2025 年秋季 计算机科学使用的数理逻辑

江政杰 （25031212046）

2025-11-26

西安电子科技大学 计算机科学与技术学院

目录

Z3 Prover 简介

什么是 Z3

图 1　Z3 Prover

- 由微软研究院开发的 SMT（可满足性模理论）求解器
- 判断逻辑公式是否可满足
- 支持多种理论：整数/实数算术、数组、位向量、量词等
- 在程序验证、自动测试、安全分析等领域广泛使用
- 提供 Python、C++、Java、SMT-LIB 等多种接口

Z3 背后的数学基础

- **逻辑基础**
  - 命题逻辑：布尔变量、合取范式（CNF）、可满足性
  - 一阶谓词逻辑：项、公式、结构、模型（model）
- **理论（Theories）**
  - 线性整数算术 LIA、线性实数算术 LRA
  - 数组理论、位向量理论等
  - 都是一阶逻辑在特定结构上的 子理论
- **SMT 问题形式化**
  - 给定背景理论 T 与公式 φ
  - 问题：φ 在某个 T-模型中是否可满足?
  - 记为：SAT_T(φ)

- **DPLL(T) 框架的数学思想**
  - 布尔抽象：把复杂约束看成布尔原子
  - SAT 求解器：在布尔层面搜索可满足赋值
  - 理论求解器：检查这些原子是否在理论 T 中 一致
- **复杂性与可行性**
  - SAT 是 NP 完全，SMT 一般更难
  - 通过拆分为“布尔 + 理论”两层，提高求解效率

Z3 的编译与安装

Z3 的编译与安装

1. **获取源码 / 安装包**
   - 官方网站 / GitHub：下载源码或预编译二进制
   - 常用平台：Linux、macOS、Windows
2. **依赖与工具**
   - 需要：C++ 编译器、Python、CMake、Ninja（可选）
3. **编译步骤**
   - 解压 / clone 仓库
   - 配置：`cmake -G "Ninja" -B build -S .`
   - 进入 `build` 目录，执行 `ninja`
   - 生成 z3 可执行文件和库文件
4. **安装与环境变量**
   - `sudo make install`（或手动拷贝可执行文件与库）
   - 将 z3 所在路径加入 `PATH`
   - 如需 Python 接口，安装 `z3-solver` 包或开启 Python 绑定
5. **测试安装**
   - 终端运行：`z3 -version`
   - 在 Python 中 `import z3`，测试简单约束求解

Z3 Python Binding 使用

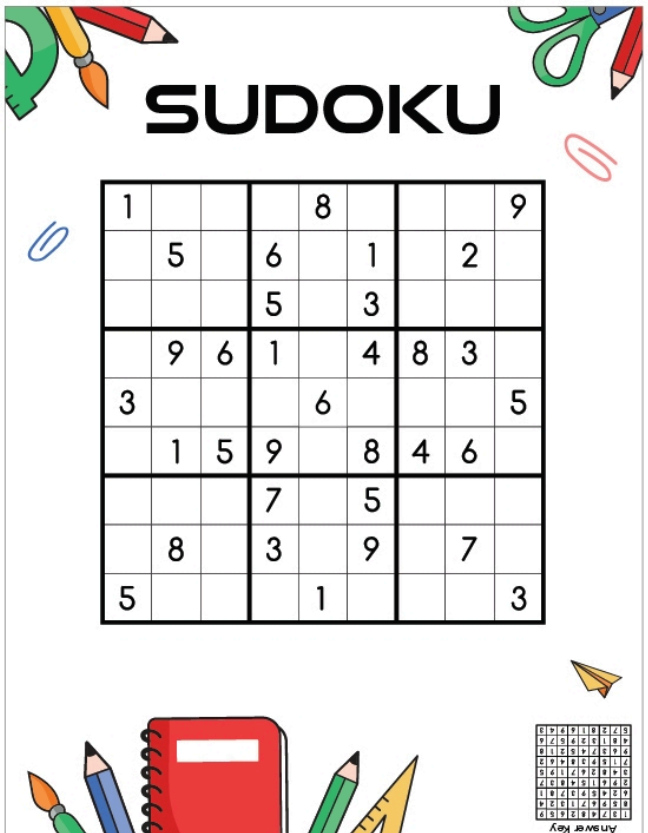

图 2　Sudoku Puzzle

- **基本规则**
  - 经典数独：9×9 方格，分成 9 个 3×3 宫
  - 在每一行、每一列、每个 3×3 宫中
  - → 数字 1–9 恰好各出现一次
- **给定条件（谜题）**
  - 部分格子预先填好数字
  - 目标：在不违背规则的前提下填满所有空格
  - 通常要求：解唯一
- **问题特点**
  - 对人类来说是益智游戏
  - 对计算机来说是一个 约束满足问题（CSP）
  - 可以用 SAT / SMT 求解器（如 Z3）自动求解或生成题目

