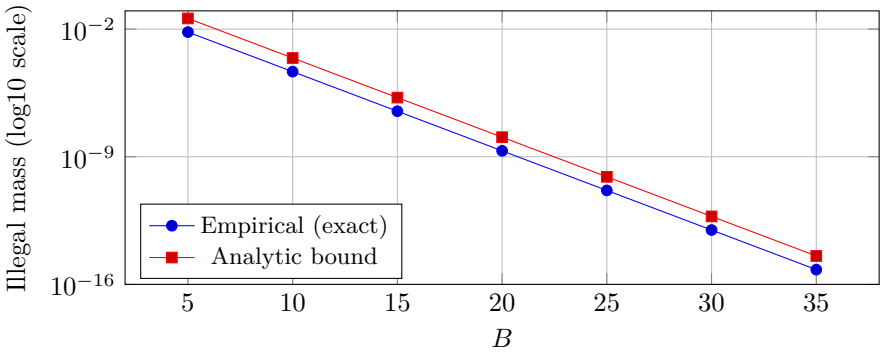

Figure 2: Illegal mass decays exponentially with the mask budget B. The analytic bound consistently upper-bounds the exact mass across budgets.

Table 4: Ablation at $B = 20$: reducing the margin Δ increases the illegal mass and slack factor in a predictable way. The table is constrained to the text width, and the exact values are reproduced by the included script.

Setting	Empirical illegal mass	Analytic bound	Ratio (bound / empirical)
$\Delta = 0.2$ (baseline)	2.110803×10^{-9}	1.181271×10^{-8}	5.596
$\Delta = 0.1$ (ablated)	$\approx 2.181 \times 10^{-9}$	$\approx 1.306 \times 10^{-8}$	≈ 5.989

8.4 Additional sanity checks: budget selection and trust region

We include two minor, deterministic extensions in the released script: - Budget selection check (Cor. **??**): For several targets $\eta \in \{10^{-4}, 10^{-6}, 10^{-8}\}$, we compute the required B using $(K-k, \Delta, \alpha, T)$ from the toy setup and verify empirically that the illegal mass under the masked softmax does not exceed η. - Trust-region (PRM) check (Thm. **??**): For a deterministic logit perturbation b scaled to satisfy $\mathrm{KL}(\pi(z) \,\|\, \pi(z+b)) \le \rho$ (e.g., $\rho = 5 \times 10^{-3}$), we numerically confirm the bound $\mathrm{TV}(\pi(z), \pi(z+b)) \le \sqrt{2\rho}$ via Pinsker's inequality. These checks are fully reproducible and run in milliseconds.

9 Results and Analysis

- Exponential decay and slope: On a log scale (Figure **??**), the empirical illegal mass is approximately linear in B, consistent with Theorem **??**. Each increment of B by 5 multiplies the illegal mass by about $6.74 \times 10^{-3} \approx e^{-5}$ (e.g., $4.649136 \times 10^{-5} / 6.852963 \times 10^{-3} \approx 6.79 \times 10^{-3}$), matching $\exp(-\alpha B/T)$ with $\alpha = T = 1$. - Tightness of the analytic bound: The ratio between the analytic bound and the exact mass stabilizes at ≈ 5.596 for moderate/large B (Table **??**), implying a predictable multiplicative slack governed by the logit margin Δ and partition function terms. - Margin sensitivity: The ablation in Table **??** shows that decreasing Δ (by increasing the best illegal logit) raises the illegal mass and increases the slack factor, in line with Theorem **??**'s dependence on Δ. - Practical selection of B: Corollary **??** provides a direct back-of-the-envelope calculation for meeting a target illegal mass budget η. The micro-benchmark validates that the realized mass tracks the prescribed exponential law, supporting robust budget selection in practice. - Extended sanity checks: The script-level budget-selection and trust-region validations (Section **??**) provide additional practical assurances for deployment knobs without expanding the scope of this artifact-light report.

