Constraint-Lattice Decoding with Posterior Certificates for Auditable Long-Context Autoregressive Reasoning

Rahul Bikram Sharma[1] and Sandeep Kumar Chauhan[2]

[1]Assam University Diphu Campus, Department of Computer Science, Diphu 782462, Assam, India

[2]Central University of Jharkhand, School of Computer Science and Technology, Brambe, Ranchi 835205, Jharkhand, India

Abstract

Large autoregressive models are increasingly used to produce long-form reasoning and structured outputs in settings where reliability must be assessed, not merely assumed. In practical deployments, failures often arise from subtle inconsistencies that accumulate across many generated tokens: entity bindings drift, quantitative relations become incompatible with earlier commitments, and local fluency masks global constraint violations. This paper introduces a constraint-lattice decoding framework that treats generation as inference over a latent lattice of semantic commitments, enabling both improved constraint adherence and explicit posterior certificates of residual risk. The core contribution is a semiring-valued lattice transducer that couples token likelihood with constraint energies defined over evolving symbolic states, yielding a tractable dynamic program for selecting continuations that optimize a calibrated tradeoff between probability and constraint satisfaction. We formalize constraint extraction as a differentiable map from prefix representations to stochastic constraint factors, and we integrate these factors into decoding through an energy-tilted posterior that supports confidence reporting. A certificate is produced by bounding the posterior mass of all lattice paths whose constraint violation exceeds a threshold, which yields a principled abstention and verification mechanism without requiring external tools. The approach is compatible with standard cache-based decoding and can operate in streaming mode, updating certificates online as tokens arrive. We analyze identifiability of constraint states under partial observability, derive stability guarantees under prefix perturbations, and describe evaluation protocols that separate local perplexity gains from global consistency improvements.

1 Introduction

Autoregressive generation has become the standard approach for producing text in modern language models [1]. At its core, the method relies on predicting each token in a sequence one at a time, conditioning on all previously generated tokens. Each step represents a locally normalized decision: the model assigns probabilities over the next possible tokens and selects one according to its decoding strategy, such as greedy search, beam search, or sampling. While this approach is elegant and effective for short sequences, it faces fundamental challenges when tasked with generating long, structured, or compositionally complex outputs. In such cases, the model's local decisions often fail to ensure global coherence, leading to contradictions, broken references, or factual errors that only become apparent much later in the text.

This issue stems from the fact that most autoregressive decoding procedures are designed to optimize a token-level likelihood objective. The training process maximizes the probability of each token given its context, without explicitly optimizing for long-range consistency or adherence to higher-level task constraints. As a result, even if the model internally represents information about global structure, its generation process does not necessarily reflect that structure in the final output. Each prediction is made in isolation, without a mechanism for checking whether the overall message remains coherent or valid. This disconnect between local token prediction and global semantic or logical consistency is at the heart of many of the reliability issues seen in large language models today.

When outputs are short, these problems are often manageable. For instance, in a simple question-answering task, inconsistencies can be detected with heuristics such as answer type validation, or corrected by reranking multiple candidate responses. Similarly, for tasks like sentence completion or headline generation, the limited scope allows for post-hoc verification—users or automated systems can easily identify nonsensical or contradictory statements.

Symbol	Type	Role in model	Example from text
x	Prompt token sequence	Conditions generation and constraints	Task specification, schema, and facts in the input prompt
$y_{1:T}$	Output token sequence	Long-form response decoded autoregressively	Reasoning chain or multi-step explanation
$s_t \in \mathcal{S}$	Latent constraint state	Compact commitments for bindings, schema, invariants	Entity table, schema bitmask, quantitative summary
c_t	Constraint factor	Stochastic factor emitted by extractor q_ψ	Equality / type / unit constraints over current prefix
$E_t(s_t)$	Constraint energy	Penalty for states that are inconsistent with inferred constraints	Higher for unit mismatch or missing required fields
$V(s_t)$	Violation measure	Aggregated violation score compared to threshold B	Count of broken checks or contradiction score
λ	Tilt strength	Balances likelihood vs. constraint satisfaction	Larger λ enforces stricter adherence

Table 1: Core notation used in the constraint-lattice formulation.

Component	Primary input	Responsibility	Typical signals
Prompt & prefix	Text x, partial $y_{<t}$	Defines task, schema, and factual context	Instructions, tools, database or API contracts
Base AR model p_θ	Tokens $(x, y_{<t})$	Proposes token-level likelihoods $p_\theta(y_t \mid \cdot)$	Logits, hidden states, top-k candidates
Constraint extractor q_ψ	Prefix representations	Infers stochastic constraint factors c_t	Entity graphs, schema masks, numeric checks
State transition p_η	(s_{t-1}, y_t)	Updates compact constraint state s_t	Binding updates, schema slot completion
Constraint lattice	Candidate s_t and y_t	Organizes reachable states over time	Nodes for s_t, edges for token-induced transitions
Semiring DP	Local weights w_t	Runs max-plus and log-sum-exp recursions	Best-path score, partition functions, marginals
Certificate head	Forward statistics	Computes posterior-style risk estimates	Mass on `BAD`, residual bounds, calibrated risk

Table 2: Main components of the constraint-lattice decoding architecture and their responsibilities.

However, as outputs grow longer, the cost of such verification skyrockets. In long-form writing, reasoning chains, or multi-step computational tasks, an early local mistake may propagate quietly, leading to errors that only manifest much later [2]. A single incorrect assumption made early in the sequence can invalidate an entire argument, yet the model has no built-in mechanism for detecting or repairing it once the token has been emitted.

The nature of these failures is diverse. Some appear as explicit contradictions—for example, a generated essay might first claim that a character died in a certain year and later describe them as alive. Others are more subtle, such as broken references where an earlier entity is reintroduced under a slightly altered name, or numerical inconsistencies in quantitative reasoning tasks. In code generation, the equivalent might be syntactic validity without semantic correctness: the program runs but produces the wrong output because earlier assumptions about variable relationships were inconsistent. All of these cases share a common root cause: the autoregressive process optimizes each decision locally, but the constraints that define correctness or coherence exist globally.

To understand why this mismatch is difficult to overcome, it helps to think about the objective function guiding generation. During training, models are taught to predict the next token in a sequence that appears in human-written text. This next-token prediction objective encourages fluency and local grammaticality but does not directly reward maintaining long-term consistency. From the model's perspective, as long as each local decision looks statistically plausible given its context, the sequence as a whole is considered high quality. There is no explicit notion of "truth" or "logical validity" embedded in this objective. Consequently, the model may generate a series of statements that each appear reasonable in isolation but collectively violate global constraints.

Another contributing factor is the absence of a reliable measure of uncertainty regarding global coherence. Traditional decoding strategies expose token-level probabilities but not a holistic estimate of consistency risk [3]. When a model completes a paragraph, it cannot report how likely it is that this paragraph contradicts earlier content. This opacity makes it difficult for systems that rely on language models to gauge the reliability of generated outputs, especially in high-stakes applications. Humans reading a long piece of text can often detect when something "feels off" and revise it accordingly, but the model itself lacks that kind of self-monitoring.

Efforts to address these issues can be grouped into several broad strategies. One approach is to modify the training objective to include signals for global coherence. For example, reinforcement learning from human feedback (RLHF) introduces a reward model that scores outputs based on holistic human judgments, encouraging more globally consistent behavior. However, while RLHF can improve alignment with user preferences and discourage overt contradictions, it still depends on the underlying autoregressive architecture and cannot guarantee consistency in every case. The reward model itself learns correlations rather than enforcing hard logical or factual constraints.

Decoding strategy	Constraint handling	Strengths	Limitations
Greedy / standard beam search	Ignores explicit constraints; uses token likelihood only	Simple, fast, widely deployed	No guarantee of global consistency; entity drift
Sampling with reranking	Applies constraint checks on completed candidates	Can enforce validators; supports diversity	Expensive for long outputs; post-hoc only
Post-generation verification	External tools check factual or logical consistency	Flexible, can plug in symbolic systems	Requires extra passes; no path-level probabilities
RLHF-tilted decoding	Constraints folded into reward model	Encourages holistic preferences	Soft preferences, cannot encode hard invariants
Constraint-lattice decoding	Constraints integrated as energies in the lattice	Jointly optimizes likelihood and constraints; yields certificates	Requires state design, lattice construction, calibration

Table 3: Comparison of decoding paradigms in terms of how they incorporate global constraints.

Semiring view	Addition $\oplus$	Forward variable	Primary use in lattice
Max-plus	$\max(a, b)$	$\alpha_t^{\max}(s)$	Best joint path and decoded continuation $\hat{y}_{1:T}$
Log-sum-exp	$\log(\exp a + \exp b)$	$\alpha_t^{\sum}(s)$	Partition function and state marginals for certification
Tropical risk difference	Max and log-sum-exp	Differences between best-path and marginal scores	Approximate risk ratios relative to optimal path
Probability (sum-product)	$a+b$ in probability space	$p_t(s)$ (normalized)	Conceptual view of posterior over state trajectories

Table 4: Semiring dynamic programming perspectives used for decoding and posterior-style certification.

