Mitigating Bias in Locally Constrained Decoding via Tractable Proposals

Meihua Dang[1] Linxin Song[2] Honghua Zhang[3] Jieyu Zhao[2] Guy Van den Broeck[3] Stefano Ermon[1]
[1]Stanford University [2]University of Southern California [3]University of California, Los Angeles

The Problem

Locally Constrained Decoding (LCD)

LCD masks tokens that **immediately violate** a constraint at each step, but it's **myopic and locally greedy.**

Example: tool call with JSON-schema as a constraint

```
{ "name": "get_weather",
  "args": { "loc": "NYC",
  "extra": { { { { ...
```

⚠ *Valid at each step, but final output is not valid JSON!*

LCD has two failure modes

1. **No Token Budget Guarantee**: may not produce a valid output given a token budget (max_tokens=k).
2. **Biased Sampling**: local renormalization distorts the LM distribution.

Our Solution

1. **GCD:** computes next-token mask taking token budget into account.
2. **P-GCD:** multiplies GCD with a distribution to incorporate probabilistic look-ahead.
3. We combine both GCD and P-GCD with Sequential Monte Carlo (SMC) Sampling, aiming for fast convergence to unbiased sampling.

Globally Constrained Decoding (GCD)

Key idea: encode the constraint as **finite automata,** tensorize it, then run **forward + backward** passes to compute valid-next-token mask m_t.

Tensorized Finite Automata

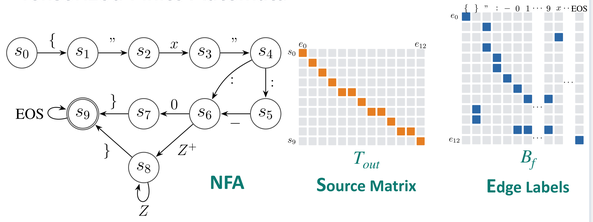


Fig. 1. An NFA corresponding to a JSON schema of the form {"x": <integer>} and its tensorized representations.

Forward pass: states reachable after observing the prefix $x_{1:t-1}$, for all tokens at x_t

$$\alpha_t(\cdot,\cdot) = \mathbf{1}\left\{T_{in}^{\top}\left(T_{out}^{\top}\cdot\alpha_{t-1}(\cdot,x_{t-1})\circ B_f\right)\right\}$$

Backward pass: states reaching accept states at exactly $n-t$ step

$$\beta_{t-1} = \mathbf{1}\left\{T_{out}\left((T_{in}\beta_t)\circ(B_f\mathbf{1}_V)\right)\right\},$$

Valid mask: whether current state can reach accept state for each token

$$m_t = \mathbf{1}\left\{\alpha_t^{\top}\beta_t\right\} \in \{0,1\}^V$$

Probabilistic GCD (P-GCD)

GCD mask m_t is only binary: By introducing an **HMM approximation for the base LM** ($p_{hmm} \approx p_{lm}$), P-GCD weighs **how likely** each next-token will lead to constraint satisfaction.

Circuit Multiplication (Theorem 4.4)

HMM p_{hmm} (A_h, B_h) × FA p_{fa} $(A_f = T_{in}T_{out}, B_f)$ ↓ **Product HMM** $p_{prod} \propto p_{hmm} \cdot \mathbf{1}\{x_{1:n} \in \alpha\}$

In particular,
$\mathbf{1}\{p_{prod}(x_t \mid x_{<t}) > 0\} = m_t(x_t)$;
or intuitively,
$p_{prod}(x_t \mid x_{<t}) = m_t(x_t) \cdot p(x_t$ leads to constraint satsification$)$

(P-)GCD Proposals for SMC Sampling

GCD proposal: $q(x_t \mid x_{<t}) \propto p_{lm}(x_t \mid x_{<t}) \cdot m_t$
P-GCD proposal: $q(x_t \mid x_{<t}) \propto p_{lm}(x_t \mid x_{<t}) \cdot p_{prod}(x_t \mid x_{<t})$

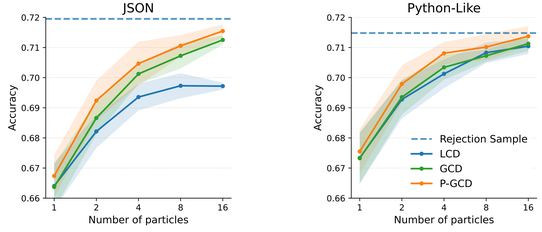
1. Autoregressive sampling from both proposals guarantees constraint satisfaction given a token budget.
2. Employ SMC Sampling to resolve failure mode 2:
   - Maintain k "particles" (like "beams" in beam search)
   - At each step,
     - sample one next token for each particle
     - update the weight of each particle using proposal
     - resample k particles w/ replacement prop to weight

(P-)GCD + SMC pipeline

1. Precomputing β_t from n to 1, the total probability of future valid completions $x_{t+1:n}$
2. Generating from $t = 1$ to n
   1. Compute α_t from α_{t-1} and x_{t-1}, the probability of reaching current states $x_{1:t}$
   2. Compute $q(x_t \mid x_{<t})$ and sample
   3. SMC: reweight and resample

Experiments

1. P-GCD converges faster with fewer SMC particles



(a) xLAM with JSON format *(b)* xLAM with Python-like format

Fig. 2. Accuracy on xLAM as a function of the number of SMC particles.

2. (P-)GCD guarantees constraint satisfaction

	JSON		Python-like	
Method	k=4	k=16	k=4	k=16
LCD	95.26	95.40	98.12	99.20
GCD	100.0	100.0	100.0	100.0
P-GCD	100.0	100.0	100.0	100.0

Tab. 1. Constraint satisfaction rates.