

PR #48200 完整报告

vllm-project/vllm

[Refactor] StructuredOutputManager x Speculative Decoding Refactor

合并时间: 2026-09-15 21:48

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48200>

执行摘要

- 一句话: 重构结构化输出与投机解码的约束起点契约
- 推荐动作: 值得精读, 且建议按「契约 → 消费方 → 测试」的顺序读: 先看 `vllm/v1/structured_output/__init__.py` 的 `_get_constraint_start()`, 它是整个 PR 唯一的语义来源; 再看 `scheduler.py` 三个调用点确认新契约的用法; 最后看 `test_structured_output_manager.py` 的 `FlowCase / _run_real_flow()` 理解验证方式。重点关注四个设计决策: (1) 用 `len + 1` 哨兵而非 `None` 表达「本窗口无约束」, 三个消费点的 `>= / > / <=` 各有语义, 不要「顺手统一」; (2) 降序扫描把语义从「第一次边界」改成「最后一次边界」, 这对交错思考模型更合理但改变了行为; (3) 断言降级为日志是刻意的 (避免 engine core 被打挂), 代价是失败请求要等 `accept` 阶段兜底; (4) 「reasoning parser 必须无状态」目前只是文档契约, 落地前不建议在生产上把 `stateful parser` 与投机解码 + 结构化输出组合使用。

功能与动机

RFC issue #48197 明确指出: `StructuredOutputManager` 的存在是因为「约束解码往往要等模型结束 reasoning 才生效」, 而配合投机解码后, `grammar` 非法的 `draft token` 需要被拒绝、被拒绝的 `draft` 不能推进 `grammar`、`reasoning` 结束边界还可能落在 `draft` 窗口中间, 调度器与 `manager` 必须「careful dance」才能正确, 直接导致了 #34650 这类 bug。Issue 里提出的改造目标是: 「Let `StructuredOutputManager` handle the token validation and grammar advancement」, 删除 `should_advance(request)` 与 `trim_reasoning_for_advance(verified tokens)`, 新增 `validate_tokens(draft tokens)` 与 `accept_tokens(verified tokens)`, 并让三个入口共享同一个 `_get_constraint_start()` helper。PR body 引用 #44297 的 repro 作为回归基准, 并给出 JSON 约束场景下 main 68% 成功率、本 PR 100% 的对比数据 (其中 main 的失败由 #44993 修复, 本 PR 保持通过)。

实现拆解

1. 新增统一契约 helper: 在 `vllm/v1/structured_output/__init__.py` 中新增 `_get_constraint_start(request, spec_tokens, spec_tokens_committed=False)`, 返回「窗口内第一个需要被 `grammar` 约束的下标」, 无约束时返回 `len(spec_tokens) + 1` 哨兵。它合并了原先散落在 `should_advance()`、`trim_reasoning_for_advance()`、`should_fill_bitmask()` 中的 `reasoning` 边界判断: 先看 `enable_in_reasoning / reasoning_ended` / 是否有 `reasoner` 的快路径, 再优先调用

ParserEngineReasoningAdapter.find_reasoning_end_offset(), 否则回退到自右向左收缩窗口调用 is_reasoning_end_streaming()。request.py 中因此删除了不再需要的 reasoning_end_token_index 字段 (旧的绝对索引方案被窗口内相对下标取代)。