Another approach involves post-generation verification or correction. In this paradigm, the model generates one or more candidate outputs, which are then evaluated by secondary systems for coherence, factuality, or logical soundness. Reranking based on these evaluations can improve quality, especially for shorter tasks. Yet for long outputs, this approach becomes computationally expensive. Evaluating thousands of tokens for consistency requires comparing every statement against all others, a task that scales poorly with length. Moreover, the verifier must be at least as capable as the generator in understanding context and detecting contradictions—a challenging requirement [4].

A third line of work focuses on structured decoding. Instead of sampling tokens independently at each step, structured decoders incorporate explicit constraints or planning mechanisms that guide generation. For example, a model might first outline the overall structure of a document—its sections, arguments, and key facts—and then generate each part according to that plan. By anchoring local decisions to a global framework, structured decoding can reduce inconsistencies. However, this approach introduces its own complexities: deciding how to represent and enforce the structure, how to allow for flexibility within constraints, and how to maintain efficiency in large-scale models.

There is also growing interest in integrating symbolic reasoning and neural generation. Hybrid systems combine the fluency of language models with the precision of symbolic logic or database querying. For instance, when generating factual statements, a model might call an external verification tool or structured knowledge base to check the validity of claims before committing them to the output. Such modular architectures can mitigate certain types of inconsistency—especially factual errors—but they rely on careful interface design and often reduce generative flexibility.

Beyond technical methods, the problem also invites deeper reflection on what "consistency" really means in the context of language models. Human language itself is often inconsistent, filled with ambiguities, contradictions, and revisions. People can hold and express conflicting ideas, yet communication remains meaningful because humans interpret text pragmatically, not strictly logically. In contrast, language models lack the meta-cognitive awareness to recognize when they are contradicting themselves or violating implicit expectations. They operate purely on statistical associations, and while these associations capture patterns of coherence from data, they do not inherently encode truth or reliability [5].

One might ask whether true global consistency is even attainable under a purely autoregressive paradigm. Since the model cannot revise past tokens, it has no way to retroactively correct earlier mistakes once new information is introduced. This constraint mirrors the one-way flow of time in human speech—we, too, cannot unsay our words—but humans compensate with self-monitoring and repair mechanisms, such as clarification, hedging, or rephrasing. A generative model, by contrast, lacks such mechanisms unless explicitly designed. Some experimental systems introduce limited forms of self-revision, where the model reviews and edits its own output after the initial pass. This iterative generation can improve coherence but departs from the strict autoregressive principle.

In applications where correctness and consistency matter, such as scientific writing, legal reasoning, or edu-

Task family	Constraint type	State feature example	Violation signal $V(s_t)$
Schema completion	Required slots, type checks	Bitmask of filled fields, per-slot type tags	Missing mandatory field, incompatible type
Entity tracking	Coreference, role stability	Map from mentions to entity IDs	Binding churn, contradictory attributes
Numeric reasoning	Arithmetic and units	Table of quantities, units, relations	Sum or unit inconsistency, range violation
Code generation	Structural and semantic checks	Stack for brackets, symbol table for variables	Unbalanced blocks, use-before-def, type error
Logical explanation	Consistency of claims	Set of asserted predicates / relations	Explicit contradiction or violated implication

Table 5: Illustrative constraint types and corresponding state features captured in the lattice.

Objective	Target components	Supervision signal	Typical loss shape
Language modeling	Base AR model p_θ	Next-token text from training corpus	Cross-entropy over $y_t \mid (x, y_{<t})$
Constraint ranking	Energies E_t, extractor q_ψ	Fluent positives vs perturbed inconsistent negatives	Margin ranking between consistent and inconsistent paths
State supervision	Transition model p_η	Optional annotations for entities, schema, variables	Token-level or event-level classification / sequence loss
Prefix risk prediction	Early prefixes and states	Whether full output eventually violates constraints	Binary or calibrated regression loss on risk scores
Certificate calibration	Certificate head / mapping g	Empirical violation indicators on held-out prompts	Temperature scaling or monotone calibration of $\hat{\rho}$

Table 6: Training and calibration objectives used to learn constraint energies, state updates, and certificates.

cational content, the limitations of autoregressive generation become particularly salient. Users expect not just fluent text but reliable information that adheres to logical and factual constraints. When these expectations are unmet, trust in the system erodes. A model that produces a beautifully phrased essay riddled with subtle contradictions may impress superficially but fail in its intended purpose. Addressing this gap requires both algorithmic innovations and clearer standards for evaluating consistency beyond surface fluency.

One possible direction is to rethink how we define the decoding objective. Instead of optimizing token-level likelihood, future systems might integrate latent variable models or energy-based formulations that evaluate entire sequences as wholes. Such global objectives could penalize contradictions or reward adherence to structured constraints during decoding. However, these methods remain computationally demanding and are difficult to scale to the vast vocabularies and sequence lengths of modern models [6]. Another direction involves uncertainty estimation—developing ways for the model to signal when it is uncertain about the consistency of its own output. Confidence scores at the discourse level could help users and downstream systems decide when to trust or verify generated text.

Evaluation also plays a critical role. Traditional metrics like perplexity or BLEU capture fluency and local overlap with reference texts but are insensitive to global coherence. Newer evaluation frameworks attempt to measure factual consistency, logical validity, and narrative continuity, but these remain active research areas. Automatic detection of contradictions or broken references is nontrivial, especially when errors depend on subtle contextual cues. Human evaluation remains the gold standard, but it is expensive and subjective, further motivating the search for reliable automated proxies.

The issue of global consistency is not unique to text generation. Similar challenges arise in other autoregressive domains such as image generation, where locally plausible details may lead to globally implausible compositions, or in music generation, where local harmonic transitions can violate long-term structure. These parallels suggest that the tension between local optimization and global coherence is a general property of sequential generative models. Solving it may require a more fundamental shift in how generative systems represent and reason about structure over time.

A common response is to add post-hoc verification or to rely on repeated sampling and selection. These strategies can improve average quality but tend to be compute-intensive and provide limited interpretability. Moreover, they often fail to separate two distinct questions: whether an output is high probability under the model, and whether it is globally consistent with a set of constraints induced by the prompt and by the model's own earlier commitments [7]. For many applications, the second question is decisive. A high-probability continuation that violates a required schema, contradicts an earlier statement, or breaks an entity binding can be unacceptable even if it is fluent. Conversely, a slightly lower-probability continuation that maintains internal consistency can be preferable.

This paper proposes an alternative: incorporate constraints directly into decoding as a structured latent inference problem. The central idea is to represent generation as traversal through a constraint lattice whose states

Certificate type	Event measured	Example query	Interpretation
Terminal violation	$V(s_T) > B$ at the end of decoding	"How likely is the final state to break constraints?"	Focuses on outcome consistency at sequence end
Path-level violation	$\exists t \leq T : V(s_t) > B$ via `BAD` state	"Did any point along the path cross the threshold?"	Aggregates all trajectories that ever become bad
Residual-mass upper bound	Omitted token/state probability	"How much risk is hidden by pruning?"	Adds conservative bound from pruned lattice mass
Selective risk	Risk vs coverage under abstention	"If we only answer on low-risk prompts, how often are we wrong?"	Supports threshold-based deployment policies
Stratified risk	Risk conditioned on domain or stratum	"Is calibration stable across task families or languages?"	Reveals heterogeneity in certificate behavior

Table 7: Different certificate variants derived from posterior mass on violating trajectories.

Mechanism	Control knob	Effect on lattice	Trade-off
Adaptive top-k tokens	Vary k by step	Limits outgoing edges per state	Larger k improves coverage but increases compute
Nucleus (top-p) pruning	Threshold on cumulative mass	Keeps high-mass tokens under p_θ	Lower p speeds decoding but enlarges residual risk
State cap M	Max states per layer	Bounds lattice width	Small M risks dropping good states; large M costs memory
Hash-based state merging	Summary hash for constraint-relevant features	Collapses equivalent summaries into one node	Reduces duplication; may blur fine-grained differences
`BAD` + residual bound	Absorbing bad state and omitted-mass term	Tracks violating paths and pruning uncertainty	Guarantees conservative upper bounds on risk

Table 8: Pruning and merging controls that keep the constraint lattice tractable while tracking uncertainty.

encode semantic commitments, and whose transitions are induced by candidate token emissions. Constraints are treated as energies on lattice states and transitions, derived from the prompt and the evolving prefix. Decoding then becomes inference in an energy-tilted posterior over lattice paths. This posterior supports two outputs simultaneously: a best-path continuation that balances likelihood and constraint satisfaction, and a certificate that quantifies residual posterior mass assigned to high-violation paths. The certificate functions as a calibrated measure of risk that can trigger abstention, request clarification, or allocate additional compute to verification when warranted.

The contribution is not a new benchmark and not a new prompting strategy. It is a decoding and certification methodology, designed to be layered onto existing autoregressive models. The methodology assumes that constraint signals can be extracted from internal representations and that these signals are imperfect. The framework therefore treats constraints probabilistically rather than as brittle filters, and it explicitly propagates uncertainty through the lattice. The resulting system is intended to reduce long-range inconsistency, to improve predictability under structured requirements, and to provide a meaningful confidence signal that reflects global coherence rather than local token entropy [8].

