

PR #39136 完整报告

sgl-project/sglang

[Fix] Preserve GLM tool argument types across JSON Schema unions

合并时间: 2026-09-13 13:47

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39136>

执行摘要

- 一句话: 修复 GLM 工具参数类型在 Schema 联合中被过早定型
- 推荐动作: 值得精读, 尤其是负责 function calling / 结构化输出与流式一致性的同学。建议重点关注三处设计决策: (a) `_argument_types` 用「约束集求交 + union 求并」替代单向推断, 以及 `const/enum` 只在与已完成取值相容时参与收窄的条件; (b) `_needs_argument_context` 用「是否需要缓冲」把延迟定型的影响限制在真正含 union 的工具上, 而非全局缓冲; (c) `_coerce_numeric_string` 刻意保留 `number/integer` 的非对称 ("1.5" 对 `integer` 保持字符串)。阅读顺序建议先看 `glm47_moe_detector.py` 的类型解析函数, 再看 `test_glm47_schema_types.py` 的 `check_arguments` 断言方式 (值 + 原生类型双重校验、多分块粒度)。后续行动建议: 推动 `glm4_moe_detector/ling3_detector` 同步修复, 并在 #35625 合并前确认两者的方案不会冲突。

功能与动机

PR body 明确指出: GLM-4.7/GLM-5 的工具调用解析器「在完整参数值可用之前就确定了一个推断出的 JSON Schema 类型」, 于是同一段生成在完整响应与流式响应下产出不同甚至错误的参数; 典型例子是声明为 `{"type": ["string", "null"]}` 的字段, 模型输出的 `null` 被序列化成字符串 `"None"` 或 `"null"` 而不是 JSON `null`。同一早定型的取舍还波及本地引用、`enum`, 以及判别字段出现在值之后的根 `anyOf/oneOf`; 流式场景下数字前缀可能在解析器尚不知道完整值是否为合法 JSON 时就被发出。作者说明本改动建立在已合并的 #36626 (在顶层 schema 组合子内发现属性) 之上, 但避免「选中第一个重复属性类型」; 并指出 #35564 是流式 / 非流式一致性的跟踪 issue, #35625 是统一修复、#36688 是较窄的 GLM 版本, 三者覆盖了 `schema-less` 缓冲与非法数字 token, 却不覆盖本 PR 处理的 `nullable` 强制转换、本地引用、`enum/const` 收窄与同级判别的根 union。

实现拆解

1. 类型解析层整体重写 (`glm47_moe_detector.py`): 删除基于 `get_schema_properties` + `infer_type_from_json_schema` 的旧 `get_argument_type`, 改为新增 `_json_type`、`_json_equal`、`_matches_discriminators`、`_argument_types`、`_get_argument_types`、`get_argument_type` 一组函数。核心 `_argument_types` 先收集若干 `constraints` 再 `set.intersection: type` (字符串或数组, `number` 自动扩展出 `integer`)、`const/enum` (仅当允许值与已完成取值相容时才收窄)、`anyOf/oneOf` (先按同级判别参数过滤分支再取并集)、`allOf` (取交集)、本地 `$ref` (支持 `#`、`#/...` 与 `~0/~1` 解码)。seen 以

id(schema) 记录，遇到环形 \$ref 直接返回 None 终止递归；悬空 \$ref 被 except (KeyError, IndexError, TypeError, ValueError) 吞掉并视为无约束。get_argument_type 只在类型唯一时返回类型名，{"number", "integer"} 归一为 number，多类型一律返回 None，从接口层面放弃「提前猜一个类型」。

2. 新增缓冲决策入口 _needs_argument_context: 它用内部 visit 递归遍历 schema 图（跟随 \$ref、进入 anyOf/oneOf/allOf），收集所有 properties 键并标记是否存在真正的 union（allOf 不算），对存在 union 的工具逐 key 调 _argument_types 判断类型是否不唯一；返回 True 时流式侧需要对参数做缓冲，等值完整后再定型。材料中没有展示该函数的具体调用点，按 PR 描述它是「缓冲 ambiguous 值与数值前缀」的判定依据。
3. 流式侧延迟定型：_get_value_type 不再在 self._current_value 仍为空时做自动推断（旧实现因此恒返回 string，把所有未声明 schema 的数字都加了引号），改为缓存声明的类型、把无法安全判断的取值推迟到 </arg_value>；普通 string 字段保持增量输出，由 test_plain_strings_still_stream_before_value_completes 断言首个分块就带出 "hello。同时用「已完成参数 + 同级判别字段」来选择根 union 分支，使 kind 与 value 无论谁先到达都能定对类型。
4. 数值转换语义收敛：新增 _convert_to_integer 与 _coerce_numeric_string，parse_arguments 的 Strategy 1/2 统一改为 _coerce_numeric_string(json.loads(...), arg_type)，把「只有字符串才参与强制转换」和「integer 不接受 1.5 这类小数」两条规则集中到同一处；维护者追加的提交正是落在这里。
5. 测试与 CI 配套：新增 test/registered/unit/function_call/test_glm47_schema_types.py，register_cpu_ci(est_time=2, suite="base-a-test-cpu") 注册进 CPU CI。check_arguments 对每个用例跑 size=0/1/7/len(text) 四种分块（0 表示非流式基线），既比对解析出的 JSON 值，也用 assertIs(type(actual[key]), type(expected[key])) 卡住原生类型，覆盖 nullable、\$defs/definitions 引用与环、根 allOf、enum 反例、123abc 这类非法数字、判别字段顺序调换与解析状态在多次 tool call 间的重置。无文档、kernel 或部署侧改动。

