Z3简介及其应用

计算机科学使用的数理逻辑-课程报告

小组成员：郝仕达 肖宇 张子豪

目录

CONTENTS

一，Z3简介

Z3是一个由MicrosoftResearch开发的高性能定理证明器，常用于软件验证和程序分析，Z3通过检查约束的可满足性来解决问题，SMT算法（SMT：可满足性模理论求解器）是其核心。它能够检查逻辑公式在给定理论组合下的可满足性，确定是否存在一组变量赋值使公式为真，并在存在时提供具体解。

Z3支持多种语言的API，如C、C++、Python等，并可用于软件/硬件验证和测试、约束求解、混合系统分析，安全性研究，生物学研究（计算机分析）以及几何问题等领域。

Z3发展历程：

初步开发

2008 年：由微软研究院的 Nikolaj Bjørner 和 Leonardo de Moura 领导的团队开始开发，隶属于 "软件工程研究"(RiSE) 小组

公开&开源

2012 年：首次公开发布。2015 年：正式开源，发布首个稳定版本，迅速成为 SMT 领域的事实标准

杰出贡献

2018 年：获得欧洲软件理论与实践联合会议 (ETAPS) 颁发"时间检验奖"，肯定其长期影响力。2019 年：Bjørner 和 de Moura 因 Z3 的工作获得Herbrand 自动推理杰出贡献奖

当前发展

持续更新，活跃于 GitHub 上的开源社区，在相关领域获得广泛应用。

二，Z3实现原理

Z3 采用CDCL(T)(Conflict-Driven Clause Learning with Theory) 架构，这是现代 SMT 求解的标准方法，结合了命题逻辑求解 (CDCL) 与理论特定推理 (T)。

总体架构：

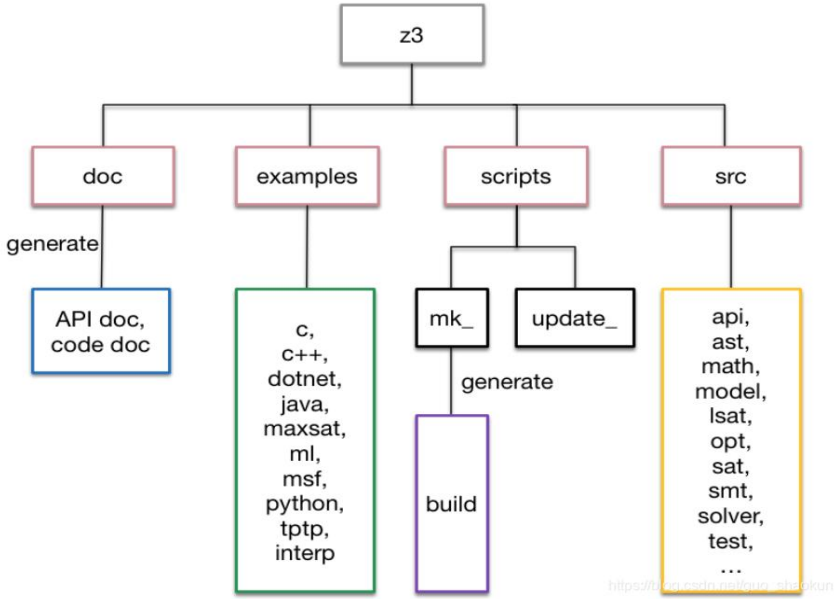
Z3 的核心由两部分组成：

•SAT 核心：处理命题逻辑部分，使用冲突驱动子句学习 (CDCL) 算法进行真值赋值搜索，类似于现代 SAT 求解器

•理论求解器：针对不同背景理论 (如算术、位向量、字符串等) 的专用推理引擎，负责检查命题赋值在特定理论下的一致性

两者通过 "黑板"(Blackboard) 机制紧密协作：SAT 核心生成候选赋值，理论求解器验证并提供冲突信息，共同引导搜索方向。

Z3项目库有以下四个子目录:
1）z3/doc，用于生成API doc文件和代码doc文件。
2）z3/examples，它提供了一组示例或基准测试，帮助用户理解如何用不同的API语言表示问题。
3）z3/scripts，包含用了于更新外部z3 API的文件
4）z3/src，它包含了z3的所有核心模块。