The remainder of the paper develops the constraint-lattice formulation, the dynamic program that enables tractable decoding, the certificate construction, and a training approach for the constraint extractor and calibration components. The final sections analyze stability and discuss evaluation methodology and implementation considerations.

2 Constraint-Lattice Model

We define the generation process as a joint distribution over an observed token sequence and a latent constraint-state trajectory. Let x denote a prompt token sequence and let $y = (y_1, \ldots, y_T)$ denote the generated response tokens. Let $s_t \in \mathcal{S}$ denote a latent constraint state after emitting $y_{1:t}$. The state space $\mathcal{S}$ is not a full symbolic world model; it is a compact representation of commitments relevant to constraint satisfaction. Examples include a set of entity bindings, a table of declared variables and their types, a multiset of asserted relations, a partially filled schema, or a low-dimensional summary of quantitative invariants. The design requirement is that the state admits incremental updates and that constraint violations can be scored from the state.

The model couples an autoregressive base distribution with a constraint energy. The base model provides conditional probabilities $p_\theta(y_t \mid x, y_{<t})$. The constraint component provides an energy $E_t(s_t; x, y_{\leq t})$ that penalizes states inconsistent with inferred constraints. The energy is induced by a constraint extractor operating on the prompt and prefix. Since the constraints are uncertain and can be incomplete, we model them as stochastic

Evaluation axis	Metric examples	Questions answered	Notes
Constraint satisfaction	Violation rate of $V(s_T)$, schema error rate	Does decoding respect required structural and logical checks?	Directly tied to energies and violation threshold B
Long-range coherence	Entity binding stability, contradiction detection	Are entities, variables, and quantities consistent over long spans?	Uses state trajectories, not just surface tokens
Certificate calibration	Reliability diagrams, Brier score	Do predicted risks match empirical violation frequencies?	Evaluated across bins of calibrated certificate values
Selective risk vs coverage	Risk-coverage curve, AUROC-style summaries	How much can abstention reduce error at given coverage?	Important for deployment with reject options
Compute vs quality	Tokens/sec, DP cost vs accuracy	What is the overhead of the lattice relative to plain decoding?	Varies with candidate set size and state budget

Table 9: Evaluation dimensions for assessing both consistency improvements and certificate quality.

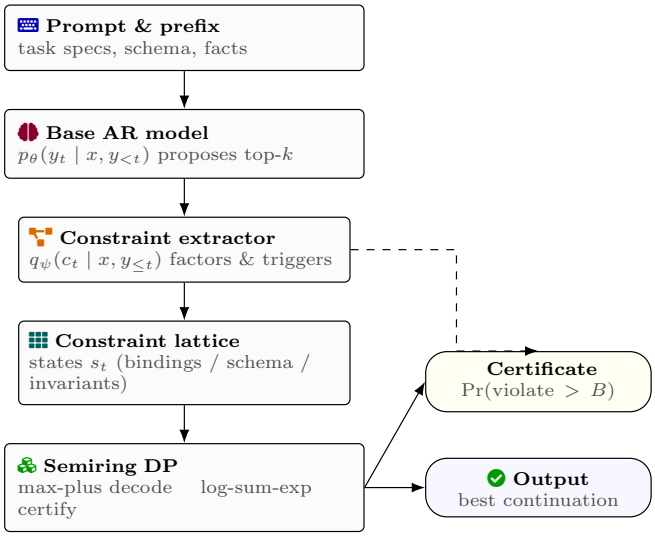

Figure 1: End-to-end constraint-lattice decoding: the base autoregressive model proposes token candidates while a constraint extractor induces stochastic factors; a compact state lattice supports semiring dynamic programming to produce both the chosen continuation and a posterior-style risk certificate.

factors. Let c_t denote a random constraint factor at time t, sampled from a distribution $q_\psi(c_t \mid x, y_{\le t})$. Each factor c_t defines a potential $\phi_{c_t}(s_t)$, and the energy is the negative log potential. The expected energy is

$$E_t(s_t; x, y_{\le t}) = E_{c_t \sim q_\psi(\cdot \mid x, y_{\le t})}\big[-\log \phi_{c_t}(s_t)\big]. \tag{1}$$

This representation permits uncertainty in constraint extraction while allowing state scoring to remain tractable.

State transitions are induced by token emissions. Given state s_{t-1} and token y_t, the state update is modeled as a stochastic kernel $p_\eta(s_t \mid s_{t-1}, y_t, x, y_{<t})$. In the simplest case, this kernel is deterministic, implementing a rule-based update that maintains bindings and relations extracted from the new token. In more complex settings, the update can be learned, for example by mapping a hidden representation of the new token and prior state to a distribution over next states [9]. The framework accommodates both by treating p_η as a transition model defined on a compact state space.

We define an energy-tilted joint distribution over paths:

$$p(y_{1:T}, s_{0:T} \mid x) \propto \Big(\prod_{t=1}^{T} p_\theta(y_t \mid x, y_{<t})\Big)\Big(\prod_{t=1}^{T} p_\eta(s_t \mid s_{t-1}, y_t, x, y_{<t})\Big) \exp\Big(-\sum_{t=1}^{T} \lambda E_t(s_t; x, y_{\le t})\Big), \tag{2}$$

where $\lambda \ge 0$ controls constraint strength. The distribution is locally normalized in y through p_θ but globally shaped by state transitions and energies. Decoding seeks a response y that is both probable under the base model and consistent under the constraint energies. A straightforward approach would search over both y and s, but that is intractable unless $\mathcal{S}$ is small and transitions are simple. The constraint-lattice formulation makes this tractable by restricting the state representation and structuring the search.

The lattice is constructed as follows. At each time step t, we maintain a set of candidate states $\mathcal{S}_t \subseteq \mathcal{S}$ reachable under a bounded set of token hypotheses. Each state is a node in the lattice layer at time t. Edges

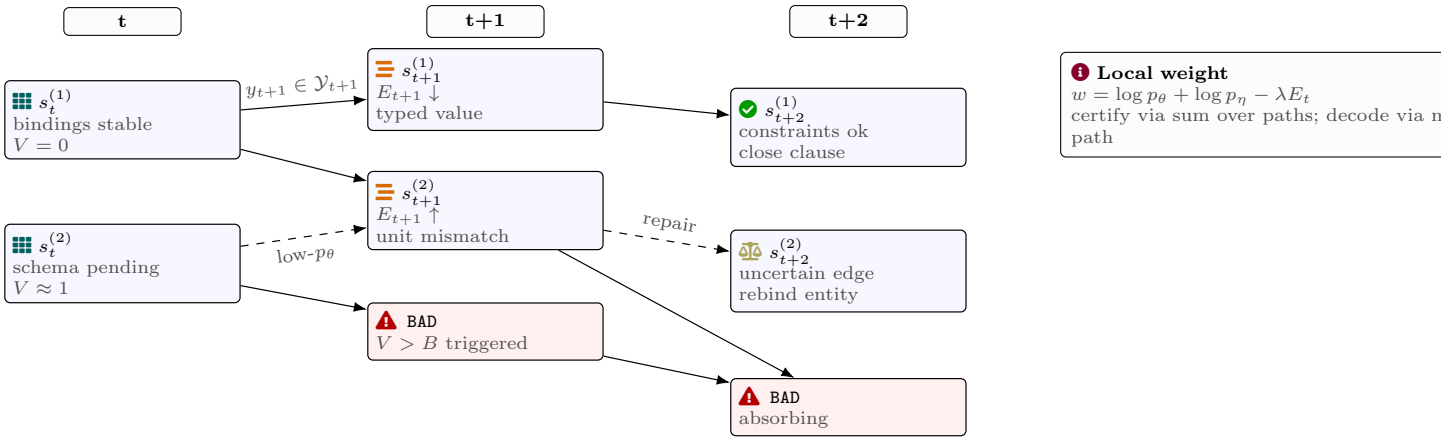

Figure 2: A pruned constraint lattice over compact states: token proposals induce state transitions, constraint energies tilt path weights, and an absorbing `BAD` state aggregates trajectories that cross a violation threshold.

