

PR #38297 完整报告

sgl-project/sglang

Auto-detect GLM-5.3 chat templates as glm45/glm47 parsers

合并时间: 2026-09-11 21:02

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/38297>

执行摘要

- 一句话: 补齐 GLM-5.3 模板自动检测, 恢复 auto 模式工具调用解析
- 推荐动作: 值得精读, 尤其适合负责模型接入与 OpenAI 兼容 API 的工程师。建议重点关注三处设计取舍: (1) `_is_glm53` 里 `not ctx.has_text("enable_thinking")` 这个排除条件——它决定了新旧模板能否互斥, 也决定了未来模板一提到该字样就会静默退回错误路径; (2) `_is_glm_family` 复用 `glm45` 规则名而不新增规则, 这是把“解析器名稳定”当作对外契约来保护, 代价是谓词语义被塞进一个名字里; (3) `test_specific_rules_precede_broad_fallbacks` 里对规则表下标做断言的做法——当 `glm45` 与 `xml_kv_tool_call` 映射到同一个值 `glm45` 时, 值断言无法发现顺序错误, 这个技巧值得在同类检测逻辑中复用。阅读时不必纠结 diff 体量 (仅 2 文件 +77/-4), 但应意识到这类基于模板字面量的检测本质上是脆弱契约, 评审同类改动时值得要求补充真实 tokenizer/ 模板快照回归。

功能与动机

PR 正文给出的现场问题是: 以 `--reasoning-parser auto --tool-call-parser auto` 服务 GLM-5.3-Flash 时工具调用静默失效——200 次客户风格请求回放中 0/200 返回 `tool_calls`, 而同一批请求走裸 `/v1/completions` 时模型 100% 输出了规范的 `</think> + <tool_call>`, 显式传 `--reasoning-parser glm45 --tool-call-parser glm47` 又完全正常, 因此判定为“纯粹是模板检测缺口”。缺口分两层: 一是 `_is_glm45` 硬性要求 `reasoning_config == ReasoningToggleConfig(toggle_param="enable_thinking", default_enabled=True)`, 而 GLM-5.3 模板没有该开关, 整条 GLM 规则链漏判, 工具侧落到通用的 `xml_kv_tool_call` 兜底 (值为 `glm45`, 其 `func_detail_regex` 要求函数名后换行, 读不了紧凑的 `<tool_call>name<arg_key>`), 推理侧落到不带 `tool_start_token` 的 `deepseek-r1`; 二是 `_is_glm47` 硬编码 `'<tool_call>' + tc.name`, 而 2026-09-04 的模板修订 (HF 690b7052) 改写成 `'<tool_call>' ~ tc.name`, 即使开关可用也会被判成 `glm45`。

实现拆解

1. 新增 GLM-5.3 谓词: `python/sglang/srt/parser/template_detection.py` 中新增 `_is_glm53(ctx)`, 用模板文本同时满足 `[gMASK]<sop>`、`Reasoning Effort:`、`<tool_call>`、`<arg_key>`、`<arg_value>` 且显式排除 `enable_thinking` 作为签名, 从语义上与 `_is_glm45` 的开关式签名区分开, 避免两条规则互相污染。
2. 引入族谓词并复用旧规则名: 新增 `_is_glm_family(ctx) = _is_glm45(ctx) or _is_glm53(ctx)`, 随后把 `REASONING_PARSER_RULES` 与

TOOL_CALL_PARSER_RULES 中 name 为 glm45 的两条 DetectionRule 谓词由 `_is_glm45` 换成 `_is_glm_family`。规则名称、value 与在表中的相对位置都不变，因此 GLM-4.5/4.6/4.7 命中的规则和最终解析器名完全一致，把回归面压到最小。

