LLM 等价于自然语言程序设计语言解释器的微分方程 FunctionCall 例程解析

- 作者：GaoZheng
- 日期：2025-07-06

一、例程任务：自然语言输入与目标建模

自然语言描述：

> 设 $y(t)$ 满足微分方程 $y'' + 4y' + 3y = 0$，初始条件为 $y(0) = 1$，$y'(0) = 0$，请输出其通解并求出特解。

该描述在 LLM 中等价于函数调用结构：

```
solve_ode("y'' + 4y' + 3y = 0", initial_conditions={"y(0)": 1, "y'(0)": 0})
```

二、语义路径转逻辑积分表达（GRL路径积分建模）

根据 GRL 路径积分理论：

- 自然语言输入被解析为一个语义路径 $\gamma = \{\text{ODE构造} \rightarrow \text{符号解析} \rightarrow \text{初值映射} \rightarrow \text{求解路径}\}$
- 每一阶段可映射为逻辑积分结构中的状态转移 $s_i \rightarrow s_{i+1}$，并伴随微分权重度量 $\mu(s_i, s_{i+1}; w)$
- 最终构成的推理路径选择 $\arg\max_{\gamma \in \Gamma} \sum_i \mu(s_i, s_{i+1}; w)$ 即为推理结果输出的最佳解路径

这体现了LLM中FunctionCall本质上是路径积分空间中的“最优路径搜索器”。

三、数学求解过程（LaTeX 结构化表达）

1. 给定微分方程：

$$\frac{d^2y}{dt^2} + 4\frac{dy}{dt} + 3y = 0$$

2. 初始条件：

$$y(0) = 1, \quad y'(0) = 0$$

3. 特征方程：

$$r^2 + 4r + 3 = 0 \Rightarrow r = -1, -3$$

4. 通解：

$$y(t) = C_1 e^{-t} + C_2 e^{-3t}$$

5. 应用初始条件：

$$\begin{cases} C_1 + C_2 = 1 \\ -C_1 - 3C_2 = 0 \end{cases} \Rightarrow C_1 = \frac{3}{2}, \quad C_2 = -\frac{1}{2}$$

6. 特解：

$$y(t) = \frac{3}{2}e^{-t} - \frac{1}{2}e^{-3t}$$

四、Python 脚本调用 WolframClient 实现完整求解

```python
from wolframclient.language import wl, wlexpr
from wolframclient.evaluation import WolframLanguageSession

# 启动本地 Wolfram 内核
session = WolframLanguageSession()

# 微分方程与初值条件
ode_expr = wlexpr("DSolve[{y''[t] + 4 y'[t] + 3 y[t] == 0, y[0] == 1, y'[0] == 0}, y[t], t]")

# 执行求解
result = session.evaluate(ode_expr)

# 输出结构化结果
print("微分方程求解结果:")
print(result)

# 关闭会话
session.terminate()
```

输出格式为 Wolfram 语言的符号解表达，可直接转为 LaTeX 显示为：

$$y(t) = \frac{3}{2}e^{-t} - \frac{1}{2}e^{-3t}$$

五、理论总结：LLM 解释器范式的程序—路径—结构统一性

- LLM 本质上等价于“自然语言驱动的解释器系统”
- 微分建模函数调用 = 路径空间上的最优逻辑积分
- Python函数调用与自然语言输入完全可互译
- GRL路径积分提供了推理路径、数学结构与程序操作的三重统一范式

LLM的这种 FunctionCall 机制，不仅仅是API调用代理，更是“语言即代码，语言即推理路径”的认知范式重构。

许可声明 (License)