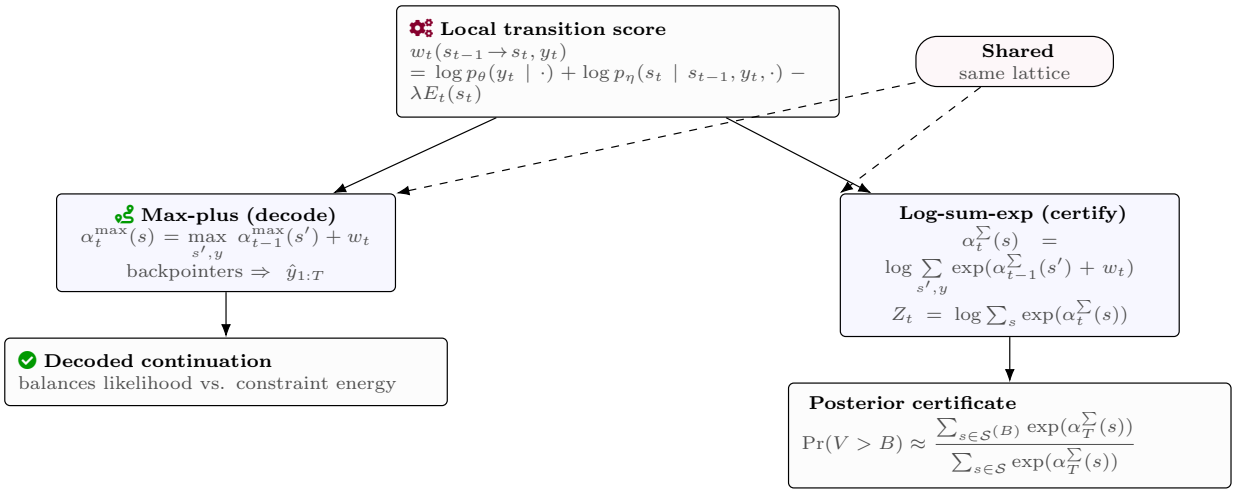

Figure 3: One lattice, two semirings: max-plus yields the best joint path for decoding, while log-sum-exp yields partition functions and marginal mass for certification under the same local transition scores.

connect s_{t-1} to s_t when there exists a token y_t such that $p_\eta(s_t \mid s_{t-1}, y_t, \cdot)$ is non-negligible. In many tasks, state updates depend on semantic events rather than individual tokens. Therefore, in practice, we group tokens into semantic emission events, such as completing a field value, closing a bracket, or emitting a numerical quantity. The framework supports this by defining emissions at variable lengths, but for notation we treat emissions as token-level and interpret y_t as the atomic emission unit of the implementation.

The scoring of a lattice path is the sum of token log-probabilities, transition log-probabilities, and negative energies. For a fixed token sequence, the marginal score is obtained by summing over state paths, which yields a partition function. For decoding, we typically seek the maximum a posteriori path in the joint space, which yields both a token sequence and a state trajectory. Alternatively, we can seek the maximum marginal token sequence by marginalizing states, but that is harder [10]. The next section derives a dynamic program that efficiently computes best paths and approximate marginals under a semiring representation.

The constraint-lattice formulation is designed to support certification. Define a violation measure $V(s_t)$ that maps a state to a nonnegative scalar summarizing constraint violation at time t. This measure can aggregate multiple constraints, such as schema completeness, contradiction counts, type errors, or quantitative inconsistency. The certificate aims to estimate the posterior probability that the violation exceeds a threshold at any point or at the end. Because V is defined on states, the certificate is computed from posterior mass over lattice paths. This shifts certification from an unstructured text space to a structured state space, where aggregation and bounding are tractable.

While the formulation resembles a hidden Markov model with emissions, it differs in two essential ways. The emission probabilities come from a large neural language model, and the constraint energies are induced by a learned extractor that depends on the entire prefix. The resulting model is not stationary, but it can still be decoded with dynamic programming when state sets are bounded and when energies decompose over time. The next section makes these computational assumptions explicit and develops the semiring dynamic program used for

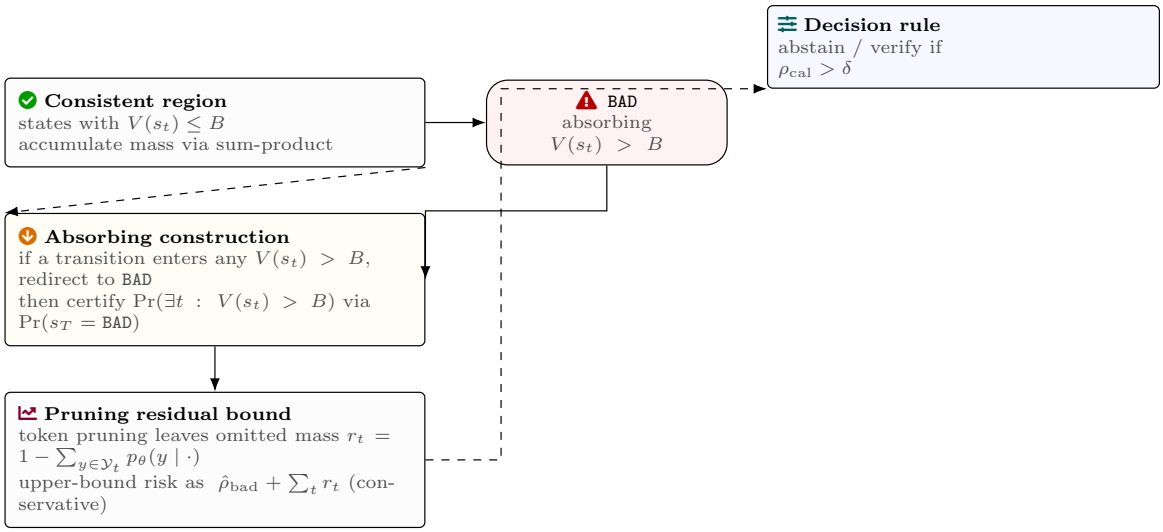

Figure 4: Certification as posterior mass on violating trajectories: an absorbing `BAD` state aggregates threshold crossings, while a residual term accounts for probability mass omitted by token/state pruning.

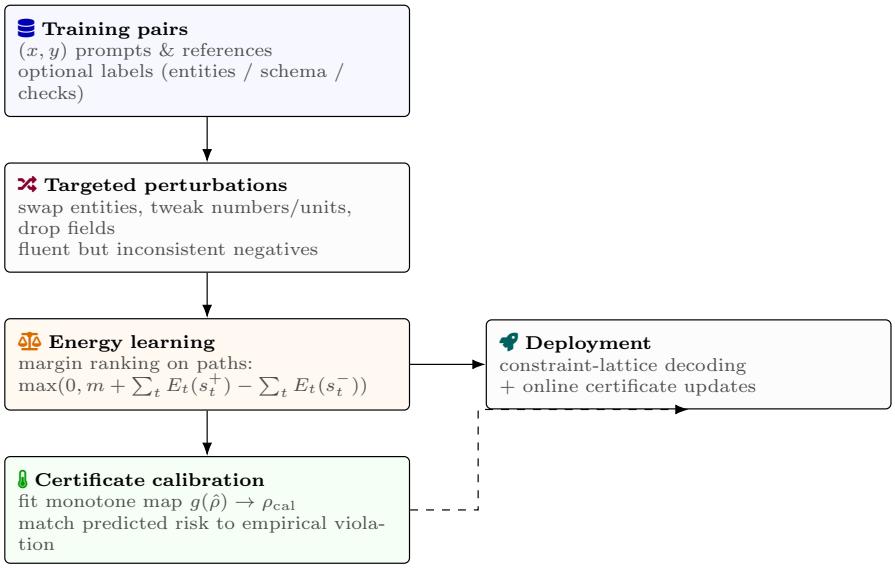

Figure 5: Learning and calibration pipeline: construct hard-but-fluent negatives to train constraint energies, then calibrate certificate values against held-out violation rates before deploying the lattice decoder with online risk reporting.

decoding and certification.

3 Semiring Dynamic Programming for Decoding and Certification

The computational goal is to perform two related operations online: select a continuation that optimizes a combined likelihood-energy objective, and compute a certificate quantifying posterior mass of high-violation paths. Both operations can be expressed using generalized forward recursions in appropriate semirings. This section derives these recursions and shows how they can be implemented in a streaming decoder with bounded memory [11].

For each transition from s_{t-1} to s_t under token y_t, define a local weight

$$w_t(s_{t-1}, s_t, y_t) = \log p_\theta(y_t \mid x, y_{<t}) + \log p_\eta(s_t \mid s_{t-1}, y_t, x, y_{<t}) - \lambda E_t(s_t; x, y_{\leq t}). \tag{3}$$

For a path, the total score is the sum of local weights. In max-product form, the best path score at state s_t is

$$\alpha_t^{\max}(s_t) = \max_{s_{t-1}, y_t} \Big(\alpha_{t-1}^{\max}(s_{t-1}) + w_t(s_{t-1}, s_t, y_t) \Big), \tag{4}$$

with initialization $\alpha_0^{\max}(s_0)$ defined by a prior over initial states. This recursion is a Viterbi-like update. The challenge is that y_t is not fixed; it is part of the decision. A direct maximization over all tokens at each step would

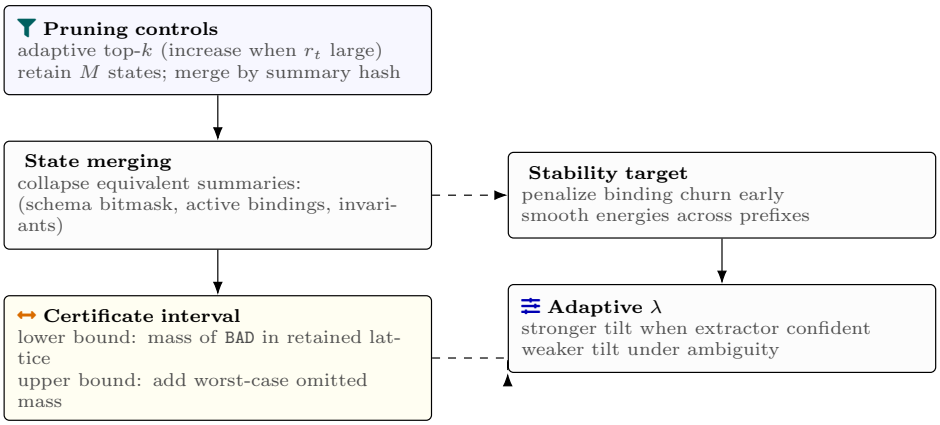

Figure 6: Practical controls for long-context decoding: adaptive pruning and state merging keep the lattice tractable, while certificates can be reported with conservative bounds that reflect omitted probability mass and varying constraint strength.

be too expensive. We therefore restrict the token set considered at each step to a candidate set $\mathcal{Y}_t$ produced by a standard decoder proposal, such as top-k or nucleus sampling on p_θ. For each proposed token $y_t \in \mathcal{Y}_t$, we compute successor states under p_η. This yields a bounded number of transitions per step, enabling Viterbi updates over a bounded lattice.