2. 重写 bitmask 与 grammar 推进逻辑: grammar_bitmask() 串行分支改为「constraint_start 之前不约束、之后才 real-fill」, 用 seen_padding 在首个 -1 后恢复 memset (消除 #44006 性能与扩散模型正确性回归), 用 failed 标志把旧的 raise AssertionError 降级为 logger.error + 剩余行不再约束; 试探性推进的 grammar 状态仍然通过 grammar.rollback(state_advancements) 回滚, 保持幂等。并行分支的 apply_bitmask 也从 should_fill_bitmask() 改为 _get_constraint_start(request, ()) == 0。
3. 新增两个公共入口: validate_tokens() 先把 -1 padding 下沉剥离 (strip_speculative_padding), 只对约束后缀调用 grammar.validate_tokens(), 保留未约束前缀; accept_tokens() 用 spec_tokens_committed=True 复用同一 helper, 只有 constraint_start > len(new_token_ids) 才提前返回 True, 否则置位 structured_req.reasoning_ended = True (这是 reasoning_ended 唯一的 latch 点) 并推进 grammar, 被拒绝时返回 False。
4. 调度器侧收敛: vllm/v1/core/sched/scheduler.py 不再 import StructuredOutputGrammar 与 strip_speculative_padding; update_from_output() 用 not accept_tokens(request, new_token_ids) 直接判定 FINISHED_ERROR, update_draft_token_ids() 与 update_draft_token_ids_in_output() 都用 validate_tokens(request, spec_token_ids), 后者仍把被裁掉的数量换算成 -1 回填并写入 num_invalid_spec_tokens。
5. 契约文档化与解析器对齐: vllm/parser/abstract_parser.py 为 is_reasoning_end / is_reasoning_end_streaming 补上「Must be a pure function」约束, 并把 find_reasoning_end_offset() 的「窗口内未发现边界」返回值从 None 改为 len(token_ids); vllm/parser/engine/parser_engine.py 同步该语义, 使调用方可以用 offset + 1 直接得到约束起点。
6. 测试整合 (配套改动): 删除 tests/v1/structured_output/test_reasoning_structured_output.py (557 行) 与 tests/v1/spec_decode/test_mtp_structured_output.py 中的大部分用例, 新增 tests/v1/structured_output/test_structured_output_manager.py, 用 FlowCase 在真实 xgrammar/guidance 后端上跑 validate → bitmask → accept → 再取 bitmask 全流程; tests/v1/core/test_scheduler.py / test_async_scheduler.py 改为 mock manager.accept_tokens, test_mtp_structured_output.py 保留 xgrammar 终止语义与引擎 reasoner 的窗口探测用例。

关键文件:

- vllm/v1/structured_output/__init__.py (模块 结构化输出; 类别 source; 类型 core-logic ; 符号 _get_constraint_start, validate_tokens, accept_tokens, grammar_bitmask) : 整个重构的语义中心: 新增 _get_constraint_start() 统一约束起点, 新增 validate_tokens()/accept_tokens() 两个公共入口, 重写 grammar_bitmask() 的 padding 与失败降级逻辑, 同时删除 should_advance()/trim_reasoning_for_advance()/should_fill_bitmask()。

- `vllm/v1/core/sched/scheduler.py` (模块 调度器; 类别 source; 类型 dependency-wiring; 符号 `update_from_output`, `update_draft_token_ids`, `update_draft_token_ids_in_output`) : 消费新契约的三个调用点被收敛: 不再 import grammar 与 padding 工具, `update_from_output` 直接依赖 `manager.accept_tokens` 的布尔返回, 两个 draft 校验入口统一调用 `manager.validate_tokens`。
- `tests/v1/structured_output/test_structured_output_manager.py` (模块 结构化输出; 类别 test; 类型 test-coverage; 符号 `FlowCase`, `MockReasoner`, `_run_real_flow`, `_row_pattern`) : 测试策略的载体: 用 `FlowCase` 在真实 `xgrammar/guidance` 后端上跑 `validate` → `bitmask` → `accept` → 再取 `bitmask` 的完整时序, 替代原先按方法逐个 mock 的测试。
- `tests/v1/spec_decode/test_mtp_structured_output.py` (模块 投机解码; 类别 test; 类型 test-coverage; 符号 `_EngineReasonerStub`, `test_bitmask_engine_reasoner_ends_midwindow_with_padding`, `test_xgrammar_accept_tokens_stops_at_termination`, `test_xgrammar_validate_tokens_stops_at_termination`) : 保留并改写为「后端 + 引擎 reasoner」的边界用例: 删除 357 行 `manager` 流程测试 (迁往新 suite), 新增 `_EngineReasonerStub` 验证引擎路径不会重扫草案前缀, 并保留 `xgrammar` 终止语义测试。
- `vllm/parser/abstract_parser.py` (模块 解析器契约; 类别 source; 类型 documentation; 符号 `is_reasoning_end`, `is_reasoning_end_streaming`, `find_reasoning_end_offset`) : 把本 PR 依赖的隐式假设写成显式契约: `is_reasoning_end` / `is_reasoning_end_streaming` 必须是纯函数, `find_reasoning_end_offset` 的「未发现」返回值语义由 `None` 改为 `len(token_ids)`。
- `vllm/v1/structured_output/request.py` (模块 结构化输出; 类别 source; 类型 data-contract; 符号 `StructuredOutputRequest.reasoning_end_token_index`) : 删除 `StructuredOutputRequest.reasoning_end_token_index`, 标志旧的「绝对索引」方案被窗口内相对下标方案取代, 是本 PR 数据契约层面的关键清理。
- `vllm/parser/engine/parser_engine.py` (模块 解析器引擎; 类别 source; 类型 core-logic; 符号 `find_reasoning_end_offset`) : `find_reasoning_end_offset` 的兜底返回值从 `None` 改为 `len(token_ids)`, 让上游可以无条件用 `offset + 1` 计算约束起点, 是本 PR 快路径的前提。
- `tests/v1/core/test_scheduler.py` (模块 调度器; 类别 test; 类型 test-coverage; 符号 `test_update_draft_token_ids_strips_ngram_padding`, `test_update_draft_token_ids_in_output_strips_padding`, `test_abort_request_when_structured_output_fsm_cannot_advance`) : 跟随 `scheduler` 接口变化改写: `abort` 用例改为 mock `manager.accept_tokens`, `ngram padding` 用例改为依赖 `StructuredOutputRequest.reasoning_ended` 而非 `monkeypatch.should_advance`。