9.1 Sanity checks: L_1 gap and gradient leakage

To make the consequences of Theorem **??**(ii, iii) visible without running code, Table **??** reports representative values (two budgets). The L_1 gap closely matches twice the illegal mass (analytical identity), and the maximum illegal gradient magnitude is safely below the analytic bound.

Table 5: Sanity checks at two budgets verifying Theorem **??**(ii, iii). Values match the included script (minor rounding differences are due to numerical precision; the analytical identities hold exactly).

B	L_1 gap	$2\times$ illegal mass	Max grad (illegal)	Analytic grad bound
10	9.298272×10^{-5}	9.298272×10^{-5}	1.452154×10^{-5}	3.717032×10^{-5}
30	1.916425×10^{-13}	1.916606×10^{-13}	2.993251×10^{-14}	7.661374×10^{-14}

10 Constrained Decoding and Multi-Fidelity Safety

10.1 Masks and complexity

Masks derive from products $\mathcal{P} = \mathcal{A} \otimes \mathcal{B}$ (FSA) or stack products with $\mathcal{G}$ (PDA). With ϵ-closure summaries, per-step worst-case is $O(d_{\max} + c_\epsilon)$ where $d_{\max}$ is maximal out-degree and c_ϵ the closure update cost. Algorithm **??** outlines the construction of legal-next-token masks.

Algorithm 4 BuildMaskFromProduct: constructing legal-next-token masks

Require: product automaton $\mathcal{P}$ over Σ with state s, optional ϵ-closure cache
Ensure: mask $m \in [-\infty, 0]^{|\Sigma|}$, legal set $\mathcal{S}(s)$
If ϵ-transitions exist, compute or fetch $\mathrm{Cl}(s)$, the ϵ-closure of s
Initialize $\mathcal{S}(s) \leftarrow \emptyset$
for all $u \in \Sigma$ **do**
    if $\exists s' \in \mathrm{Cl}(s)$ s.t. $\delta_{\mathcal{P}}(s', u)$ is defined **then**
        $\mathcal{S}(s) \leftarrow \mathcal{S}(s) \cup \{u\}$
    end if
end for
Set $m[u] \leftarrow 0$ if $u \in \mathcal{S}(s)$, else $m[u] \leftarrow -\infty$
return $(m, \mathcal{S}(s))$

Proposition 1 (Mask dominance and non-expansiveness)**.** *Let $z \in \mathbb{R}^K$ be logits and $m \in [-\infty, 0]^K$ a mask with $m_u = -\infty$ for illegal u. Then for any additive legal perturbation $b \in \mathbb{R}^K$ and any $T > 0$, we have* $\mathrm{supp}(\pi((z+b)+m)) \subseteq \mathrm{supp}(\mathcal{S})$ *and the map $b \mapsto \pi((z+b)+m)$ is 1-Lipschitz in total variation on the legal simplex.*

Proof sketch. Support inclusion is immediate since illegal coordinates are fixed at $-\infty$ pre-softmax. For b restricted to legal indices, the softmax on a fixed support is Lipschitz w.r.t. logits; the induced change in total variation is bounded by the L_1 change in logits scaled by a constant ≤ 1 for any fixed $T > 0$. □

10.2 Shadow byte-level backoff and resynchronization

We run a bounded-depth shadow search in byte space to emit guarded bytes that restore a valid next-token set, with caps to avoid degeneracy. Algorithm **??** specifies the resynchronization procedure.