- 放宽 glm47 正则：`_is_glm47` 的前置条件由 `_is_glm45(ctx)` 改为 `_is_glm_family(ctx)`，正则从 `\{[-\s]*['"]<tool_call>['"]\s*\+\s*tc\.name` 改成 `\s*[+~]\s*`，同时接受 Jinja 的 `+` 与 `~` 拼接写法。由于工具规则表中 glm47 本就排在 glm45 与 `xml_kv_tool_call` 之前，GLM-5.3 现在会先命中 glm47，拿到能解析紧凑格式的工具解析器。
- 不改推理模式检测：`detect_reasoning_pattern / ReasoningToggleConfig` 路径未动，`auto` 模式产出的解析器名现在与显式 flag 完全一致（正文表述为“`auto mode now yields exactly the parser names the explicit flags set`”），`thinking` 常开语义仍沿用 glm45 检测器的默认值。
- 测试配套：`test/registered/unit/parser/test_template_manager.py`（`register_cpu_ci(est_time=2.0, suite="base-a-test-cpu")`）新增 `_glm53_template(concat)` 工厂按 `+/~` 参数化造模板；新增 `test_glm53_effort_template_resolves_glm_parsers` 同时断言推理侧为 glm45、工具侧为 glm47；在 `test_tool_call_parser_rule_values_via_snippets` 的用例表中补 `glm47_tilde_concat_tool_call`；在 `test_specific_rules_precede_broad_fallbacks` 中补 glm45 早于 `deepseek_r1_think_tags` 的位置断言。作者说明三处用例在 main 上均失败。无文档、配置、schema 或部署改动。

关键文件：

- `python/sglang/srt/parser/template_detection.py`（模块 模板检测；类别 source；类型 core-logic；符号 `_is_glm53`, `_is_glm_family`, `_is_glm47`, `_is_glm45`）：本 PR 的唯一源码改动点：新增 `_is_glm53` 谓词与 `_is_glm_family` 并集，替换 REASONING_PARSER_RULES / TOOL_CALL_PARSER_RULES 中两条 glm45 规则的谓词，并把 `_is_glm47` 的拼接正则放宽为 `[+~]`。整个修复的行为边界都由这几个函数决定。
- `test/registered/unit/parser/test_template_manager.py`（模块 模板单测；类别 test；类型 test-coverage；符号 `_glm53_template`, `test_glm53_effort_template_resolves_glm_parsers`, `test_specific_rules_precede_broad_fallbacks`, `test_tool_call_parser_rule_values_via_snippets`）：全部新增覆盖都在这里：
`_glm53_template` 工厂按 `+/~` 参数化复现 GLM-5.3 模板形态，
`test_glm53_effort_template_resolves_glm_parsers` 同时断言推理侧 glm45 与工具侧 glm47，用例表新增 `glm47_tilde_concat_tool_call`，顺序断言补上 glm45 早于 `<think>` 兜底。作者说明这些用例在 main 上均失败。

关键符号：`_is_glm53`, `_is_glm_family`, `_is_glm47`, `_is_glm45`, `_glm53_template`,
`test_glm53_effort_template_resolves_glm_parsers`,
`test_specific_rules_precede_broad_fallbacks`

关键源码片段

`python/sglang/srt/parser/template_detection.py`

本 PR 的唯一源码改动点：新增 `_is_glm53` 谓词与 `_is_glm_family` 并集，替换 REASONING_PARSER_RULES / TOOL_CALL_PARSER_RULES 中两条 `glm45` 规则的谓词，并把 `_is_glm47` 的拼接正则放宽为 `[+~]`。整个修复的行为边界都由这几个函数决定。