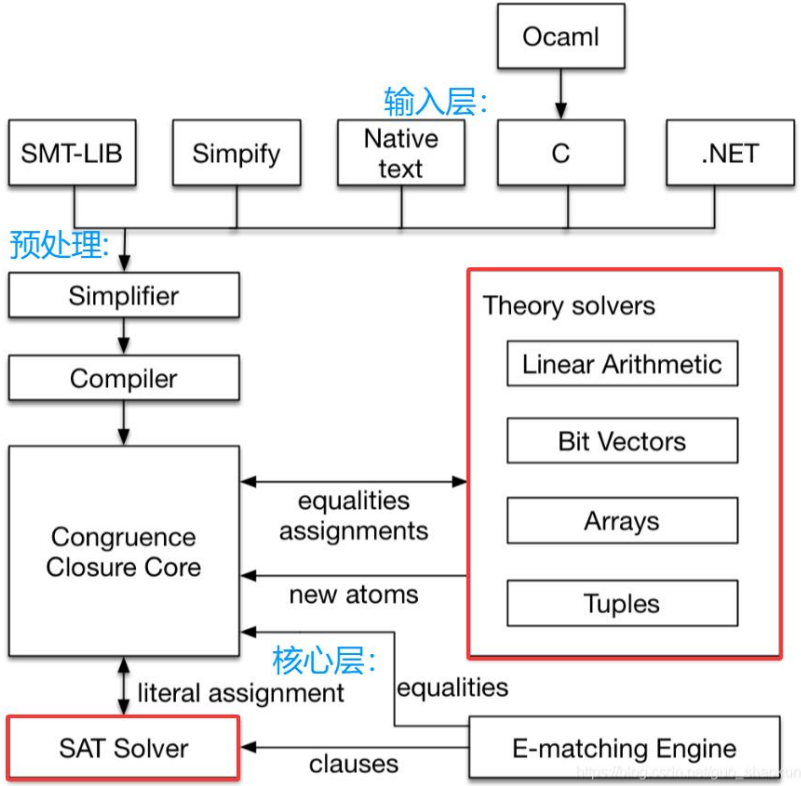
二，Z3实现原理

1.输入层：
Z3 支持多种输入方式 / 接口（SMT-LIB 标准语言、Simplify 语言、原生文本、Ocaml 编译后的 C 代码、.NET 框架接口），覆盖不同场景的使用需求。

2.预处理层：
各类输入先进入 “简化器（Simplifier）” 做初步约束简化，再通过 “编译器（Compiler）” 处理后，传递到 “同余闭包核心（Congruence Closure Core）” 。

3.核心协作层：
同余闭包核心与右侧的 “**理论求解器**” 交互（传递等式、赋值、新原子等信息），理论求解器负责处理特定领域（线性算术、位向量等）的约束验证；
“E 匹配引擎” 将等式转换为子句传递给 “**SAT 求解器**”，SAT 求解器则把文字赋值反馈给同余闭包核心，形成 “命题推理 + 理论推理” 的协作循环，最终完成约束的可满足性判定。

Z3基本架构：



二，Z3实现原理

Z3的实现原理建立在以下三个支柱之上：

1. 可满足性模理论 - Satisfiability Modulo Theories (SMT)

1.1.SAT (布尔可满足性问题)：

这是计算复杂性理论中的基础问题，关注于判定一个布尔逻辑公式是否存在使其为真的变量赋值。现代SAT求解器采用如CDCL 这样的高效算法来应对该NP完全问题。

1.2.为了处理超越纯布尔逻辑的现实问题，Z3引入了多种具有特定语义的背景理论。这些理论定义了特定领域的对象、运算和公理，例如：算术理论;位向量理论;数组理论;未解释函数理论;SMT = SAT + 理论.