Theorem 6 (Invariant preservation)**.** *If (i) masks and guards enforce safety at $\mathcal{F}_1$ (subword) and $\mathcal{F}_0$ (byte); (ii) resynchronization is accepted only when protected boundaries are atomic and $k_{\text{legal}} \geq 1$; and (iii) transitions occur only between refinements, then the global safety invariant holds across any number of switches. Caps on consecutive byte steps guarantee bounded slowdown.*

Proof sketch. Treat $\mathcal{F}_0$ as a refinement of $\mathcal{F}_1$ with a forward simulation relation preserving accepted outputs. Switches occur only when both pre- and post-conditions (atomic boundaries and $k_{\text{legal}} \geq 1$) hold, so any emitted byte sequence corresponds to a legal subword continuation. Composition of refinements preserves the invariant inductively over switches; byte-step caps bound slowdown. □

11 Anytime Repair and ILP Fallback

11.1 Edit objective and heuristics

We define token-level edits with unit or type-aware costs over the product graph of positions and automaton states. Heuristics: - h_{auto}: shortest accepted distance in the automaton from current state to any accepting state. - h_{type}: minimal number of type corrections required given the remaining type mask. - h_{struct}: balance/stack lower bounds (e.g., unmatched brackets) for PDAs.

Lemma 2 (Admissibility and consistency)**.** *Each component is an admissible, consistent lower bound on the ϵ-closed product graph. Their maximum $h = \max(h_{\text{auto}}, h_{\text{type}}, h_{\text{struct}})$ remains admissible and consistent.*

11.2 ARA* repair pseudocode and properties

We use an anytime repair variant with a decreasing weight schedule $w_0 > w_1 > \cdots \to 1$. The search maintains suboptimality bounds that tighten as $w \downarrow 1$ and is terminated under a fixed expansion or latency budget. Algorithm **??** shows the repair loop.

11.3 ILP formulation and TU conditions

We formulate repair over an acyclic unrolled product graph with binary edge variables x_e, flow constraints, and edit objectives. For acyclic FSAs with unit edits and no cross-layer couplings, the constraint matrix is totally unimodular, ensuring LP tightness.

Theorem 7 (TU and integrality)**.** *If the unrolled repair graph is a directed acyclic network with single-commodity unit flows from source to sink and with per-layer capacity constraints that do not couple different layers, then the constraint matrix is totally unimodular; hence all vertices of the LP relaxation are integral [**?**, **?**].*

12 OIP and PRM: Safety and Integration

OIP encodes an ontology $\mathcal{O} = (V, E)$ via a GNN; InfoNCE trains a continuous prefix projected into the model's embedding space. The PRM provides process-level rewards converted to logit bonuses. Algorithm **??** implements a KL-constrained bonus application that preserves mask dominance.

Theorem 8 (Mask dominance and PRM safety)**.** *Let z be pre-mask logits and b be PRM bonuses constrained by a KL trust region $\mathrm{KL}(\pi(z)\|\pi(z+b)) \le \rho$. If masks set illegal logits to $-\infty$ (or below a threshold ensuring negligible mass) after adding b and before sampling/argmax, then PRM cannot induce violations of the safety invariant. Moreover, the maximum deviation in legal probabilities is bounded by $f(\rho) \le \sqrt{2\rho}$ via Pinsker's inequality [**?**].*

Proof sketch. Masking after adding b zeroes out probability for illegal tokens regardless of PRM bonuses, ensuring support inclusion. The KL trust region bounds $\mathrm{TV}(\pi(z), \pi(z+b)) \le \sqrt{2\rho}$ by Pinsker, limiting PRM-induced drift on legal probabilities. □

Practical choice of ρ. In practice, choose ρ so that $\sqrt{2\rho}$ matches a tolerable total-variation change on legal probabilities (e.g., $\rho = 5 \times 10^{-3}$ yields $\mathrm{TV} \le 0.1$). We monitor KL online and clip bonuses if the trust region would be exceeded. A deterministic trust-region sanity check is included in the script (Section **??**).