关键文件：

- python/sglang/srt/function_call/glm47_moe_detector.py（模块 工具解析器；类别 source；类型 core-logic；符号 _json_type, _json_equal, _matches_discriminators, _argument_types）：唯一源码改动文件，重写了工具参数类型推断（_argument_types 等）、缓冲决策（_needs_argument_context）与数值强制转换（_coerce_numeric_string），直接决定 GLM-4.7 流式与非流式工具调用参数是否一致。
- test/registered/unit/function_call/test_glm47_schema_types.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestGlm47SchemaTypes, check_arguments, test_field_types, test_local_references_and_cycles）：新增的 CPU 单元测试，用 check_arguments 在 1 字符 / 7 字符 / 整段三种分块下同时校验解析值与原生 Python 类型，是这次类型收窄与流式缓冲行为的回归护栏。

关键符号：_argument_types, _matches_discriminators, _get_argument_types, get_argument_type, _needs_argument_context, _json_type, _json_equal, _coerce_numeric_string, _convert_to_integer, parse_arguments, check_arguments

关键源码片段

test/registered/unit/function_call/test_glm47_schema_types.py

新增的 CPU 单元测试，用 `check_arguments` 在 1 字符 / 7 字符 / 整段三种分块下同时校验解析值与原生 Python 类型，是这次类型收窄与流式缓冲行为的回归护栏。

```
class TestGlm47SchemaTypes(unittest.TestCase):
    def check_arguments(self, schema, expected, raw_values=None):
        # 同一份 schema / 期望值，分别走非流式与 1 字符、7 字符、整段三种分块流式
        tools = [
            Tool(type="function", function=Function(name="inspect", parameters=schema))
        ]
        pairs = []
        for key, value in expected.items():
            # raw_values 允许用例单独指定模型侧原始文本（例如 "null" 还是 "null"）
            raw = (raw_values or {}).get(
                key, value if isinstance(value, str) else json.dumps(value)
            )
            pairs.append(f"<arg_key>{key}</arg_key><arg_value>{raw}</arg_value>")
        text = "<tool_call>inspect" + "".join(pairs) + "</tool_call>"
        for size in (0, 1, 7, len(text)):
            with self.subTest(schema=schema, expected=expected, chunk_size=size):
                detector = Glm47MoeDetector()
                if size == 0:
                    # size == 0 走 detect_and_parse，作为非流式基线
                    calls = detector.detect_and_parse(text, tools).calls
                else:
                    calls = []
                    for offset in range(0, len(text), size):
                        calls.extend(
                            detector.parse_streaming_increment(
                                text[offset : offset + size], tools
                            ).calls
                        )
                # 所有分块的参数片段拼接后必须是一段合法且等值的 JSON
                actual = json.loads("".join(call.parameters for call in calls))
                self.assertEqual(actual, expected)
                # 值相等还不够：null 不能被序列化成 "None"/"null"，必须卡住原生类型
                for key in expected:
                    self.assertIs(type(actual[key]), type(expected[key]))

    def test_numeric_prefix_does_not_commit_to_invalid_json(self):
        # 数值型参数在值闭合前必须缓冲："123" 不能先于 "abc" 发出，
        # 否则客户端会先看到 123 再看到非法 JSON
        tools = [
            Tool(
                type="function",
                function=Function(
                    name="inspect",
```

```

        parameters={"properties": {"value": {"type": "number"}}},
    ),
)
]
detector = Glm47MoeDetector()
prefix = detector.parse_streaming_increment(
    "<tool_call>inspect<arg_key>value</arg_key><arg_value>123", tools
).calls
self.assertNotIn("123", "".join(call.parameters for call in prefix))
suffix = detector.parse_streaming_increment(
    "abc</arg_value></tool_call>", tools
).calls
self.assertEqual(
    json.loads("".join(call.parameters for call in prefix + suffix)),
    {"value": "123abc"},
)

```

评论区精华

本 PR 无行内 review 评论，讨论集中在 Issue 线程中的一次「维护者代改」：

JustinTong0323 以 APPROVED 并在评论里说明 GitHub 不允许维护者向 org 拥有的 fork 推送，因此把修复提交推到上游分支 [xinyuan/glm47-39136-review-fixes \(25c26ddd5f\)](#) 请作者 cherry-pick。该提交做了四件事：删除 ("auto", "number", "integer") 守卫之后已经不可达的两个 `number` 分支；修正 integer 强制转换——带引号的 "1.5" 落到 {"type": "integer"} 字段时不再变成 float 1.5 而保持字符串，并新增用例固定这种 `number/integer` 的非对称；清理 `test_field_types` 与 `test_native_strings_reject_disallowed_json_types_and_enum_values` 之间重复的用例（`test_root_unions_preserve_each_value_type` 的 `anyOf/oneOf` 循环已被 `test_completed_siblings_disambiguate_root_union` 完全覆盖，只保留根 `allOf` 用例）并删掉复述性注释；本地跑过 11 个 schema-types 测试与既有 `TestGlm47MoeDetector`，ruff 干净。作者回复「done!」后维护者执行 `/tag-and-rerun-ci`。评论末尾还留了一条本 PR 范围外的提醒：[glm4_moe_detector/ling3_detector](#) 存在同类问题，留作后续。

