PR #28149 完整报告

sgl-project/sglang

Support GLM-4.7 function calling via structural tags

合并时间：2026-06-14 14:57

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/28149

执行摘要

- 一句话：为 GLM-4.7 启用结构标签指导的函数调用
- 推荐动作：值得精读。该 PR 展示了如何为特定模型接入 xgrammar 原生结构标签的完整流程，包括运行时探测、thinking 模式处理、以及优雅降级。其中的设计决策（例如通过 lru_cache 缓存探测结果、在 get_structural_tag 中调用父类方法避免重复实现）值得在其他模型检测器中借鉴。

功能与动机

GLM-4.7 MoE 函数调用检测器原先 supports_structural_tag() 硬编码为 False，导致无法使用结构化 / 语法引导的工具调用，只能依赖自由文本生成，可靠性较低。PR 正文明确指出："This forced these models to rely on free-form generation for function calls, which is less reliable."

实现拆解

实现分为以下步骤：

1. 运行时探测 xgrammar 可用性：在文件 glm47_moe_detector.py 顶部新增 _glm47_native_structural_tag_available() 函数，使用 lru_cache 缓存结果。通过调用 get_model_structural_tag(model="glm_4_7", ...) 并捕获异常来判断 xgrammar 是否注册了 glm_4_7 标签，避免在 xgrammar 版本过旧时直接返回 True 导致后续崩溃。
2. 启用结构标签支持：将 supports_structural_tag() 的返回值从 False 改为调用 _glm47_native_structural_tag_available()。新增 get_structural_tag() 方法，在支持时调用父类方法，不支持时返回 None。新增 get_structural_tag_name() 方法返回 "glm_4_7"，映射到 xgrammar 的原生结构标签。
3. 处理 thinking 模式：get_structural_tag() 接受 thinking_mode 参数，当 thinking_mode=True 时保留 </think> 推理前缀；当 False 时由 ReasonerGrammarBackend 负责该前缀，避免重复。
4. 更新导入依赖：新增对 functools.lru_cache、ToolChoice、StructuralTag、get_model_structural_tag 等的导入，确保新代码可用。
5. 新增单元测试：在 test_function_call_parser.py 中添加两个测试方法。test_get_model_structural_tag 验证结构标签在各种 thinking_mode 和 tool_choice 下能正确编译为 xgrammar 的 grammar 对象，并检查序列化输出包含 glm_xml、<tool_call>、

</think> 等关键标记。test_required_tool_choice_falls_back_when_native_tag_is_unavailable 模拟 get_model_structural_tag 不可用（通过 patch），验证 supports_structural_tag() 返回 False、get_structural_tag() 返回 None，且 FunctionCallParser 在 tool_choice="required" 时能正确回退到 JSON Schema 约束。

关键文件：

- python/sglang/srt/function_call/glm47_moe_detector.py（模块 函数调用；类别 source；类型 core-logic；符号 _glm47_native_structural_tag_available, supports_structural_tag, get_structural_tag, get_structural_tag_name）：核心逻辑文件，新增运行时探测函数、启用结构标签支持、实现 get_structural_tag 和 get_structural_tag_name 方法。
- test/registered/unit/function_call/test_function_call_parser.py（模块 测试；类别 test；类型 test-coverage；符号 test_get_model_structural_tag, test_required_tool_choice_falls_back_when_native_tag_is_unavailable）：新增两个测试方法，覆盖结构标签正常编译和 xgrammar 不可用时的回退路径，确保核心逻辑的正确性。

关键符号：_glm47_native_structural_tag_available, supports_structural_tag, get_structural_tag, get_structural_tag_name, test_get_model_structural_tag, test_required_tool_choice_falls_back_when_native_tag_is_unavailable

关键源码片段

python/sglang/srt/function_call/glm47_moe_detector.py

核心逻辑文件，新增运行时探测函数、启用结构标签支持、实现 get_structural_tag 和 get_structural_tag_name 方法。

```
# 运行时探测 xgrammar 是否支持 glm_4_7 结构标签
# 使用 lru_cache 确保每次进程只探测一次
@lru_cache(maxsize=1)
def _glm47_native_structural_tag_available() -> bool:
    # "glm_4_7" 仅在较新的 xgrammar 中注册，所以导入可能成功但模型名不被识别
    # 探测一次，如果缺失则自动降级
    if get_model_structural_tag is None:
        return False
    try:
        get_model_structural_tag(
            model="glm_4_7", tools=[], tool_choice="auto", reasoning=False
        )
        return True
    except Exception:
        return False

class Glm47MoeDetector(BaseFormatDetector):
    # ... 原有代码 ...

    def supports_structural_tag(self) -> bool:
        # 返回运行时探测结果，而非硬编码 False
```

```
        return _glm47_native_structural_tag_available()

    def get_structural_tag(
        self,
        tools: Union[List[Tool], None] = None,
        tool_choice: Union[ToolChoice, Literal["auto", "required"]] = "auto",
        thinking_mode: bool = False,
    ) -> Optional[StructuralTag]:
        if not self.supports_structural_tag():
            return None
        # 调用父类方法生成结构标签，父类会利用 get_structural_tag_name 获取模型名
        return super().get_structural_tag(
            tools=tools, tool_choice=tool_choice, thinking_mode=thinking_mode
        )

    def get_structural_tag_name(self) -> str:
        # 返回 xgrammar 中注册的 glm_4_7 标签名
        return "glm_4_7"
```