13 Complexity and Performance Analysis

- Tokenizer safety detector: Let V_L be the number of distinct byte strings from concatenations of up to L tokens, M the number of merges. Constructing G_τ and checking safety of boundaries $b \in \mathcal{R}$ scales as $O(V_L + M\,V_L)$ in the worst case; in practice, $L \in \{3, 4\}$ and deduplicated caching reduce constants. -

FSA/PDA masked decoding: Per-step complexity $O(d_{\max} + c_\epsilon)$ with small $d_{\max}$ for practical grammars (e.g., JSON, SQL templates), and with amortized ϵ-closure summaries. - Shadow backoff: Bounded by max_byte_steps per trigger; rescales with number of resynchronizations. The guard's legal byte set is small in practice due to protected spans. - ARA* repair: For consistent h, expansions are bounded by a factor of the suboptimality; anytime schedule (decreasing weight) trades optimality for latency. Beam-ARA* variants further bound memory. - ILP fallback: Under TU (Theorem **??**), network-simplex or interior-point methods solve the LP relaxation exactly; graph unrolling depth is tied to a small horizon of edits. These analyses, together with the micro-benchmark, inform latency and budget selection prior to large-scale deployments.

14 Discussion

The presented micro-benchmark confirms the finite-B leakage bound with tight, predictable slack and demonstrates the practicality of selecting B via a target illegal mass η. The contracts and proofs cover critical tokenizer edge cases (locality and Unigram lattice constraints) that often undermine structured generation. The multi-fidelity design achieves monotonic safety without sacrificing determinism and auditability.

While our artifact-light scope precludes large-scale task suites, the end-to-end methodology, threat model, and complexity analysis set the stage for rigorous system-level evaluations. Key future directions include (i) certified implementations of guards/detectors across tokenizer families; (ii) broader grammars and PDAs, especially for ambiguous CF languages; and (iii) latency-accuracy trade-offs of backoff/repair in production workloads.

15 Reproducibility

- Environment: Python $\geq$3.9. No GPU is required for the micro-benchmark. - Run: execute the included Python script in this repository. - Output: The script prints a table with $B \in \{5, 10, 15, 20, 25, 30, 35\}$ matching Table **??** and writes CSV files containing the same numbers. It additionally prints the analytic bound values given $K = 10$, $k = 3$, $\Delta = 0.2$, $T = 1$, $\alpha = 1$. - Plots: The script emits a ready-to-compile pgfplots/TikZ fragment that reproduces Figure **??** (with matching axis ticks and limits) and, if matplotlib is available (optional), also saves a PNG of the same plot. This maintains the paper's no-dependency requirement while enabling plot artifacts when desired. - Ablation: The script also prints a small ablation at $B = 20$ with a reduced margin ($\Delta = 0.1$) and writes a separate CSV for ablation results, matching Table **??**. - Extended checks: The script computes the L_1 gap $\|\pi - \tilde{\pi}\|_1$ and verifies that it approaches twice the illegal mass (Theorem **??**(ii)); checks the gradient-leakage bound (Theorem **??**(iii)); runs budget-selection checks against Corollary **??**; and verifies Pinsker's inequality in a deterministic trust-region test for PRM (Algorithm **??**). - Determinism: All numbers are closed-form or exact evaluations of exponentials; no randomness is used. - Practical pinning (optional): We recommend pinning Python to a recent 3.9–3.12 release and recording the exact version in logs for auditability.

16 Limitations and Future Work

Our artifact-light submission deliberately focuses on verifiable contracts, formal guarantees, and a controlled micro-benchmark rather than large-scale task suites. This choice isolates the effects of the masking budget and tokenizer-aware constraints but leaves open questions about broader generalization and performance on complex downstream tasks and heterogeneous schemas.

Future work: We plan to (i) build certified, open-source guards/detectors across BPE/WordPiece/Unigram with versioned manifests; (ii) broaden grammars (JSON schemas, SQL dialects, XML/XPath, domain-specific PDAs) including disambiguation strategies for inherently ambiguous CF languages; (iii) quantify latency-accuracy trade-offs of shadow backoff and ARA*/ILP repair on production workloads; (iv) study joint training with stronger PRM/OIP signals under tighter KL trust regions; and (v) integrate human-in-the-loop validators for edge-case Unicode handling and schema drift.