- integer 字段的字符串强制转换非对称性 (correctness): 已采纳并合并；number 允许 "1.5" → 1.5，integer 只接受可转成 int 的字符串，否则原样保留为字符串。
- 删除 ("auto", "number", "integer") 守卫后的不可达分支 (design): 已删除，逻辑等价但可读性提升。
- 测试用例重复与复述性注释清理 (testing): 已按建议裁剪，schema-types 的 11 个测试与既有 `TestGlm47MoeDetector` 测试本地通过。
- `glm4_moe_detector / ling3_detector` 的同类缺陷留作后续 (design): 明确为后续工作，本 PR 不做；导致 GLM 各解析器在同类 schema 下行为短期继续分叉。
- 维护者无法向 org 拥有的 fork 推送的协作流程 (other): 提交已被作者合入，PR 以 APPROVED 状态合并。

风险与影响

- 风险：1) 同族解析器未同步导致行为分叉：改动只落在 glm47_moe_detector.py，而 glm4_moe_detector.py (glm/glm45) 与 ling3_detector 携带几乎相同的早定型逻辑，维护者已在评论中确认留作后续；在修复落地前，同一份工具 schema 在不同 GLM 解析器下仍会给出不同的流式 / 非流式参数类型，客户端无法自行规避。2) 流式粒度与时延的显式权衡：类型不确定与数值型参数被缓冲到 `</arg_value>` 才发出，首包时延变大、参数分片变粗；对大对象 / 长数字参数的增量体验下降，PR body 承认这是有意为之但未给出量化数据。3) 递归 Schema 解析的 CPU 开销与边界：`_argument_types` 与 `_needs_argument_context` 都会遍历 schema 图，复杂嵌套工具定义（深层 `$defs` + 多层 `anyOf`）会带来额外开销；seen 按 `id()` 去重可防环，但 `id` 复用与同构 schema 的重复计算没有缓存，也未看到复杂度上限保护。4) 接口语义变更的潜在外部影响：模块级 `get_argument_type` 由「尽量返回一个类型字符串」改为「多类型返回 None」，若本仓库其它检测器或下游代码导入该符号，行为会随之变化；材料中未展示其它调用点，无法确认影响面。5) 测试与 CI 覆盖不完整：新测试只针对 `Glm47MoeDetector`，未覆盖 `glm4_moe_detector`；PR body 亦说明结构标签能力测试需要更新的本地 `xgrammar`，未覆盖本次改动；CI 状态显示 Extra 与 AMD ROCm 10 lane 为 ，需要确认这些失败是否与本改动无关。
- 影响：用户侧：使用 GLM-4.7/GLM-5 做 tool calling 且 schema 含 nullable、enum/const、本地 `$ref` 或根 union 的场景，流式与非流式会返回一致且类型正确的参数（null 不再变成 "null"，123abc 不再变成非法 JSON），减少客户端 `json.loads` 失败与业务侧类型错乱。系统侧：只影响 function call 解析路径，不触及模型执行、注意力或 kernel，单请求额外开销是 schema 遍历与参数缓冲，量级远小于推理本身；对使用纯 string 参数的工具几乎无感知。工程侧：确立了「类型推断推迟到取值完整、union 用判别字段收窄」的实现范式，是 #35625 统一修复的可复用参考；同时暴露 `glm4_moe_detector/ling3_detector` 尚未同步的技术债，跨解析器一致性在短期内仍不成立。
- 风险标记：仅覆盖 glm47 解析器，流式缓冲改变首包时延，递归 Schema 解析开销，与 #35625/#36688 重叠

关联脉络

- PR #36626 材料未提供原始标题：在顶层 schema 组合子内发现 properties（PR body 称已合并）：本 PR 的直接前置：它让解析器能在顶层组合子内部发现属性，本 PR 在其基础上避免「选中第一个重复属性类型」，改用已完成取值与同级判别字段选择分支。
- PR #35625 [function_call] Make streaming tool-call parsing agree with non-streaming: 同一问题域的统一修复提案（open），PR body 明确说明其与本 PR 有重叠但未覆盖 nullable 强制、本地引用、enum/const 收窄与同级判别的根 union，后续合并时需协调。
- PR #36688 [Fix] GLM detectors: streamed tool-call arguments disagree with non-streaming: 针对 glm/glm45/glm47 的窄版本修复，与 #35625 重叠；本 PR 只改 `glm47_moe_detector`，与之存在修复范围与实现路径的重叠风险。