关键符号: `_get_constraint_start`, `validate_tokens`, `accept_tokens`, `grammar_bitmask`, `strip_speculative_padding`, `update_from_output`, `update_draft_token_ids`, `update_draft_token_ids_in_output`, `find_reasoning_end_offset`

关键源码片段

[vllm/v1/core/sched/scheduler.py](#)

消费新契约的三个调用点被收敛：不再 import grammar 与 padding 工具，update_from_output 直接依赖 manager.accept_tokens 的布尔返回，两个 draft 校验入口统一调用 manager.validate_tokens。

```
# vllm/v1/core/sched/scheduler.py
# 三个调用点收敛为两个 manager 方法，调度器不再直接接触 grammar。

# 1) update_from_output(): 采样 / 验证后的 token 落地时推进 grammar。
# manager 内部用 _get_constraint_start() 裁掉 reasoning 前缀，
# 因此这里不再需要 should_advance() + trim_reasoning_for_advance() 两步。
if new_token_ids and not self.structured_output_manager.accept_tokens(
    request, new_token_ids
):
    logger.error(
        'Unexpected: grammar rejected tokens %s for request %s. '
        'Terminating request.',
        new_token_ids,
        req_id,
    )
    request.status = RequestStatus.FINISHED_ERROR
    request.resumable = False
    stopped = True

# 2) update_draft_token_ids(): ngram/ 草案生成后立即做一次约束校验。
# -1 padding 的剥离已下沉到 manager，这里只透传。
request.spec_token_ids = self.structured_output_manager.validate_tokens(
    request, spec_token_ids
)

# 3) update_draft_token_ids_in_output(): 写入 scheduler output 前同名校验，
# 被裁掉的数量换算成 -1 回填，供 rejection sampler 识别为占位。
spec_token_ids = self.structured_output_manager.validate_tokens(
    request, spec_token_ids
)
num_invalid_tokens = orig_num_spec_tokens - len(spec_token_ids)
if num_invalid_tokens:
    spec_token_ids.extend([-1] * num_invalid_tokens)
    num_invalid_spec_tokens[req_id] = num_invalid_tokens
```

tests/v1/structured_output/test_structured_output_manager.py

测试策略的载体：用 FlowCase 在真实 xgrammar/guidance 后端上跑 validate → bitmask → accept → 再取 bitmask 的完整时序，替代原先按方法逐个 mock 的测试。

```
# tests/v1/structured_output/test_structured_output_manager.py
# 测试策略：不 mock grammar，而是用真实 xgrammar/guidance 后端，
# 按调度器真实时序把一条请求跑完 validate -> bitmask -> accept -> bitmask。
@dataclass(frozen=True)
class FlowCase:
    """One validate -> bitmask -> accept -> post-accept bitmask scenario."""
```

```

raw_drafts: tuple[str, ...] # 草案字面量; EOS 用哨兵, -1 由流程自行补齐
expected_row_pattern: str # 每行期望 C (约束) / U (未约束)
expected_reasoning: bool | None # 流程结束后 reasoning_ended 的期望值
expect_terminated: bool # 流程结束后 grammar 是否应终止
prefix: str = '' # 已提交前缀, 用于构造 reasoning / JSON 边界
expected_validated: tuple[str, ...] | None = None
reasoning_ended: bool | None = False
xfail_guidance: str | None = None # guidance 在终止态 EOS 校验上的已知差异