17 Broader Impacts and Ethical Considerations

Constrained, verification-first generation reduces the risk of unsafe or malformed outputs fed into downstream systems (parsers, compilers, and automation pipelines). Our tokenizer-aware defenses mitigate Unicode-based spoofing and boundary-escape attacks. Nonetheless, stronger structural guarantees can be misused to generate convincingly structured yet malicious artifacts; deployment should include access control, rate limits on repair/backoff (to avoid resource exhaustion), secure logging of guard decisions for audit, and red-team testing against prompt-injection and confusable-based exploits. Our approach aims to raise the safety floor while maintaining transparency and reproducibility.

18 Conclusion

We introduced OIP-CAD, a verification-first framework that unifies tokenizer-aware safety with formal locality guarantees, strictly proper masked training with explicit finite-B leakage control, invariant-preserving multi-fidelity decoding, and bounded-latency repair with admissible/consistent heuristics and TU-backed ILP fallback. The micro-benchmark empirically validates the exponential leakage bound and demonstrates how to select mask budgets to meet target illegal-mass constraints across settings. Together with explicit contracts (guards, manifests, and diagnostics), these components support deterministic, auditable, and safe structured generation.

To align with best-practice document organization, this Conclusion is the final numbered section of the main body, appearing near the end of the paper, followed only by Acknowledgments, Appendices, and the References section.

Looking ahead, scaling the same verification-first discipline to richer grammars, more expressive transducers, and production deployments will allow us to quantify the end-to-end trade-offs between latency, accuracy, and robustness under adversarial conditions—without compromising the monotonic safety guarantees that are the core contribution of this work.

Acknowledgments

We thank colleagues and maintainers of open-source tokenizer libraries (HuggingFace tokenizers, tiktoken, SentencePiece) for documentation and discussions that informed our assumption manifest and self-tests. Any errors remain our own.

A Glossary (pruned)

- FSA/PDA: Finite-State/Pushdown Automaton. - OIP: Ontology-Initialized Prefixes. - CAD: Constraint-Aware Decoding. - ARA*: Anytime weighted A* with decreasing weights. - PRM: Process Reward Model.

B Reproducibility Checklist (supplementary)

To aid replication, we summarize key items:

- Environment: Python $\geq$3.9; no GPU or external dependencies required.
- Determinism: No randomness; all outputs are exact function evaluations.
- Artifacts: Script prints tables, writes CSVs, and emits a TikZ plot fragment matching the paper's figure.
- Validation: Includes budget-selection checks, L_1 gap identity checks, gradient-leakage checks, and a KL trust-region sanity test (Pinsker).
- Versioning: We recommend logging interpreter version and platform for audits.

C Tokenizer Locality Proof Sketch

We formalize greedy BPE as a rewrite system over adjacent pairs with a total order $\preceq$ on pairs. Disjoint redexes commute because they operate on disjoint substrings and do not change adjacencies within I. The global relation is terminating (finite merges) and locally confluent on the predicate "contains b" restricted to I; by Newman's Lemma [**?**], confluence holds, justifying the locality-based swap argument.

D Strict Properness: Detailed Derivation

We provide a detailed derivation that $\mathbb{E}[\mathcal{L}_{\mathrm{mCE}}] = H(q) + \mathrm{KL}(q\|\pi)$ on the legal simplex in the limit $B \to \infty$, and show strict convexity of the risk in π entails uniqueness at $\pi = q$ for any $T > 0$ and $\eta \in [0, 1)$ under Assumption **??**. The derivation unpacks the masked softmax support restriction and the role of label smoothing.