python/sglang/srt/parser/template_detection.py —— GLM 家族谓词（本次重写后的形态）

```
def _is_glm45(ctx):
    # GLM-4.5 / 4.6 的原始签名：模板带 [gMASK]<sop> 或 /think、/nothink,
    # 且推理配置恰好是“默认开启的 enable_thinking 开关”。
    # 本函数在 PR 中未被修改，是 GLM-5.3 漏判的“参照物”。
    return (
        (
            ctx.has_text("[gMASK]<sop>")
            or ctx.has_pattern(r"(?!<)/nothink")
            or ctx.has_pattern(r"(?!<)/think")
        )
        and ctx.has_vocab("<tool_call>")
        and ctx.reasoning_config
        == ReasoningToggleConfig(toggle_param="enable_thinking", default_enabled=True)
        and (ctx.has_vocab("<luserl>") or ctx.has_vocab("<lendoftextl>"))
    )

def _is_glm53(ctx):
    # 新增：GLM-5.3 保留 GLM-4.5 的 prompt 与工具调用格式，
    # 但把 enable_thinking 开关换成了常开的 "Reasoning Effort:" 头。
    # 显式排除 enable_thinking 是为了与 _is_glm45 互斥，避免两条规则语义重叠；
    # 注意这里三个工具标记用的是模板文本（has_text），而 _is_glm45 用的是词表（has_vocab）。
    return (
        ctx.has_text("[gMASK]<sop>")
        and ctx.has_text("Reasoning Effort:")
        and not ctx.has_text("enable_thinking")
        and ctx.has_text("<tool_call>")
        and ctx.has_text("<arg_key>")
        and ctx.has_text("<arg_value>")
    )

def _is_glm_family(ctx):
    # 关键抽象：对外仍然沿用 glm45 这个规则名与解析器名，只把谓词扩成并集。
    # 规则表的名称与相对顺序都不动，因此 GLM-4.5/4.6/4.7 的解析结果完全不变，
    # 把本次修复的回归面压到最小。
    return _is_glm45(ctx) or _is_glm53(ctx)

def _is_glm47(ctx):
    # GLM-4.7 的紧凑工具调用格式：'<tool_call>' 与函数名直接拼接。
    # HF 2026-09-04 模板修订把拼接运算符由 + 改成 ~，两种写法都必须接受，
    # 否则即便开关判断通过，GLM-5.3 也会退回 glm45 工具解析器。
```

```

return _is_glm_family(ctx) and ctx.has_pattern(
    r"\{[-\s]*['\"]<tool_call>['\"]\s*[+~]\s*tc\.name"
)

```

规则挂载点（节选，省略无关规则）：谓词替换后 GLM-5.3 命中 glm45 + glm47

```

REASONING_PARSER_RULES = (
    # ...
    # 由 _is_glm45 换成 _is_glm_family；必须早于 deepseek_r1_think_tags 这条
    # “<think> 兜底”，否则 GLM-5.3 会再次被判成 deepseek-r1 并丢掉 tool_start_token。
    DetectionRule(name="glm45", value="glm45", predicate=_is_glm_family),
    # ...
)

```

```

TOOL_CALL_PARSER_RULES = (
    # ...
    # glm47 必须先于 glm45 命中，才能拿到能解析紧凑 <tool_call>name<arg_key> 的解析器；
    # glm45 则必须先于 xml_kv_tool_call 兜底，后者要求函数名后换行、读不了紧凑格式。
    DetectionRule(name="glm47", value="glm47", predicate=_is_glm47),
    DetectionRule(name="glm45", value="glm45", predicate=_is_glm_family),
    # ...
    DetectionRule(name="xml_kv_tool_call", value="glm45", predicate=_is_xml_kv_tool_call),
)

```

test/registered/unit/parser/test_template_manager.py

全部新增覆盖都在这里：_glm53_template 工厂按 +/~ 参数化复现 GLM-5.3 模板形态，test_glm53_effort_template_resolves_glm_parsers 同时断言推理侧 glm45 与工具侧 glm47，用例表新增 glm47_tilde_concat_tool_call，顺序断言补上 glm45 早于 <think> 兜底。作者说明这些用例在 main 上均失败。

test/registered/unit/parser/test_template_manager.py

```

def _glm53_template(concat):
    """GLM-5.3 模板形态：thinking 常开、由 ``Reasoning Effort:`` 头驱动，
    没有 ``enable_thinking`` 开关；工具调用沿用 GLM-4.7 的紧凑格式，
    HF 的不同修订只差 ``+`` / ``~`` 拼接运算符。"""
    return (
        "[gMASK]<sop>\n"
        "{%- set effective_reasoning_effort = reasoning_effort if reasoning_effort is defined "
        "and reasoning_effort in ['low', 'high'] else 'max' -%}\n"
        "<|system|>Reasoning Effort: {{ effective_reasoning_effort | capitalize }}\n"
        "{% for tc in m.tool_calls %}\n"
        f"{{{%- ' <tool_call>' {concat} tc.name -}}}\n"
        "{% set _args = tc.arguments %}"
        "{% for k, v in _args.items() %}"
        "<arg_key>{{ k }}</arg_key><arg_value>{{ v }}</arg_value>"
        "{% endfor %}</tool_call>\n"
        "{% endfor %}\n"
        "<|assistant|>{{{%- ' <think>' -}}}"
    )