```

```

def _row_pattern(bitmask) -> str:
    # 整行全 -1 视为未约束 (U), 否则视为被 grammar 约束 (C)。
    return ''.join('U' if (row == -1).all() else 'C' for row in bitmask)

```

```

def _run_real_flow(
    manager: StructuredOutputManager,
    request: Request,
    raw_drafts: list[int],
    expected_validated: list[int],
    expected_row_pattern: str,
    expected_reasoning: bool | None,
    expect_terminated: bool,
) -> None:
    assert len(expected_validated) <= len(raw_drafts)

    validated = manager.validate_tokens(request, list(raw_drafts))
    assert validated == expected_validated

    # 被拒绝的草案用 -1 补齐后再进入 bitmask, 直接覆盖 padding 场景。
    padded = validated + [-1] * (len(raw_drafts) - len(validated))
    bitmask = manager.grammar_bitmask(
        requests={request.request_id: request},
        structured_output_request_ids=[request.request_id],
        scheduled_spec_decode_tokens={request.request_id: padded},
    )
    assert bitmask is not None
    assert bitmask.shape[0] == len(raw_drafts) + 1
    assert _row_pattern(bitmask) == expected_row_pattern

    # grammar_bitmask() 必须回滚试探性推进, 重复调用结果稳定 (幂等)。
    assert manager.validate_tokens(request, list(raw_drafts)) == expected_validated

    structured_req = request.structured_output_request
    grammar = structured_req.grammar
    assert not grammar.is_terminated()
    # bitmask[i] 是「已接受前 i 个草案」后的 grammar 状态,
    # 因此提交 accepted 前缀后重新取到的 bitmask 必须与 row i 一致。
    expected_post_accept_row = bitmask[len(expected_validated)].copy()

```

```

request.append_output_token_ids(list(expected_validated))
assert manager.accept_tokens(request, list(expected_validated))

post_accept_bitmask = manager.grammar_bitmask(
    requests={request.request_id: request},
    structured_output_request_ids=[request.request_id],
    scheduled_spec_decode_tokens={},
)
assert post_accept_bitmask is not None
assert post_accept_bitmask.shape[0] == 1
assert (post_accept_bitmask[0] == expected_post_accept_row).all()
assert structured_req.reasoning_ended == expected_reasoning
assert grammar.is_terminated() is expect_terminated

```

评论区精华

1. 引擎级致命断言 (abmfy, CHANGES_REQUESTED → 已解决)

「This assert replaces the old `post_reasoning_end_in_window` tolerance, and it can fire in practice... When it fires, the `AssertionError` propagates out of `schedule()` and takes down the whole engine core」

abmfy 用 CPU-only 的 manager 级 probe 复现出 `Step3p5ReasoningParser` 的一次性 latch `_end_token_pending` 会让 `validate_tokens` 与 `grammar_bitmask` 两次 walk 得到不同的 `constraint_start`, 未验证的 marker token 落进 `i >= constraint_start` 区域并触发断言。作者在 [9bd40d7](#) 中把断言换成 `failed` 标志 + `logger.error` (fail-closed), 并解释「We could assert False here but that would crash the entire engine. The right fix is the fail the request but that requires some additional plumbing that should probably be a follow up」。顺带修掉了 `state_advancements += 1` 无条件自增在 `python -O` 下多回滚一个 token 的隐患。

2. reasoning parser 的纯度契约 (design, 部分未解决, 形成前置依赖)

abmfy: 「This walk runs from three entry points per step... which implicitly requires `is_reasoning_end_streaming` to be pure」 yzong-rh: 「`is_reasoning_end_streaming` should probably be pure and idempotent. Will fix `Step3p5ReasoningParser` in another PR」

作者拒绝在热路径加 `__debug__` 双重探针 (理由是性能与「Two calls can return the same value even with state mutation」), 改为在 `abstract_parser.py` 写入纯度契约。abmfy 最终 approve 时给出排序条件: 「#53174 should land before this PR」, 并提示应在 PR 描述里声明该依赖。

3. 热路径 O(context) 拷贝 (performance, 已修复)

abmfy: 「every structured-output request pays an O(context) copy per step for its entire lifetime... ~355µs @200k, per request per step on the engine-core critical path — That's the regression class #50886 just removed」