The above recursion yields the best joint path under the restricted lattice. To recover the token sequence, we store backpointers. This produces a decoded output that balances token probability and constraint energies. The approach differs from reranking because constraint energies influence decisions at every step, not only at the end, which prevents early drift that would later be expensive to repair.

Certification requires approximating posterior mass, which corresponds to sum-product inference in the same lattice. Define forward weights in log space:

$$\alpha_t^{\sum}(s_t) = \log \sum_{s_{t-1}, y_t} \exp\Big(\alpha_{t-1}^{\sum}(s_{t-1}) + w_t(s_{t-1}, s_t, y_t)\Big). \tag{5}$$

The partition function at time t is $Z_t = \log \sum_{s_t} \exp(\alpha_t^{\sum}(s_t))$. From these quantities, one can compute marginal probabilities of states and transitions, enabling estimation of the probability of constraint violations.

Let B be a violation threshold [12]. Define the set of terminal states whose violation exceeds the threshold, $\mathcal{S}_T^{(B)} = \{s_T : V(s_T) > B\}$. The posterior probability that terminal violation exceeds B is

$$\Pr(V(s_T) > B \mid x) = \frac{\sum_{s_T \in \mathcal{S}_T^{(B)}} \exp(\alpha_T^{\sum}(s_T))}{\sum_{s_T \in \mathcal{S}_T} \exp(\alpha_T^{\sum}(s_T))}. \tag{6}$$

This ratio can be computed stably using log-sum-exp. A more conservative certificate considers violation at any time step, not only at the end. Define an absorbing bad state `BAD` that represents that the path has exceeded the violation threshold at some prior time. We augment the lattice so that any transition into a state with $V(s_t) > B$ forces the state to `BAD` thereafter. Then the posterior mass of bad paths is simply the marginal probability of `BAD` at time T. This yields a certificate of the form $\Pr(\exists t \le T : V(s_t) > B \mid x)$, which is often more relevant because intermediate inconsistencies can be unacceptable even if later repaired.

The above is exact in the restricted lattice. In a real decoder, the lattice is an approximation due to token pruning and state pruning. Certification must account for omitted mass. We handle this by maintaining an explicit residual mass bound at each step. Let $\mathcal{Y}_t$ be the considered tokens and let r_t be an upper bound on the omitted token probability mass under the base model, $r_t = 1 - \sum_{y \in \mathcal{Y}_t} p_\theta(y \mid x, y_{<t})$. Since the constraint energy can only reduce weights relative to base probabilities for $\lambda \ge 0$, omitted mass in the tilted model is at most omitted mass in the base model scaled by a factor accounting for transition probabilities and energies. A simple conservative bound treats omitted transitions as potentially all bad. Then the certificate upper bound becomes the computed bad mass plus the cumulative omitted mass bound. This is loose but monotone and can be tightened by estimating typical energy ranges.

The semiring view unifies decoding and certification. In the max-plus semiring, addition is max and multiplication is plus, yielding best-path decoding [13]. In the log-sum-exp semiring, addition is log-sum-exp and multiplication is plus, yielding partition functions and marginals. Implementationally, the same lattice construction and local weights are reused; only the aggregation operator changes. This makes certification feasible at near the cost of beam decoding, especially when state sets are small.

To make state sets small, the state representation must be designed carefully. A useful design is to factor state into components that are mostly independent and to prune states by merging equivalent constraint-relevant summaries. For example, if the constraint energy depends only on a schema completeness bitmask and a set of active entity bindings, states with identical bitmask and bindings can be merged even if they differ in irrelevant internal details. This merging is crucial for keeping the lattice manageable. The framework does not prescribe a single state representation; it provides a decoding and certification mechanism conditional on a chosen representation.

A further improvement is to use a tropicalized certificate that estimates the log probability of bad paths relative to best paths, which can be computed using difference of max and log-sum-exp quantities. This yields a risk score that remains informative even when full posterior mass is difficult to compute due to pruning. The certificate can then be interpreted as a calibrated measure of uncertainty, provided calibration is performed on held-out data.

This section has focused on inference. The next section addresses how to learn the constraint extractor and the transition model so that constraint energies reflect meaningful violations and state updates capture relevant commitments. The learning problem is framed as fitting energies and transitions to supervision signals that can be heterogeneous, including correctness labels, consistency annotations, and synthetic perturbations.

4 Learning Constraint Energies, State Updates, and Calibration

The constraint-lattice decoder assumes the existence of a constraint extractor $q_\psi(c_t \mid x, y_{\le t})$ and a state transition model $p_\eta(s_t \mid s_{t-1}, y_t, \cdot)$. In many tasks, some aspects of these components can be rule-based [14]. For example, schema adherence can be tracked deterministically, and bracket matching can be enforced by deterministic state updates. However, long-form reasoning also involves semantic constraints that are not easily captured by rules, such as whether an entity mention refers to the same referent as earlier, whether a numerical statement contradicts earlier quantities, or whether a causal explanation is consistent with previous claims. For these, learned models are needed. This section presents a learning formulation that supports both rule-based and learned components and yields calibrated certificates.

We assume a training dataset of prompts and target outputs, $\{(x^{(i)}, y^{(i)})\}$. Depending on the application, supervision may include additional annotations such as entity spans, schema labels, or contradiction markers. The learning objective combines three terms: a task likelihood term for the base model (or a fixed base model if not trained), a constraint discrimination term that encourages energies to separate consistent from inconsistent states, and a calibration term that aligns certificate scores with empirical error.

The constraint discrimination term is constructed by generating negative samples that violate constraints. Given a training pair (x, y), we produce perturbed outputs $\tilde{y}$ by applying operations designed to break specific constraints while preserving surface plausibility. Examples include swapping entity names, altering a numerical quantity, changing a unit, permuting steps in a procedural explanation, or deleting a required field in a schema. The goal is to generate hard negatives that are fluent but inconsistent. We then run the state transition model to obtain state sequences for y and $\tilde{y}$, and we train the constraint extractor so that energies are lower on the consistent path than on inconsistent paths by a margin. A representative loss is

$$\mathcal{L}_{\text{rank}} = E\Big[\max\big(0, m + \sum_t E_t(s_t^+) - \sum_t E_t(s_t^-)\big)\Big], \tag{7}$$

where s_t^+ is the state trajectory for the true output and s_t^- is for the perturbed output, and m is a margin. This loss encourages energies to reflect violations that matter for distinguishing correct from perturbed outputs. Because constraint extraction is uncertain, the energy uses expectation over c_t, which is differentiable when q_ψ is reparameterized or implemented as a mixture with straight-through estimators.

State update learning depends on whether p_η is deterministic or learned. If learned, we can treat the latent state trajectory as an inference variable and learn p_η by maximizing likelihood of annotated state components when available [15]. For example, if entity bindings are annotated at some positions, one can supervise those binding variables. When annotations are not available, one can learn p_η jointly with energies by treating state updates as latent but constrained by energy minimization. In practice, a stable approach is to define a partially deterministic state representation where easily extractable elements are tracked deterministically, and to learn only the ambiguous components, such as coreference clusters or implicit variable alignment. This reduces identifiability issues and makes decoding more reliable.

Calibration is essential because certificates are only useful if their values correspond to empirical risk. The certificate computed from the lattice yields an estimated probability of violation exceeding a threshold. We calibrate this estimate using held-out data. Let $\hat{\rho}(x)$ be the computed bad-path probability for a chosen threshold and decoding configuration. Let $e(x)$ be an empirical error indicator, such as whether the output fails a consistency check or is judged incorrect by an oracle. We fit a monotone calibration map g such that $g(\hat{\rho}(x))$ matches $\Pr(e = 1 \mid \hat{\rho})$.

A simple method is temperature scaling on the logit of $\hat{\rho}$:

$$\rho_{\text{cal}} = \sigma\Big(\frac{1}{T}\log\frac{\hat{\rho}}{1-\hat{\rho}}\Big), \tag{8}$$

with T learned to minimize log loss. More flexible monotone calibrators can be used, but temperature scaling has the advantage of preserving ordering. Calibration can be stratified by domain and language when needed, but the framework is not inherently tied to any particular stratification. The goal is that the certificate remains meaningful across the range of prompts the system sees.

The constraint energies can also be trained to be prefix-consistent. A common failure mode is that an early prefix is scored as consistent, but later becomes inconsistent due to a latent binding error that was not detected early [16]. To reduce this, we train the extractor to predict future violation risk from early prefixes. Given a prefix at time t, we define the eventual violation label as whether the completed output violates constraints. We then train a prefix risk predictor derived from the lattice forward pass, encouraging early detection. This improves streaming certification because the model can raise uncertainty before committing to a long continuation.