test/registered/unit/function_call/test_function_call_parser.py

新增两个测试方法，覆盖结构标签正常编译和 xgrammar 不可用时的回退路径，确保核心逻辑的正确性。

```
  def test_get_model_structural_tag(self):
    """GLM-4.7/GLM-5 使用 xgrammar 的原生 "glm_4_7" 结构标签。"""
    import xgrammar as xgr

    # 验证检测器支持结构标签且名称正确
    self.assertTrue(self.detector.supports_structural_tag())
    self.assertEqual(self.detector.get_structural_tag_name(), "glm_4_7")

    # thinking_mode=True 时保留 </think> 推理前缀
    structural_tag = self.detector.get_structural_tag(
        self.tools, thinking_mode=True
    )
    self.assertIsInstance(structural_tag, xgr.StructuralTag)
    serialized = structural_tag.model_dump_json()
    self.assertIn("glm_xml", serialized)
    self.assertIn("<tool_call>", serialized)
    self.assertIn("</think>", serialized)

    # thinking_mode=False 时去掉推理前缀（由 ReasonerGrammarBackend 处理）
    structural_tag = self.detector.get_structural_tag(
        self.tools, thinking_mode=False
    )
    self.assertIsInstance(structural_tag, xgr.StructuralTag)

    # tool_choice="required" 也必须能编译成 grammar
    structural_tag = self.detector.get_structural_tag(
```

```
            self.tools, thinking_mode=True, tool_choice="required"
        )
        self.assertIsInstance(structural_tag, xgr.StructuralTag)
        self.assertIsInstance(
            xgr.Grammar.from_structural_tag(structural_tag), xgr.Grammar
        )

    def test_required_tool_choice_falls_back_when_native_tag_is_unavailable(self):
        """当 xgrammar 不支持 glm_4_7 时，supports_structural_tag 返回 False，
        get_structural_tag 返回 None，tool_choice="required" 回退到 JSON Schema。"""
        from unittest.mock import patch
        from sglang.srt.function_call.function_call_parser import FunctionCallParser
        from sglang.srt.function_call.glm47_moe_detector import (
            _glm47_native_structural_tag_available,
        )

        # 模拟 get_model_structural_tag 不可用（例如 xgrammar 版本过旧）
        with patch(
            "sglang.srt.function_call.glm47_moe_detector.get_model_structural_tag",
            None,
        ):
            _glm47_native_structural_tag_available.cache_clear()
            self.assertFalse(self.detector.supports_structural_tag())
            self.assertIsNone(self.detector.get_structural_tag(self.tools))

            # 需要 tool_choice="required" 时，回退到 json_schema 约束
            parser = FunctionCallParser(self.tools, "glm47")
            constraint = parser.get_structure_constraint("required")
            self.assertIsNotNone(constraint)
            self.assertEqual("json_schema", constraint[0])
            _glm47_native_structural_tag_available.cache_clean() # 清理缓存避免影响其他测试
```

评论区精华

唯一的 review 评论来自 gemini-code-assist[bot]，指出如果 xgrammar 未安装或版本过旧，supports_structural_tag() 返回 True 会导致 get_structural_tag() 返回 None，进而使解析器调用 structure_info()，而该方法会抛出 NotImplementedError，导致约束失败。

解决：作者后来通过添加 _glm47_native_structural_tag_available() 运行时探测函数，在 supports_structural_tag() 中返回探测结果，确保在 xgrammar 不可用时返回 False，避免后续错误。该探测函数使用 lru_cache 缓存结果，每次进程只探测一次。

- xgrammar 不可用时可能导致 NotImplementedError (correctness): 作者通过添加 _glm47_native_structural_tag_available 运行时探测函数解决，该函数在调用 get_model_structural_tag 并捕获异常，返回布尔值供 supports_structural_tag 使用。同时使用 lru_cache 避免重复探测。

风险与影响

- 风险：
  1. xgrammar 版本兼容性：_glm47_native_structural_tag_available() 通过调用 get_model_structural_tag 并捕获异常来探测，但该探测可能因为其他原因（如网络问题）而失败，导致即使 xgrammar 支持也无法启用。不过由于探测在进程启动时执行一次，且异常捕获广泛，风险较低。
  2. glm_4_7 标签生命周期：若未来 xgrammar 删除或重命名该标签，探测会捕获异常并自动降级，不会崩溃。但需要保持同步更新。
  3. thinking_mode 处理一致性：thinking_mode=False 时，结构标签中不包含 </think> 前缀，但 ReasonerGrammarBackend 需要正确接管。若配置不当可能导致推理格式错误。
  4. 测试覆盖：两个测试方法覆盖了正常路径和回退路径，但未覆盖中间件集成测试，例如实际调用推理引擎验证生成的函数调用格式。 - 影响：影响范围：仅影响 GLM-4.7 MoE 模型（及潜在的 GLM-5 模型）的函数调用行为。用户无需手动迁移，原有自由文本调用方式不受影响。

影响程度：正面——为 GLM-4.7 用户提供更可靠的函数调用能力，约束生成能减少解析错误和格式异常。

系统影响：无性能或安全影响，变更仅涉及检测器的逻辑分支。

- 风险标记：依赖 xgrammar 版本，运行时探测无法覆盖所有异常场景

关联脉络

- 暂无明显关联 PR