Z3的工作原理是将高级约束抽象化并集成到一个统一的框架中，由一个核心的SAT引擎协同多个理论求解器共同工作。

2. 决策过程

对于其支持的每一种理论，Z3都内置了一个专门的决策过程。该过程负责判定一组在该理论下的约束是否可满足，能够基于理论自身的公理和规则高效地推理出约束的一致性或不一致性。

3. 求解引擎的工作流程

Z3内部遵循一个高度协同的工作流程：

1.输入与解析：系统接收并解析用特定语言编写的约束公式。

2.抽象化与布尔化：将复杂公式转换为一个等价的、由布尔变量和理论原子公式组成的组合结构。

3.CDCL SAT求解器核心驱动： SAT求解器作为核心引擎，开始探索布尔变量的赋值空间。

4.理论一致性检查：SAT求解器的临时赋值会触发对应的理论求解器进行一致性检查。理论求解器评估当前累积的理论约束集。如果发现冲突，理论求解器会向SAT求解器返回一个冲突子句。SAT求解器通过子句学习机制吸收该冲突信息，并执行回溯以调整搜索路径。

5.结果输出：SAT（可满足）： 找到满足所有约束的赋值，并输出模型。UNSAT（不可满足）： 穷尽搜索后确认无解。UNKNOWN（未知）： 在资源限制内无法判定。

二，Z3实现原理

SAT 核心：CDCL 求解器高性能实现

Z3 的 SAT 核心是冲突驱动子句学习（CDCL）的优化实现，既是独立的 SAT 求解工具，也是 Z3 SMT 框架的核心布尔推理组件，其设计围绕“性能”与“扩展性”展开。

结构字段	目的
m_assignment	变量当前赋值（真 / 假 / 未定义）
m_justification	变量赋值的推导原因（对应子句）
m_decision	标记变量是否可用于决策分支
m_activity	变量活动分数（决策启发式的依据）
m_phase	变量默认赋值方向（优先真 / 假）
m_best_phase	重启后最优的变量赋值方向
m_eliminated	跟踪已被消除的变量（简化求解）
m_external	标记暴露给理论求解器的变量