```

)

```
class TestTemplateManagerReasoningDetection(unittest.TestCase):
    def test_glm53_effort_template_resolves_glm_parsers(self):
        # 修复前：模板没有 enable_thinking 开关，_is_glm45 整条规则链漏判，
        # 推理侧落到 deepseek-r1（无 tool_start_token），
        # 工具侧落到 xml_kv 兜底（值也是 glm45），
        # 而 glm45 的 func_detail_regex 要求函数名后换行，
        # 读不了紧凑的 <tool_call>name<arg_key>，于是工具调用全部丢失。
        vocab = ["<tool_call>", "<arg_key>", "<arg_value>", "<luserl>", "<lendoftextl>"]
        for concat in ("+", "~"):
            # 两个 HF 修订都要覆盖：+ 是旧写法，~ 是 2026-09-04 修订后的写法。
            template = _glm53_template(concat)
            with self.subTest(concat=concat):
                force, config, parser = self._detect(template, vocab)
                self.assertEqual(parser, "glm45")
                self.assertEqual(
                    detect_tool_call_parser(
                        template, _DummyTokenizer(vocab), config, force
                    ),
                    "glm47",
                )

    def test_specific_rules_precede_broad_fallbacks(self):
        # 值断言无法发现顺序错误：glm45 与 xml_kv_tool_call 映射到同一个值 "glm45",
        # 所以这里直接对规则表下标断言，把“具体规则先于兜底”的契约钉住。
        reasoning_index = {
            rule.name: i for i, rule in enumerate(REASONING_PARSER_RULES)
        }
        self.assertLess(
            reasoning_index["glm45"], reasoning_index["deepseek_r1_think_tags"]
        )
```

评论区精华

Review 侧信息量极低：只有 b8zhong 一条 APPROVED，body 为空，
`review_comments_count = 0`，3 个 commit 中 2 个是把 main 合并进分支的 merge commit，属于典型的“作者自证 + 维护者直接放行”流程。真正有分量的技术论证全部写在 PR 正文里：用 devbox e2e 对照表证明 auto 与显式 parser 的行为已经等价（46% OK vs 47% OK，`tool_calls` 均 100%，completion p50 181 ms vs 188 ms），并按 46% 之外的 SCHEMA_FAIL 归因为模型在 `reasoning_effort=high` 下漏必填 key、与检测无关。作者还主动划定边界，声明本 PR 独立于 #37977（effort-name aliasing）与模型支持 PR #36507。由于缺少 reviewer 的独立复核，规则表的顺序敏感性与 `has_text` / `has_vocab` 语义差异没有被第三方审过。

- GLM-5.3 复用 glm45 规则名而非新增规则 (design)：采用族谓词复用旧规则名，避免改动规则表顺序触发其他模型回归；代价是 glm45 这个名字同时承载两种模板语义。

- 工具调用拼接运算符 + 与 ~ 的兼容判据 (correctness): 两种写法都被接受, 测试同时覆盖 + 与 ~; ~ 与 + 的类型语义差异未在检测层处理, 但当前只影响已发布的两种修订。
- auto 与显式 parser 的等价性验证 (testing): 认定修复后 auto 与显式行为等价, SCHEMA_FAIL 属模型侧问题; 但该结论只有作者单机 100 请求的样本, 缺少第三方复核与 CI 侧端到端覆盖。

风险与影响

• 风险:

1. 字面量匹配脆弱且会静默回退: `_is_glm53` 完全依赖模板文本中出现 Reasoning Effort: 且 不出现 `enable_thinking`。若后续 HF 修订在注释、kwargs 透传分支或文档片段里带上 `enable_thinking` 字样, 谓词立刻为假, 模板会再次静默落到 `deepseek-r1 + xml_kv_tool_call` 兜底, 症状与本次修复前一致 (工具调用全部丢失), 且启动日志不会报错——这正是本次故障“静默”特征的延续。
2. `has_text` 与 `has_vocab` 语义混用: `_is_glm53` 用 `ctx.has_text("<tool_call>") / has_text("<arg_key>") / has_text("<arg_value>")` 检查模板文本, 而 `_is_glm45` 与 `_is_xml_kv_tool_call` 用 `ctx.has_vocab(...)` 检查 tokenizer 词表。两者不是同一来源, 若某个 tokenizer 未把这几个串注册为 added token, 可能出现推理侧已判为 GLM 家族、工具侧兜底路径却不成立的组合, 行为不一致且难以从日志定位。
3. 正则放宽到 `[+~]` 的语义盲区: ~ 与 + 在 Jinja 中语义不同 (字符串拼接 vs 加法 / 拼接), 检测层只按模式匹配, 无法感知拼接对象类型; 目前只影响已发布的两种 HF 修订, 但把这条正则当作“所有 GLM-4.7 形态”的判据会偏乐观。
4. 残留功能缺口 (本 PR 范围外): 正文自述 `reasoning_effort: "none"` 与 `chat_template_kwargs: {enable_thinking: false}` 都无法真正关闭 thinking, CoT 仍会出现在 content 而不进入 reasoning 字段, 用户按文档“关思考”会看到反常内容, 作者明确把这块划给 #37977。
5. 验证强度不均衡: 单测只用模板字符串 + `_DummyTokenizer`, 没有真实 GLM-5.3 tokenizer 的快照回归; 真实模型侧只有作者单机 (4xB200, 100 请求、并发 8) 验证, 且 SCHEMA_FAIL 仍高达 54%, 说明本次只修好了“工具调用完全不出现”, 不解决模型漏参数。
6. CI 状态异常: PR 正文的 CI 状态块显示 Base 通过, 但 Extra (Run #34520573214) 与 AMD ROCm 10 (Run #34520573614) 为失败, 配合 `bypass-fastfail` 标签与本 PR 的纯 CPU 单测属性, 需要确认这些失败是否与本改动无关。
7. 回归面控制较好: `_is_glm45` 本体未改, 仅新增 `_is_glm_family` 并替换规则谓词, 规则名与顺序保持原样, 其他模型 (Qwen3、DeepSeek、Hunyuan 等) 不受影响, 且解析器选择发生在启动阶段, 运行期无额外开销。- 影响: 用户侧: GLM-5.3-Flash 用户用 `--reasoning-parser auto --tool-call-parser auto` 即可恢复 function calling (正文数据: `tool_calls` 由 0/200 变为 100/100, 且 `TAG_LEAK/NO_TOOL_CALL/LENGTH` 全部为 0), 显式指定 parser 的用户行为不变; 推理侧解析器名从错误的 `deepseek-r1` 变为 `glm45`, 与显式 flag 对齐。系统侧: 改动只在 `template_detection.py` 的启动期启发式逻辑内, 不触碰 attention、KV cache、调度、CUDA graph 等运行期路径, 无性能影响。团队侧: 确立了“新模型改版时用 family 谓词复用旧规则名、并把顺序契约写成下标断言

测试”的接入范式，对后续 GLM 系列或其他同族模型模板演进有直接参考价值。影响程度为局部模块但对外可见症状严重，属于“小改动、高用户价值”的修复。 - 风险标记：启发式模板匹配脆弱，静默回退无告警，has_text/has_vocab 语义不一致，缺少真实 tokenizer 回归，enable_thinking 关闭语义残留，CI Extra/ROCm 运行失败未说明，无独立评审

关联脉络

- PR #35486 [Responses] Fix empty-prompt routing for token-first chat encoders (kimi_k3, inkling): 同属请求侧 chat 模板 / 编码路由的一致性修复（涉及 serving_chat.py、chat_encoding.py 的分支选择），与本 PR 一样是“模板渲染与解析器选择必须对齐”的故障模式，可作为同类问题的对照阅读。
- PR #39019 [Test] Fix Q8KV8 sparse-prefill pool fixture after page-size rename: 同为“上游改名 / 格式漂移击穿下游解析”的案例：本 PR 的直接触发点就是 HF 模板把 + 改成 ~，两者都说明外部格式修订会让基于字符串匹配的代码静默失配。
- PR #37977 GLM-5.3 reasoning effort 别名（PR 正文提及的独立 PR，标题为描述性转述）：PR 正文明确声明本 PR 与 #37977 (effort-name aliasing) 独立；reasoning_effort: "none" / chat_template_kwargs: {enable_thinking: false} 无法真正关闭 thinking 的残留问题被划归该 PR，是同一条 GLM-5.3 支持线的后续。
- PR #36507 GLM-5.3 模型支持 PR（PR 正文提及的独立 PR，标题为描述性转述）：正文写明“Independent of ... the model-support PR #36507”，即模型本体接入由该 PR 承载，本 PR 只补 template detection 缺口，两者共同构成 GLM-5.3-Flash 端到端可用的前提。