两处修复：32a5a63 让切片发生在 reasoning_ended / 无 reasoner 的早退之后，并对空窗口提前返回；ee9e30f 去掉 islice 与星号解包，改用 delta_ids = spec_tokens[:num_spec_tokens]。作者实测（Qwen3.5-2B, MTP + reasoning + JSON）输出吞吐 +19%、Mean ITL -16%。

4. -1 padding 行的填充行为 (correctness, 已修复)

abmfy: 「the old loop stopped real-filling at the first -1... For diffusion LLMs under async scheduling it's a correctness change」

cba6819 恢复「首个 -1 之后翻转为 memset」，seen_padding 同时作用于 bonus row；作者承认「My bad」。

5. 测试覆盖缺口 (testing, 未采纳)

abmfy: 「The suite's only reasoner is stateless and delta-only, so the invariant the whole design rests on... has zero coverage」

作者回应 MockReasoner 实现的就是 refactor 前已假设的无状态契约，多 token marker 与基类默认实现属于 parser 层测试，「More or less won't fix」；但采纳了删除 test_scheduler.py / test_async_scheduler.py 中已失效的 grammar mock 残留。

6. 哨兵语义与 re-open 窗口 (design, 留白)

abmfy: 「the len(spec_tokens) + 1 sentinel forces three different comparisons at the three consumers... Anyone 'harmonizing' these later gets the silent-permanent-loss failure mode」

作者保留哨兵（计划缓存 _get_constraint_start 结果，None 留给 cache miss），只补文档。mgoin 随后指出 Kimi K3 parser 下「close reasoning → 非法 token → reopen reasoning」会让整窗不被 mask，作者确认该场景在 main 上同样没有处理，并解释降序扫描只能看到最后一次边界：「Main handled this inconsistently... With this PR, we scan right-to-left only across all three stages」。

- grammar_bitmask 断言可能打挂整个 engine core (correctness): 作者在 9bd40d7 中把断言降级为 failed 标志 + logger.error，剩余行不再约束 (fail-closed)；FINISHED_ERROR 需要新返回通道，作者声明留作后续。
- is_reasoning_end_streaming 的纯度 / 幂等契约无法被代码强制 (design): 契约文档化 + 拆出 #53174；abmfy approve 时要求「#53174 should land before this PR」，并建议在 PR 描述里声明依赖。有状态 parser 的静默丢约束与 off-by-one 在本分支仍可复现。
- 结构化输出请求每步付出 O(context) 拷贝 (performance): 作者在 32a5a63 把切片移到早退之后并加空窗口提前返回，ee9e30f 用 delta_ids = spec_tokens[:num_spec_tokens] 替代 islice/ 星号解包；实测 ~56μs vs main ~141μs per request per step @8k。只读 Sequence 视图留作 TODO。
- 首个 -1 之后是否仍真实填充 bitmask 行 (correctness): 作者在 cba6819 恢复 seen_padding 逻辑并覆盖 bonus row；同时说明 #44006 的 bonus row 期望被有意翻转，安全性依赖 rejection sampler 自动拒绝 PLACEHOLDER 草案这一未写明的约定。

- 新 flow suite 缺少有状态 reasoner 与多 token marker 覆盖 (testing): 作者以「reasoning-end 检查应无状态」与「parser 层测试不应混入 manager 测试」为由不采纳, 仅补文档与清理失效 mock; abmfy 在 approve 中再次提示可加 __debug__ 双重探针。
- len + 1 哨兵与 re-open 窗口语义 (design): 作者保留哨兵 (计划缓存 _get_constraint_start, None 留给 cache miss) 并补注释; 对 re-open 场景确认 main 同样未处理, 属于既有空白而非本次回归, 未在本 PR 内修复。

风险与影响

• 风险:

1. 核心路径回归面广: grammar_bitmask() 串行分支被整段重写, 同时承载 AR、扩散模型 (is_diffusion + canvas_length)、投机解码 padding、异步调度四种组合。seen_padding / failed / constraint_start 三个条件的组合语义 (如 bonus row 在 constraint_start == len(req_tokens) 时仍要 real-fill) 没有不变式测试兜底, active_later_token_invalid 等 FlowCase 的期望值已与删除的 #44006 回归用例相反, 安全性依赖「rejection sampler 会拒绝 PLACEHOLDER(-1)」这一未写进代码的约定。
2. 隐性契约: reasoning parser 必须是纯函数。三个入口 (validate_tokens / grammar_bitmask / accept_tokens) 每 step 都会重走同一窗口, 任何持有跨调用状态的 parser (当前树内的 Step3p5ReasoningParser._end_token_pending) 都会出现「静默丢约束」或 off-by-one, 且 $i + 1 \geq 1$ 使漏检不可恢复、请求会以无约束输出继续服务。本 PR 只把致命崩溃降级为日志, 根因留给 #53174 与文档契约。
3. 未决的 re-open 窗口: mgoin 指出 close $\rightarrow$ 中间 token $\rightarrow$ reopen 的窗口在三个阶段都不再被约束; 虽然 main 的处理本身也不一致, 但这一路径在 Kimi K3 + XGrammar 上可复现, 若上游再次出现类似 #43388 的调度变化, 风险会被放大。
4. 性能未完全收敛: _get_constraint_start() 每 step 最多被调用三次, request.all_token_ids.copy() 仍在 no-fire 路径上物化; 作者的实测显示合并 #55223 后 TTFT / 吞吐与 main 持平, 但 TODO: Build a read-only Sequence view 尚未落地, 长上下文 (108k+) 下每步仍有 ~28 μ s/ 请求量级的额外拷贝。
5. API 兼容性: StructuredOutputRequest.reasoning_end_token_index 被删除, should_advance / trim_reasoning_for_advance 从 manager 公共接口消失; 任何下游 fork 或插件式 reasoning parser 若引用这些符号会直接 ImportError / AttributeError。
 - 影响: 用户侧: reasoning 模型 (Qwen / DeepSeek / GLM 等) + JSON/STRUCTURAL_TAG 约束 + 投机解码的组合从「部分请求静默不约束」变为契约统一; PR body 的 50 次请求对比中 JSON 约束成功率从 68% 提升到 100%, 作者在 MTP + reasoning 场景下实测吞吐 +19%、Mean ITL -16%。系统侧: 调度器热路径少了两层间接调用与一次 grammar 依赖, Scheduler 不再 import StructuredOutputGrammar; grammar 推进的职责边界从「调度器 + manager 协商」变为「manager 单点决策」, 后续新增约束类型或推理后端只需遵循一个 helper。团队侧: 这是尚未落地的 RFC 的第一次主干实现, #53174 (Step3p5 parser 无状态化) 成为事实上的后续依赖; 同时删除了 900+ 行旧测试, 测试守护从「按方法逐个 mock」转为「按真实后端全流程」的范式, 后续扩展 reasoning 场景需要补 FlowCase 而非改 mock。
 - 风险标记: 核心路径变更, 隐性纯度契约, 依赖 PR#53174 先落地, 哨兵值语

义易踩坑，测试覆盖缺口，长上下文拷贝未收敛

关联脉络

- PR #56200 [Refactor] Derive `is_reasoning_end` from the engine grammar: 同一条功能线的前序重构，同样围绕 `is_reasoning_end / find_reasoning_end_offset` 与 `vllm/parser/engine/parser_engine.py`；本 PR 在此基础上把边界探测下沉到 `_get_constraint_start` 并删掉多模型手工覆写。
- PR #53174 Step3p5 reasoning parser 修复（讨论中给出的链接，上下文未提供正式标题）：review 中确定的前置依赖：Step3p5ReasoningParser 的一次性 `_end_token_pending latch` 会让三次 walk 得到不同 `constraint_start`，造成引擎崩溃（已降级）与静默丢约束（未修）；abmfy 要求其先于本 PR 合入。
- PR #44297 strict tool calling (STRUCTURAL_TAG) 约束修复（PR body 引用，未提供正式标题）：PR body 用其 repro gist 作为回归基准，RFC #48197 也把它列为修复仅覆盖 STRUCTURAL_TAG 场景、JSON/REGEX 仍受损的前序工作。
- PR #44993 JSON/REGEX 等非 STRUCTURAL_TAG 约束在投机解码下的修复（issue #48197 引用，未提供正式标题）：RFC 中标记为修复 `should_advance` 差异与 `num_output_placeholders` 推断问题的 PR；PR body 的 JSON 约束对比数据（main 68% → 本 PR 100%）以它为基线。