1.核心数据结构

2. 决策启发式（变量选择策略）

通过动态优先级选择变量，提升搜索效率：

2.1VSIDS（可变状态无关衰减和）：

冲突中涉及的变量活动分数增加，定期衰减所有变量分数，优先选近期参与冲突的变量；

2.2CHB（基于冲突历史的分支）：

基于 “多臂赌博机” 思想，跟踪变量的冲突参与 “奖励”，用指数移动平均线估计奖励，动态调整选择优先级。

3 扩展接口（与理论求解器的集成）

SAT 核心通过extension接口支持 SMT 场景的协作，提供以下能力：

特定理论传播：接收理论求解器的约束，进行命题层推理；

冲突解释：将理论求解器的冲突转换为命题子句，用于 CDCL 的冲突学习；

模型验证：结合理论求解器结果，验证完整模型的合法性；

增量求解：通过 “推送 / 弹出” 操作管理状态，支持动态添加约束。

基于 CDCL 框架的 SAT 求解器系统架构

1,CDCL SAT Solver（SAT 求解器核心组件）

包含VSIDS：SAT 求解中经典的变量选择启发式算法；

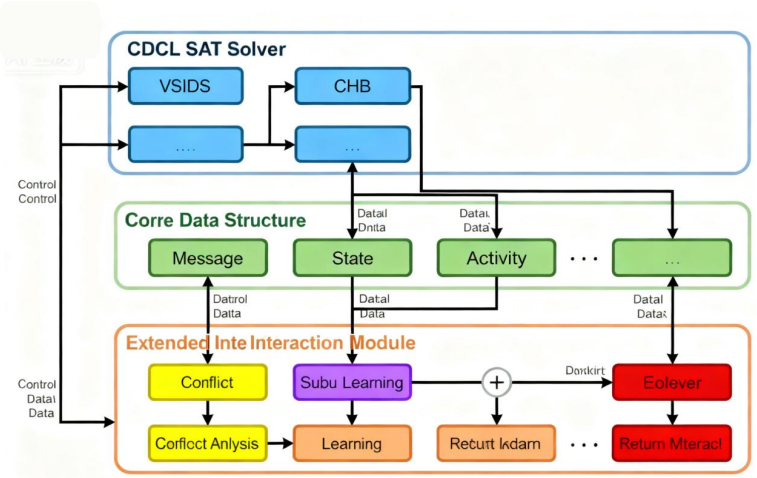
其他组件（如CHB）：是求解器的辅助启发式 / 子句管理组件。

2,Corre Data Structure（数据结构层）

负责存储 SAT 求解的核心数据，包含,Message：消息 / 信号传递；State：求解状态数据等。

3,Extended Interaction Module（扩展处理模块）

对应 CDCL 的冲突 - 学习 - 回溯流程等。



三，Z3工具演示

1.密码破译

在密钥未知情况下，通过已知明文推导出密钥并进行密码破译

通过弱加密 使用LCG作为密钥流生成器

加密过程为：明文 xor 密钥流=密文

通过已知的明文和密文可以反推得到密钥流

用z3求解原始密钥

```
z3开始求解...
耗时: 28.2178 秒

破解成功!
恢复的密钥: 0xdeadbeef
验证成功!
完整解密内容: b'FLAG{Z3_Solver_1s_Turing_Complete}'
```

```
未知的密钥: 0xdeadbeef
原始的明文: b'FLAG{Z3_Solver_1s_Turing_Complete}'
获取的密文: 2cb9487f188204a387210a306036dd373d087af2e356d137600b085dfc5b2[illegible]d8f207
已知的明文: b'FLAG'
反推得到的密钥流: ['0x6a', '0xf5', '0x9', '0x38']

z3开始求解...
```

三，Z3工具演示

2.基于范例的归纳合成

```
观察到的样例: [(2, 5), (4, 11), (10, 29)]
推导出的函数逻辑为: f(x) = 3 * x + -1
```

3.自动化漏洞挖掘

```
待验证路径条件: And(input_x > 100, input_y < 0, input_x + input_y == 50)
验证结果: 路径可达 (存在潜在Bug)
反例输入: x = 101, y = -51
验算检查: 101 + (-51) = 50
```

4.运筹&AI

```
求解成功！最优调度方案如下:
  时间片 1: Task B
  时间片 2: Task A
  时间片 3: Task D
  时间片 4: Task C
```

6.等价性证明

```
待证明公式: (x + y) == (x ^ y) + 2 * (x & y)
未找到反例！
结论: (x + y) == (x ^ y) + 2 * (x & y) 等式成立！
```

5.形式化验证

```
验证 32 位浮点数的结合律
验证结果: 计算机中 32 位浮点数的结合律不成立！
反例如下
    x = -1*(2**-28)
    y = 1.093758583068847656 25*(2**-36)
    z = 1.187569379806518554 6875*(2**-36)
```

四，Z3在其他领域应用

Z3 作为微软研究院开发的高效 SMT (可满足性模理论) 求解器，已从单纯的形式验证工具扩展为横跨多个技术领域的通用推理引擎。除已详述的程序验证、软件安全、硬件验证等核心应用外，Z3 在以下领域也发挥着关键作用：

1. 人工智能与机器学习安全

1.1神经网络验证：

验证神经网络在对抗样本攻击下的稳健性，防止输入微小扰动导致错误分类，检测神经网络输出是否符合预期范围，保障自动驾驶、医疗诊断等安全关键系统的可靠性，验证 AI 模型决策过程的可解释性，满足欧盟《人工智能法案》等法规要求。