在 Z3 中建模数独问题

1. **决策变量（Variables）**
   - 用 `Int` 变量表示每个格子：`X[r][c]`，共 9×9 个
   - 数学上：$X_{r,c} \in \{1, ..., 9\}$
2. **基本约束（Domain & Structure）**
   - 每个格子：`1 ≤ X[r][c] ≤ 9`
   - 每一行：`Distinct(X[r])`
   - 每一列：`Distinct([X[r][c] for r in range(9)])`
   - 每个 3×3 宫：对 `block` 调用 `Distinct(block)`
3. **题目线索（Clues）**
   - 对于 puzzle 中非 0 的格子：
     - 加约束 `X[r][c] == puzzle[r][c]`
   - 保证求解结果满足原题给出的数字
4. **求解与模型提取**
   - `s.check() == sat`：存在解
   - 用 `m = s.model()` 读取每个 `X[r][c]` 的值
   - 组装成 9×9 解答并打印数独棋盘

大模型结合 Z3 求解逻辑问题

LLM Agent 简介

- **LLM Agent 是什么?**
  - 由大语言模型驱动的“智能代理”
  - 不只回答问题，还能 **规划步骤、调用工具、读写代码**
- **典型工作流程**
  1. **理解题目**：从自然语言逻辑题中抽取对象、关系、约束
  2. **自动建模**：生成 Z3 代码或 SMT 约束（变量、域、规则）
  3. **调用 Z3**：运行求解器，得到 SAT / UNSAT 及模型
  4. **解释结果**：把模型翻译回自然语言答案和推理过程
- **为什么要结合 Z3?**
  - LLM：擅长语言理解与策略规划
  - Z3：擅长精确的逻辑推理与约束求解 → 结合后既能“听懂题目”，又能“严谨算对”
- **Agent 的特点**
  - 可反思：发现矛盾时重新建模或修改约束
  - 可复用：同一框架可以处理数独、逻辑谜题、时间表安排等
  - 更接近“自动定理证明助手”的工作方式

Einstein's Riddle

- **场景设定**
  - 一排 **5 间房子**，从左到右排列
  - 每间房子有：房子颜色、居住国籍、饮料、香烟品牌、宠物
  - 同一属性在 5 间房中都 **互不相同**
- **线索形式**（举例）
  - “英国人住在红房子”
  - “西班牙人养狗”
  - “咖啡在绿色房子里喝”
  - “绿色房子在象牙色房子右边”
  - “挪威人住在第一间，并且挨着蓝房子”
  - “某人抽 Old Gold 且养蜗牛”“日本人抽 Parliaments”等
- **任务**
  - 只根据这些线索，推理出
    - 每一间房子的 **颜色 / 国籍 / 饮料 / 香烟 / 宠物**
  - 经典问题版本：
    - “谁养某种宠物?”、“谁喝某种饮料?”
- **特点**
  - 典型的 **约束满足 / 逻辑推理谜题**
  - 非常适合用表格、程序，或 Z3 等求解器来建模与求解
  - 常被称为“据说只有 2% 的人能完全手算出来”

Einstein's Riddle Example

1. There are five houses.
2. The Englishman lives in the red house.
3. The Spaniard owns the dog.
4. Coffee is drunk in the green house.
5. The Ukrainian drinks tea.
6. The green house is immediately to the right of the ivory house.
7. The Old Gold smoker owns snails.
8. Kools are smoked in the yellow house.
9. Milk is drunk in the middle house.
10. The Norwegian lives in the first house.
11. The man who smokes Chesterfields lives in the house next to the man with the fox.
12. Kools are smoked in the house next to the house where the horse is kept.
13. The Lucky Strike smoker drinks orange juice.
14. The Japanese smokes Parliaments.
15. The Norwegian lives next to the blue house.

Now, who drinks water? Who owns the zebra?

Einstein's Riddle Solution

House	1	2	3	4	5
Color	Yellow	Blue	Red	Ivory	Green
Nationality	Norwegian	Ukrainian	Englishman	Spaniard	Japanese
Drink	**Water**	Tea	Milk	Orange juice	Coffee
Smoke	Kools	Chesterfield	Old Gold	Lucky Strike	Parliament
Pet	Fox	Horse	Snails	Dog	**Zebra**

表 1　Solution