In addition to rank-based training, we can use a likelihood-based energy model. If we have binary labels for whether a completed output is consistent, we can train energies so that the partition function under the energy-tilted model assigns higher probability mass to consistent outputs. This resembles noise-contrastive estimation where negatives are sampled from the base model. While powerful, it can be unstable because the space of outputs is large. The lattice approximation makes it more tractable by restricting attention to a bounded candidate set per step, but the method remains compute-intensive. Therefore, the recommended training strategy is to use targeted perturbation negatives that focus energy learning on meaningful constraint violations.

A practical implementation detail is how to construct the constraint factors c_t. Constraint factors are derived from internal representations of the base model and from the prefix text. A common representation is a constraint graph whose nodes are entities, variables, or schema slots, and whose edges represent relations such as equality, inequality, coreference, containment, temporal precedence, or unit compatibility. The extractor produces a distribution over such graphs [17]. Energies are then computed by evaluating how well the current state satisfies the graph. Because graphs are discrete, we either sample graphs or use expected adjacency matrices with relaxed edges. The latter makes energies differentiable but can blur discrete structure. In practice, a hybrid approach works: maintain a small set of sampled graphs and average energies across them, which preserves multimodality. To support multilingual and cross-script evaluation without making it central to the method, one can stratify calibration and certificate validation by writing system or by language variety when available. In particular, when measuring certificate calibration across heterogeneous strata, it is useful to ensure that each stratum has sufficient examples and that scripts with different tokenization properties are represented in the calibration set; participatory benchmark documentation that enumerates language varieties and writing systems can serve as a reference taxonomy for such stratification [18].

This section has described how to learn the energy and transition components and how to calibrate certificates. The next section analyzes theoretical properties relevant to stability and interpretability, focusing on how the lattice approximation interacts with energy tilting and how certificates behave under pruning and prefix perturbations.

5 Stability Analysis and Theoretical Properties

The constraint-lattice framework introduces approximation at two levels: constraints are extracted stochastically and may be imperfect, and the lattice used for decoding is pruned relative to the full output space. This section analyzes how these approximations affect decoded outputs and certificates, and establishes conditions under which the method yields stable behavior under small perturbations.

We begin with stability under prefix perturbations. Consider two prompts or prefixes that are close in representation space, or two decoding runs that differ by a small perturbation in early tokens. Because the base model is highly nonlinear, small changes can cascade. The lattice framework aims to reduce cascades by enforcing constraint-consistent state evolution. A sufficient condition for stability is that the state update and energy functions are Lipschitz with respect to prefix representation and that constraint energies strongly penalize divergent bindings [19]. Under such conditions, paths that diverge early incur higher energy unless divergence is supported by the constraints, which reduces the probability of inconsistent drift.

More formally, let h_t denote the base model hidden representation of the prefix at time t, and let the constraint extractor produce factor parameters as a function of h_t. Suppose that for any two prefixes with representations h_t and h'_t, the expected energy difference for a fixed state satisfies

$$|E_t(s; h_t) - E_t(s; h'_t)| \le L_E \|h_t - h'_t\|_2, \tag{9}$$

and the state transition kernel satisfies a contraction property in an appropriate metric on distributions over states. Then the induced posterior over states is stable up to a factor that depends on the cumulative perturbation in h_t.

While these conditions are strong and not guaranteed, they provide a lens for designing the constraint extractor: it should produce energies that vary smoothly with the prefix representation except when new evidence strongly indicates a different constraint regime. This can be encouraged by regularizing the extractor to be stable under dropout perturbations of h_t and by using entropy penalties to avoid overconfident discrete constraint predictions when evidence is weak.

Next, consider the effect of lattice pruning on certificates. The sum-product recursion yields exact posterior mass only over the retained lattice. Omitted transitions can hide probability mass that could include high-violation paths. Therefore, certificates computed on the pruned lattice should be interpreted as lower bounds on bad mass, unless corrected. The conservative correction treats all omitted mass as bad, yielding an upper bound. The resulting interval between lower and upper bounds quantifies certificate uncertainty due to pruning [20]. A desirable decoder keeps this interval small by selecting candidate sets that capture most probability mass. This can be achieved by adaptive k selection: increase the number of token candidates when the omitted mass bound is large, or when constraint energies indicate high sensitivity. This yields a compute allocation mechanism where harder steps receive more expansion.

A related property concerns monotonicity in λ. Increasing λ strengthens constraint enforcement. In the lattice model, increasing λ decreases weights of high-energy states, which reduces bad mass and can increase confidence, but it can also force the decoder to choose low-likelihood tokens that may reduce task correctness if constraints are mis-specified. This tradeoff suggests treating λ as a tunable parameter and calibrating it on held-out data with respect to an objective that balances correctness and constraint satisfaction. One can also make λ state-dependent, increasing it when constraint extractor confidence is high and decreasing it when confidence is low. This yields an adaptive energy tilt that preserves robustness.

We also analyze identifiability of constraint states. If the state representation is too coarse, multiple incompatible outputs can map to the same state, and energies cannot distinguish them. If the state representation is too fine, the lattice becomes intractable. A useful design principle is to choose state variables that correspond to constraints that can be extracted reliably and that have high impact on global consistency. Entity bindings and schema completion typically satisfy this [21]. Highly nuanced pragmatic constraints may not. The method's success depends on the existence of such intermediate structure. When it exists, dynamic programming can exploit it; when it does not, the lattice degenerates to a beam search with weak energies.

Finally, we consider certificate interpretability. Certificates are derived from posterior mass over bad paths. Unlike heuristic confidence scores, they have a probabilistic meaning conditional on the model and lattice approximation. This meaning is still model-relative, but it supports consistent decision rules such as abstention thresholds. A certificate can be used to guarantee that, under the model, the probability of exceeding a violation threshold is below a chosen level. Calibration connects this model-relative meaning to empirical error. When calibration is successful, the certificate approximates a frequentist guarantee: outputs with certified risk below δ have empirical violation rates near δ on held-out data.

These theoretical properties motivate the evaluation methodology. The next section describes how to measure improvements due to constraint-lattice decoding and how to validate certificates without conflating them with superficial gains in token likelihood.

6 Evaluation Methodology and Implementation Considerations

Evaluating constraint-lattice decoding requires metrics that capture global consistency and certificate quality. Standard perplexity improvements are neither necessary nor sufficient [22]. The method can improve consistency without improving perplexity because it trades probability mass for constraint satisfaction. Therefore, evaluation focuses on three axes: constraint violation rates, long-range coherence metrics, and certificate calibration.

Constraint violation rates can be measured using task-specific validators. For schema tasks, violations include missing fields, invalid types, and bracket mismatches. For reasoning tasks, violations include contradictions with prompt facts, inconsistent entity references, and quantitative incompatibilities. The state representation often provides a natural violation measure $V(s_T)$ that can be computed deterministically from the final state, and the same measure can be computed externally for validation. This yields a direct metric: the frequency with which outputs exceed a violation threshold. Because the decoder explicitly optimizes energies based on V, this metric is directly relevant.

Long-range coherence metrics require more nuance. Coherence includes consistency of entity mentions over long spans, preservation of variable definitions, and non-contradiction among statements. While automated metrics are imperfect, the state trajectory provides internal signals such as binding churn and relation conflicts. One can measure the number of times an entity binding is revised, the number of constraint edges that flip truth values, and the time-to-detection of contradictions. These metrics reveal whether the decoder reduces drift early rather than attempting late repair.

Certificate calibration is evaluated by comparing predicted bad-path probabilities to empirical violation rates. This is done by binning outputs by calibrated certificate value and computing empirical rates in each bin [23]. Calibration should be evaluated under multiple decoding configurations, including different candidate set sizes and different λ schedules, because pruning affects certificates. A well-calibrated certificate should remain stable when candidate sets are moderately varied, indicating that the certificate reflects meaningful risk rather than artifacts of pruning.

It is also important to evaluate coverage versus risk. A certificate can enable abstention: the system can refuse to produce an output when risk is too high, or it can request clarification. In non-interactive evaluations, abstention can be scored separately. The relevant curve is the selective risk curve: violation rate on non-abstained outputs as a function of coverage. Constraint-lattice decoding is expected to yield a better curve because it reduces both actual violations and overconfidence, assuming calibration is effective.

Implementation considerations determine feasibility. The lattice method requires maintaining state sets and computing energies. For many structured tasks, state updates can be implemented with lightweight parsers and incremental extractors. Energies can be computed as sums of small constraint factors. The heaviest component is constraint extraction from the neural model representation, but this can be amortized by sharing computations with the base model and by evaluating constraints only when certain triggers occur, such as completion of a clause or detection of a number. The method can also operate in a two-level mode: use a cheap approximate energy during most steps and a more expensive energy at boundaries, while keeping the lattice consistent.

A key engineering parameter is how many states to retain [24]. Retaining too many states increases cost; retaining too few risks losing the correct path or misestimating certificates. A practical strategy is to retain a fixed number of states per step via pruning by forward weight, and to merge states with identical constraint-relevant summaries. Merging is especially effective when the state includes a schema bitmask or a bounded set of bindings. Merging can be implemented by hashing the summary and aggregating forward weights by log-sum-exp for sum-product and by max for max-product.