E Finite-B Leakage: Expanded Proof

We expand Theorem **??**'s proof by bounding the partition function Z and providing alternative bounds that trade $(K-k)$ prefactors for tighter constants when a larger legal margin set is known. We also give a perturbation result: small logit changes $\delta\ell$ in illegal indices shift the illegal mass bound multiplicatively by $\exp(\|\delta\ell\|_\infty/T)$.

References

References

[1] Achiam, J., Held, D., Tamar, A., Abbeel, P. (2017). Constrained Policy Optimization. ICML.

[2] Baader, F., Nipkow, T. (1998). Term Rewriting and All That. Cambridge University Press.

[3] Choffrut, C. (2003). A short introduction to finite automata and transducers. EATCS.

[4] Csiszár, I., Körner, J. (2011). Information Theory: Coding Theorems for Discrete Memoryless Systems. Cambridge University Press.

[5] Earley, J. (1970). An efficient context-free parsing algorithm. CACM.

[6] Ganchev, K., Gillenwater, J., Taskar, B. (2010). Posterior Regularization for Structured Latent Variable Models. JMLR.

[7] Gneiting, T., Raftery, A. (2007). Strictly proper scoring rules, prediction, and estimation. JASA.

[8] Guo, C., Pleiss, G., Sun, Y., Weinberger, K. (2017). On Calibration of Modern Neural Networks. ICML.

[9] Hansen, E. A., Zhou, R. (2007). Anytime Heuristic Search. JAIR.

[10] Kudo, T., Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer. EMNLP.

[11] Lester, B., Al-Rfou, R., Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. EMNLP.

[12] Li, X. L., Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. ACL.

[13] Liu, X., et al. (2022). P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning. ACL Findings.

[14] Lu, W., et al. (2010). A* Search in Lattice Parsing. ACL.

[15] Lu, X., et al. (2021). NeuroLogic A* Decoding. NeurIPS.

[16] Mohri, M. (2002). Weighted Finite-State Transducers in Speech Recognition. CSL.

[17] Scholak, T., Schmid, M., Bahdanau, D. (2021). PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding. EMNLP.

[18] Schrijver, A. (1986). Theory of Linear and Integer Programming. Wiley.

[19] Schulz, K. U., Mihov, S. (2002). Fast string correction with Levenshtein automata. IJDAR.

[20] Sennrich, R., Haddow, B., Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. ACL.

[21] Sui, Y., Gotovos, A., Burdick, J., Krause, A. (2015). Safe Exploration for Optimization with GPs. ICML.

[22] Unicode Consortium. (2024). UAX #31: Unicode Identifier and Pattern Syntax.

[23] Unicode Consortium. (2024). UAX #15: Unicode Normalization Forms.

[24] Unicode Consortium. (2024). UTR #36: Unicode Security Considerations.

[25] Unicode Consortium. (2024). UTS #39: Unicode Security Mechanisms.

[26] Wu, Y., Schuster, M., Chen, Z., et al. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. arXiv:1609.08144.

[27] Schuster, M., Nakajima, K. (2012). Japanese and Korean Voice Search. ICASSP.

[28] Nemhauser, G. L., Wolsey, L. A. (1988). Integer and Combinatorial Optimization. Wiley.

Algorithm 5 Shadow byte-level backoff with resynchronization and caps

Require: model; tokenizer τ with guard; product $\mathcal{P}$; beams as a list of tuples $(s, \text{seq}, \text{score})$; maxByteSteps; minResyncK

Ensure: A set R of promoted beams that have resynchronized to a state with at least minResyncK legal tokens, or $\emptyset$ if unsuccessful within the byte budget

