**系统技术文档**

**1. 系统概述**

含有隐含关系的数学问题求解系统，采用模块化设计，通过多个独立组件的协同工作完成分析和求解。系统采用面向对象设计模式，并配备日志追踪系统。

**2. 系统架构**

**2.1 整体流程**

1. **math\_problem\_solver.py** 接收问题文本。
2. **nlp\_processor.py** 对输入文本进行自然语言处理。
3. **relation\_extractor.py** 和 **relation\_matcher.py** 提取数学关系。
4. **relation\_builder.py** 构建方程组。
5. **solution\_generator.py** 求解方程组。
6. **inference\_tracker.py** 记录整个解题过程。
7. **math\_problem\_solver.py** 是一个主函数输出格式化结果。

**2.2 组件职责**

| **组件名称** | **主要职责** | **输入/输出** |
| --- | --- | --- |
| **math\_problem\_solver.py** | 系统主程序，协调各组件工作 | 输入：问题文本 输出：拟人化解答 |
| **nlp\_processor.py** | 自然语言处理器，处理输入问题文本 | 输入：原始文本 输出：ProcessedText对象 |
| **relation\_extractor.py** | 从文本中提取数学关系 | 输入：处理后的文本 输出：数学关系列表 |
| **relation\_matcher.py** | 匹配数学关系模式 | 输入：处理后的文本 输出：匹配模式结果 |
| **relation\_builder.py** | 根据提取的关系构建方程组 | 输入：数学关系 输出：标准化方程组 |
| **solution\_generator.py** | 求解方程组并生成解答 | 输入：方程组 输出：解答结果 |
| **inference\_tracker.py** | 记录解题的推理过程 | 输入：各阶段处理数据 输出：完整推理链 |

**3. 详细设计**

**3.1 nlp\_processor.py**

自然语言处理器负责对输入文本进行初步处理，具体功能包括：

**文本分词**：将输入文本分解为词语。**词性标注**：标注每个词的词性。

**依存句法分析**：分析句子中的依存关系。**语义角色标注**：识别句子中的语义角色。

输出结果为 ProcessedText 对象，包含处理后的文本结构。

**3.2 relation\_extractor.py 和 relation\_matcher.py**

这两个模块协同完成数学关系的提取和模式匹配工作：

**1.relation\_extractor.py**：

提取显式关系（如直接数值关系）；**提取隐式关系（如变化率、时间关系）**；

模式匹配和关系验证。

**2.relation\_matcher.py**：定义数学关系模式。匹配数值模式、单位模式和运算符模式。

**3.3 relation\_builder.py**

关系构建器负责将提取的数学关系转换为标准化的方程组，具体功能包括：

**变量提取**：从数学关系中提取变量。**约束条件添加**：**识别并添加约束条件。**

**方程组构建**：根据数学关系生成方程组。**方程标准化**：将方程组转换为标准格式，方便求解。

**3.4 solution\_generator.py**

解题生成器负责求解方程组并生成最终解答，具体功能包括：

**方程组求解**：使用数值方法或符号方法求解方程。**单位换算**：对结果进行必要的单位换算。

**3.5 inference\_tracker.py**

推理跟踪器记录解题过程中的每一步，具体功能包括：

记录关系提取过程。记录方程构建过程。求解步骤：生成完整的推理链，便于调试和分析。

**4. 日志系统**

系统使用 logging.yaml 配置文件来管理日志，主要配置包括：

**日志级别**：DEBUG、INFO、WARNING、ERROR。

**输出格式**：标准化的日志格式。

**文件处理器**：将日志保存到文件。

**控制台处理器**：将日志输出到控制台。

**示例配置**

yaml

version: 1

formatters:

standard:

format: '%(asctime)s - %(name)s - %(levelname)s - %(message)s'

handlers:

console:

class: logging.StreamHandler

level: INFO

formatter: standard

file:

class: logging.FileHandler

level: DEBUG

formatter: standard

filename: math\_solver.log

root:

level: DEBUG

handlers: [console, file]

以下是系统的使用示例代码：

from math\_problem\_solver import MathProblemSolver

# 初始化求解器

solver = MathProblemSolver()

# 输入数学问题

problem\_text = "小明比小红大2岁，他们的年龄之和是20岁，求各自的年龄。"

# 求解问题

solution = solver.solve(problem\_text)

# 输出结果

print(solution.formatted\_output())