The method integrates naturally with beam search. Each beam hypothesis corresponds to a partial token sequence and a set of constraint states with associated weights. Rather than maintaining a single state per beam, the lattice approach maintains multiple states per beam and merges across beams when states coincide. This yields a structured beam that is more robust to early ambiguity in bindings. Certification is computed from the sum-product weights across all beams and states.

Finally, the method should be evaluated across domains rather than tuned to a single task. The aim is to demonstrate that constraint-lattice decoding improves global consistency whenever constraints can be represented compactly, and that certificates provide useful uncertainty estimates. The evaluation should therefore include tasks with different constraint structures, such as schema completion, multi-step explanation, and numerical reasoning. For multilingual usage, the certificate should be validated under stratified conditions, but the method itself does not depend on multilinguality. This aligns with the goal of providing a general decoding and certification mechanism that is not centered on any particular dataset [25].

7 Conclusion

This paper introduced constraint-lattice decoding, a framework designed to bridge the gap between locally normalized autoregressive generation and globally constrained reasoning. Traditional autoregressive decoders generate sequences token by token, conditioning each prediction on the tokens that precede it. While this process yields fluent and coherent language at a local level, it fails to ensure that the final output satisfies the higher-level constraints that define task correctness. Constraint-lattice decoding addresses this issue by explicitly integrating constraint energies into the decoding process through a latent state lattice. This allows the model not only to generate text but also to monitor, quantify, and calibrate its own risk of inconsistency.

At the heart of the method is the notion of representing semantic commitments as compact latent states. Each state in the lattice encodes partial information about the constraints relevant to the task—facts that must remain true, quantities that must balance, entities that must stay consistent, or relationships that must hold across the sequence. As tokens are generated, these states evolve incrementally, reflecting the model's changing understanding of what has been said and what constraints have been activated or violated. The constraint energies serve as potentials defined over these states, quantifying how compatible a given token transition is with the current constraint configuration. By coupling token likelihoods with these constraint potentials, the decoding process moves beyond pure fluency optimization and toward structured consistency optimization.

This integration makes it possible to perform dynamic programming recursions over the state lattice. Instead of evaluating one linear sequence of token predictions, the decoder explores a lattice of possible paths, each corresponding to a combination of linguistic choices and constraint configurations. The dynamic programming formulation allows the system to efficiently compute both best-path estimates and aggregate probabilities over all paths, depending on the semiring chosen. This dual capability forms the basis for two complementary outcomes:

generation of high-consistency outputs and certification of residual inconsistency risk [26]. The former provides the model's best attempt at a coherent output, while the latter quantifies the likelihood that unseen inconsistencies remain.

A key feature of constraint-lattice decoding is the use of a semiring formulation that unifies best-path decoding and sum-product certification. In standard autoregressive decoding, beam search attempts to approximate the most likely sequence, but it provides no reliable measure of how much probability mass lies outside the explored beams. The semiring framework, by contrast, tracks both probabilities and constraint energies jointly, supporting not only maximization but also marginalization. This means that, during decoding, the system can maintain a running account of how much of the total probability mass corresponds to consistent paths versus inconsistent ones. When the process concludes, the remaining unaccounted-for mass serves as a residual bound on the uncertainty—the so-called residual mass bound—which acts as a certificate of how confident the system should be in the generated output's consistency.

This certificate mechanism is more than a theoretical curiosity. It provides a practical way to audit model behavior after generation. Traditional language models can produce confident-sounding but factually or logically incorrect statements without any internal indication of risk. Constraint-lattice decoding changes this by making inconsistency risk a first-class quantity. The posterior distribution over constraint states at the end of decoding reflects how much uncertainty remains about the satisfaction of the constraints. A low residual mass indicates that most plausible continuations agree on the constraint outcomes, suggesting high confidence. A large residual mass, on the other hand, signals that significant probability remains on alternative paths that could violate constraints. This turns what was previously an unobservable property—the hidden fragility of model consistency—into an explicit, quantifiable measure.

The ability to produce such certificates depends crucially on the structure of the constraint representations [27]. For many tasks, constraints can be modeled compactly. For example, in factual generation, a state might represent the truth status of a set of known entities. In arithmetic reasoning, it might represent the current partial sum or count. In logical inference, it might encode whether certain predicates have been satisfied. The more compactly these properties can be encoded and updated, the more efficiently the dynamic programming recursions can operate. This efficiency makes constraint-lattice decoding viable for streaming use, where text is generated and certified incrementally rather than after full completion. The framework supports online decoding, allowing it to operate at near-interactive speeds while still maintaining global consistency tracking.

Learning effective constraint energies is another central challenge addressed by the framework. Since explicit supervision for constraint satisfaction is rarely available, the paper proposes a learning formulation based on targeted perturbation negatives. In this setup, the system is trained not only on fluent reference examples but also on deliberately perturbed variants that are locally fluent yet globally inconsistent. For instance, a training example might swap entity references or alter numerical quantities in ways that preserve grammaticality but violate semantic coherence. The constraint energies are trained to assign higher costs to these inconsistent perturbations, thereby learning to discriminate between superficially plausible and truly consistent outputs. Over time, this targeted contrastive learning sharpens the model's ability to detect subtle violations that standard language modeling objectives would overlook.

A further component of the approach is calibration [28]. After decoding, the system maps posterior bad-path probabilities—the fraction of probability mass on inconsistent paths—to empirical error rates observed during validation. This calibration step allows the model's internal risk estimates to correspond to real-world reliability. For example, if the model reports a 5% inconsistency probability, calibration ensures that roughly 5% of its outputs actually contain inconsistencies in practice. Such calibration maps transform the certificates from theoretical constructs into actionable tools. Users and downstream systems can use them to decide when to trust an output, when to seek human review, or when to allocate more computational resources for re-decoding or verification.

Importantly, constraint-lattice decoding is conceived as a decoding and auditing layer rather than a replacement for base model training. It does not alter the parameters of the underlying language model; instead, it augments the decoding process with a structured reasoning layer that interprets and constrains the model's predictions. This modular design has several advantages. It allows existing models to benefit from constraint-aware decoding without retraining. It also keeps the scope of the method clear: the base model remains responsible for fluency and general linguistic competence, while the constraint lattice ensures global coherence and provides interpretable confidence estimates. The division of labor mirrors how human writers rely on both intuitive composition and explicit reasoning checks to maintain consistency.

The effectiveness of the framework depends on two practical ingredients: the availability of compact, incrementally updateable state representations, and the presence of constraint extractors that provide informative, if uncertain, signals. A constraint extractor is a module that maps the evolving text into constraint-relevant states—for example, identifying entities and their attributes, tracking quantities, or recognizing logical dependencies. These extractors need not be perfect; they can provide probabilistic or noisy signals, as the lattice formulation naturally accommodates uncertainty through its energy potentials [29]. However, they must be expressive enough

to capture the task's essential consistency requirements. When such extractors exist, constraint-lattice decoding can transform an autoregressive model into a system that reasons about its own coherence in real time.

When implemented effectively, this approach provides a principled alternative to common ad-hoc methods like post-hoc reranking or heuristic confidence estimation. Reranking methods typically generate multiple candidate outputs and select the one that appears most coherent according to external scoring models. While useful, this strategy is computationally expensive and lacks probabilistic grounding. Heuristic confidence measures, such as token-level entropy or output diversity, provide limited insight into true inconsistency risk. Constraint-lattice decoding, by contrast, embeds coherence evaluation directly within the decoding process, ensuring that each generation step respects both local likelihood and global constraints. The result is a unified system capable of producing outputs that are not only fluent but also verifiably consistent.

The interpretability of the resulting risk certificates adds an auditing dimension that pure language models lack. In safety-critical or high-reliability contexts—such as scientific reporting, legal drafting, or autonomous system planning—users need to know not just what the model predicts but how confident it is in its consistency. A certificate derived from the residual mass bound provides a clear, quantitative answer: it represents the probability that the model's output violates its own constraints. This allows organizations to set risk thresholds, automate fallback procedures, or integrate human oversight dynamically. For example, a certificate indicating low inconsistency risk might allow automatic publication of a report, whereas a high-risk certificate could trigger additional verification or request more computation for tighter bounds.

Looking ahead, the paper outlines several promising directions for future work [30]. One avenue involves expanding the class of constraint representations that can be handled efficiently. Current implementations focus on relatively simple constraint structures, but many real-world tasks involve richer semantics, such as nested logical conditions or probabilistic dependencies between constraints. Extending the lattice framework to accommodate these more complex relationships could significantly broaden its applicability. Another research direction concerns improving uncertainty modeling in constraint extraction. Better probabilistic models of extraction uncertainty would allow more accurate propagation of constraint beliefs through the lattice, resulting in tighter and more reliable certificates.