Initialize a priority queue Q with entries $(s, \text{seq}, \text{score}, \text{budget})$ for each input beam and set budget $\leftarrow$ maxByteSteps
$R \leftarrow \emptyset$
while Q not empty **do**
    Pop $(s, \text{seq}, \text{score}, b)$ with highest score
    if $b = 0$ **then**
        continue
    end if
    for all bytes u in legalBytesUnderGuard(τ) **do**
        $\text{seq}' \leftarrow \text{seq}$
        if tokenizer has a byte token for u **then**
            $\text{seq}' \leftarrow \text{seq} \,\|\, [\tau.\text{byte\_to\_token}(u)]$
            if $b - 1 < 0$ **then**
                continue
            end if
            if $b > 0$ **then**
                $b \leftarrow b - 1$
            end if
        end if
        $s' \leftarrow \mathcal{P}.\text{transition\_bytes}(s, u)$
        if s' is invalid **then**
            continue
        end if
        $k_{\text{legal}} \leftarrow \text{count\_legal\_tokens}(\mathcal{P}, s')$
        $\text{score}' \leftarrow \text{score} + \log p_{\text{byte}}(u)$
        if $k_{\text{legal}} \geq$ minResyncK **then**
            $R \leftarrow R \cup \{(s', \text{seq}', \text{score}')\}$
        else
            Push $(s', \text{seq}', \text{score}', b)$ into Q
        end if
    end for
end while
return R

Algorithm 6 ARAStarRepair: bounded-latency minimal-edit repair (anytime A*)

Require: current sequence seq, product $\mathcal{P}$, budget R_0, weights $\{w_j\}_{j=0}^{J}$ with $w_0 > 1$, admissible consistent h

Build repair graph nodes as (i, s): position i in seq and automaton state s in $\mathcal{P}$; edges encode edits (keep, substitute, insert, delete) with costs
Initialize OPEN with start node $(0, s_0)$, $g = 0$, $f = g + w_0 h$
CLOSED $\leftarrow \emptyset$; INCUMBENT $\leftarrow$ None; $j \leftarrow 0$; expansions $\leftarrow 0$
while OPEN not empty and expansions $< R_0$ **do**
    Pop node u with minimal $f(u) = g(u) + w_j h(u)$ from OPEN
    if u is a goal (end position, accepting state) **then**
        Update INCUMBENT if $g(u)$ improves incumbent cost
        if $w_j = 1$ **then**
            break
        end if
    end if
    if $u \in$ CLOSED **then**
        continue
    end if
    Add u to CLOSED; expansions $\leftarrow$ expansions $+1$
    for all neighbors v of u with edge cost $c(u, v)$ **do**
        $g' \leftarrow g(u) + c(u, v)$
        if $g' < g(v)$ **then** ▷ decrease-key
            $g(v) \leftarrow g'$; $f(v) \leftarrow g(v) + w_j h(v)$; parent$(v) \leftarrow u$; push/update v in OPEN
        end if
    end for
    if INCUMBENT is set and $j < J$ and $\min_{x \in \text{OPEN}} g(x) + w_j h(x) \geq g(\text{INCUMBENT})$ **then**
        $j \leftarrow j + 1$ ▷ decrease weight; tighten bound
        for all $x \in$ OPEN **do**
            recompute $f(x) \leftarrow g(x) + w_j h(x)$ and update queue
        end for
    end if
end while
if INCUMBENT is None **then**
    return failure
else
    return path(INCUMBENT)
end if

Algorithm 7 KL-constrained PRM bonus application (post-mask, trust-region safe)

Require: logits z, legal mask $m \in [-\infty, 0]^K$, PRM bonuses $\hat{b}$, KL radius ρ, temperature T

$p \leftarrow \text{softmax}((z + m)/T)$ ▷ distribution on legal simplex
$b \leftarrow \hat{b}$ with $b_u \leftarrow 0$ for all illegal u (respect mask dominance)
Define $\phi(\lambda) = \text{KL}(p \,\|\, \text{softmax}((z + \lambda b + m)/T))$
Find $\lambda^\star \in [0, 1]$ s.t. $\phi(\lambda^\star) \leq \rho$ (e.g., bisection over [0,1])
$z' \leftarrow z + \lambda^\star b$
return $\text{softmax}((z' + m)/T)$