A third frontier is adaptive compute allocation based on certificate tightness. In the current formulation, decoding proceeds at a fixed computational budget regardless of how confident or uncertain the system becomes. Yet the certificate itself provides a natural signal for dynamic allocation: if the residual mass bound remains high, additional computation—through expanded beam width, finer state granularity, or secondary passes—can be justified to reduce uncertainty. Conversely, when the certificate tightens rapidly, decoding can stop early, saving resources. This adaptive control could make constraint-lattice decoding not only more consistent but also more efficient and predictable in large-scale deployments.

The implications of this framework extend beyond text generation. The underlying idea—coupling local probabilistic prediction with global constraint tracking—can apply to any autoregressive process where consistency matters. For example, in sequential decision-making or planning tasks, the same principles could ensure that action sequences respect physical or logical constraints while still leveraging learned probabilistic models. In multimodal settings, such as image captioning or data-to-text generation, the lattice could enforce alignment between textual descriptions and visual or structured inputs. Thus, constraint-lattice decoding represents a general paradigm for embedding structured reasoning into generative systems without sacrificing their flexibility or fluency [31].

References

[1] A. Pulka, "Selection of search strategies for solving 3-sat problems," *International Journal of Applied Mathematics and Computer Science*, vol. 24, no. 2, pp. 283–297, Jun. 26, 2014. DOI: 10.2478/amcs-2014-0021

[2] M. Sridharan and T. Mota, "Towards combining commonsense reasoning and knowledge acquisition to guide deep learning," *Autonomous Agents and Multi-Agent Systems*, vol. 37, no. 1, Nov. 1, 2022. DOI: 10.1007/s10458-022-09584-4

[3] K. Basu, S. C. Varanasi, F. Shakerin, and G. Gupta, "Square: Semantics-based question answering and reasoning engine," *Electronic Proceedings in Theoretical Computer Science*, vol. 325, pp. 73–86, Sep. 19, 2020. DOI: 10.4204/eptcs.325.13

[4] S. Banerjee, "Generating complex explanations for artificial intelligence models: An application to clinical data on severe mental illness.," *Life (Basel, Switzerland)*, vol. 14, no. 7, pp. 807–807, Jun. 26, 2024. DOI: 10.3390/life14070807

[5] J. Lee and I. Kim, "Vision-language-knowledge co-embedding for visual commonsense reasoning.," *Sensors (Basel, Switzerland)*, vol. 21, no. 9, pp. 2911–, Apr. 21, 2021. DOI: 10.3390/s21092911

[6] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, "Unleashing the potential of prompt engineering for large language models.," *Patterns (New York, N.Y.)*, vol. 6, no. 6, pp. 101 260–101 260, May 8, 2025. DOI: 10.1016/j.patter.2025.101260

[7] A. Weichselbraun, J. Steixner, A. M. P. Brasoveanu, A. Scharl, M. C. Göbel, and L. J. B. Nixon, "Automatic expansion of domain-specific affective models for web intelligence applications," *Cognitive computation*, vol. 14, no. 1, pp. 1–18, Jan. 30, 2021. DOI: 10.1007/s12559-021-09839-4

[8] M. Slota, J. Leite, and T. Swift, "Splitting and updating hybrid knowledge bases," *Theory and Practice of Logic Programming*, vol. 11, no. 4-5, pp. 801–819, Jul. 6, 2011. DOI: 10.1017/s1471068411000317

[9] J. G. Wolff, "Information compression as a unifying principle in human learning, perception, and cognition," *Complexity*, vol. 2019, no. 1, pp. 1–38, Feb. 20, 2019. DOI: 10.1155/2019/1879746

[10] M. Kejriwal, "Knowledge graphs: A practical review of the research landscape," *Information*, vol. 13, no. 4, pp. 161–161, Mar. 23, 2022. DOI: 10.3390/info13040161

[11] F. Bentley, J. A. Swift, R. Cook, and S. A. Redsell, ""i would rather be told than not know" - a qualitative study exploring parental views on identifying the future risk of childhood overweight and obesity during infancy.," *BMC public health*, vol. 17, no. 1, pp. 684–684, Aug. 29, 2017. DOI: 10.1186/s12889-017-4684-y

[12] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges.," *Multimedia tools and applications*, vol. 82, no. 3, pp. 3713–3744, Jul. 14, 2022. DOI: 10.1007/s11042-022-13428-4

[13] K. Alahverdzhieva, A. Lascarides, and D. Flickinger, "Aligning speech and co-speech gesture in a constraint-based grammar," *Journal of Language Modelling*, vol. 5, no. 3, pp. 421–464, Jan. 18, 2018. DOI: 10.15398/jlm.v5i3.167

[14] W. Marek, A. Nerode, and J. B. Remmel, "A theory of nonmonotonic rule systems i," *Annals of Mathematics and Artificial Intelligence*, vol. 1, no. 1, pp. 241–273, Sep. 1, 1990. DOI: 10.1007/bf01531080

[15] H. Xiong, Y. Zhou, J. Liu, and Y. Cai, "Class-dependent and cross-modal memory network considering sentimental features for video-based captioning.," *Frontiers in psychology*, vol. 14, pp. 1 124 369–, Feb. 15, 2023. DOI: 10.3389/fpsyg.2023.1124369

[16] R. Moratz and J. O. Wallgrün, "Spatial reasoning with augmented points: Extending cardinal directions with local distances," *Journal of Spatial Information Science*, vol. 2012, no. 5, pp. 1–30, Dec. 19, 2012. DOI: 10.5311/josis.2012.5.84

[17] M. J. Santofimia, F. Moya, F. J. Villanueva, D. Villa, and J. C. López, "How intelligent are ambient intelligence systems," *International Journal of Ambient Computing and Intelligence*, vol. 2, no. 1, pp. 66–72, Jan. 1, 2010. DOI: 10.4018/jaci.2010010106

[18] T. A. Chang et al., "Global piqa: Evaluating physical commonsense reasoning across 100+ languages and cultures," *arXiv preprint arXiv:2510.24081*, 2025.

[19] F. Rancourt, P. Vondrlik, D. Maupomé, and M.-J. Meurs, "Investigating self-rationalizing models for commonsense reasoning," *Stats*, vol. 6, no. 3, pp. 907–919, Aug. 29, 2023. DOI: 10.3390/stats6030056

[20] H. Lieberman, H. Liu, P. Singh, and B. Barry, "Beating common sense into interactive applications," *Ai Magazine*, vol. 25, no. 4, pp. 63–76, Dec. 15, 2004.

[21] M. Oaksford and N. Chater, "The uncertain reasoner: Bayes, logic, and rationality," *Behavioral and Brain Sciences*, vol. 32, no. 1, pp. 105–120, Feb. 12, 2009. DOI: 10.1017/s0140525x0900051x

[22] C. Bartley, W. Liu, and M. Reynolds, "Aaai - enhanced random forest algorithms for partially monotone ordinal classification," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3224–3231, Jul. 17, 2019. DOI: 10.1609/aaai.v33i01.33013224

[23] M. A. El-Salam, A. K. Hussein, and A. A. Fahmy, "Understanding a simple arabic stories using event calculus," *American Journal of Applied Sciences*, vol. 10, no. 10, pp. 1298–1306, Oct. 1, 2013. DOI: 10.3844/ajassp.2013.1298.1306

[24] Z. Hu, J. Liu, Z. Liu, Y. Liu, Z. Xie, and Y. Song, "Rmath: A logic reasoning-focused datasets toward mathematical multistep reasoning tasks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 22, pp. 24 104–24 112, Apr. 11, 2025. DOI: 10.1609/aaai.v39i22.34585

[25] M. G. Voskoglou and A.-B. M. Salem, "Benefits and limitations of the artificial with respect to the traditional learning of mathematics," *Mathematics*, vol. 8, no. 4, pp. 611–, Apr. 16, 2020. DOI: 10.3390/math8040611

[26] C. Giraud-Carrier and T. Martinez, "An integrated framework for learning and reasoning," *Journal of Artificial Intelligence Research*, vol. 3, no. 1, pp. 147–185, Aug. 1, 1995. DOI: 10.1613/jair.93

[27] Y. Izmirlioglu and E. Erdem, "Reasoning about cardinal directions between 3-dimensional extended objects using answer set programming," *Theory and Practice of Logic Programming*, vol. 20, no. 6, pp. 942–957, Sep. 22, 2020. DOI: 10.1017/s1471068420000411

[28] A. Lieto et al., "A sensemaking system for grouping and suggesting stories from multiple affective viewpoints in museums," *Human–Computer Interaction*, vol. 39, no. 1-2, pp. 109–143, Aug. 9, 2023. DOI: 10.1080/07370024.2023.2242355

[29] J. Beverley and A. Hicks, "Evaluation and design of generalist systems (edges)," *AI Magazine*, vol. 44, no. 2, pp. 206–207, Jun. 13, 2023. DOI: 10.1002/aaai.12095

[30] J. Hernández-Orallo, "Evaluation in artificial intelligence: From task-oriented to ability-oriented measurement," *Artificial Intelligence Review*, vol. 48, no. 3, pp. 397–447, Aug. 19, 2016. DOI: 10.1007/s10462-016-9505-7

[31] K. Shen and M. Kejriwal, "Quantifying confidence shifts in a bert-based question answering system evaluated on perturbed instances.," *PloS one*, vol. 18, no. 12, e0295925–e0295925, Dec. 20, 2023. DOI: 10.1371/journal.pone.0295925