1.2大语言模型 (LLM) 推理验证：

Meta 的 CoT-Verifier 利用 Z3 验证 LLM 链式思维推理过程，定位逻辑错误，形式化验证 AI 生成内容的一致性，防止 "幻觉" 问题，将 LLM 输出转换为形式证明语言，通过 Z3 验证其数学正确性。

2. 网络安全与渗透测试

2.1漏洞挖掘与分析：

符号执行结合 Z3 破解软件保护机制，如逆向分析混淆代码、加密算法，自动生成触发程序漏洞的精确输入，助力安全测试，分析恶意软件的控制流和数据依赖，提取攻击模式。

2.2安全协议验证：

验证 TLS、SSL 等通信协议的握手过程安全性，防止中间人攻击，分析区块链共识机制和智能合约安全性，如 IBM 的 Zeus 系统，验证 RBAC (基于角色的访问控制) 模型的权限分配一致性。

3. 自动驾驶与智能交通

3.1系统验证：

验证 ADAS 系统的感知、决策、控制全链路正确性，分析自动驾驶车辆在复杂路况下的行为，如交叉路口通行策略，形式化验证车辆控制系统的安全边界，防止软件故障导致危险。

3.2路径规划优化：

求解最优行驶路径，满足速度、油耗、安全等多重约束，验证自动泊车等复杂场景下的轨迹生成算法

4. 数据库与大数据处理

4.1查询优化：

数据库查询执行计划生成，通过约束求解选择最优连接顺序和索引策略，优化 OLAP 分析查询，如 Apache Doris 利用 Z3 进行 Data Trait 分析，解决数据库模式设计中的依赖冲突

4.2数据一致性保障：

验证数据库事务的 ACID 属性，防止并发操作导致数据异常，检测 ETL 流程中的数据转换约束是否满足，确保数据质量

四，Z3在其他领域应用

Z3 作为微软研究院开发的高效 SMT (可满足性模理论) 求解器，已从单纯的形式验证工具扩展为横跨多个技术领域的通用推理引擎。除已详述的程序验证、软件安全、硬件验证等核心应用外，Z3 在以下领域也发挥着关键作用：

5. 生物信息学与计算生物学

5.1基因分析：

Z3-4Biology 项目利用 SMT 求解器分析生物系统的复杂动态，通过符号推理解决生物建模挑战,基因调控网络建模与验证，分析基因表达的布尔逻辑关系,蛋白质结构预测与功能分析，验证分子对接的几何约束

5.2生物系统模拟：

建模细胞信号通路，验证分子相互作用网络的动态行为,分析代谢网络中的物质转化关系，求解最优营养供给方案。

6. 物联网 (IoT) 与嵌入式系统

6.1设备安全验证：

验证 IoT 设备固件的完整性，检测后门和漏洞,形式化验证嵌入式实时操作系统内核，如 FreeRTOS,确保智能家居设备的通信协议安全性，如 Z-Wave 认证。

6.2资源受限系统优化：

为低功耗设备生成最优代码，满足计算资源和能耗约束，验证物联网设备的身份认证和访问控制机制

7. 数学定理证明与形式化方法

7.1自动定理证明：

与 Coq、Lean 等证明助手集成，增强自动化推理能力，验证数学猜想，如 DeepMind 将强化学习引入 Lean 证明环境，辅助形式化数学知识库构建，如 Xena 项目。

7.2计算代数：

求解多项式方程组、整数规划等数学问题

验证密码学算法的数学安全性，如椭圆曲线加密。

五，总结

Z3 已从单一的形式验证工具发展为跨领域的约束求解与逻辑推理基础设施，其核心优势在于，多领域建模能力，统一处理布尔逻辑、整数、实数、位向量、字符串等多种数据类型的复杂约束；高效求解引擎，能处理大规模约束系统，在工业级应用中表现优异；与多种技术栈集成，支持 Python、C++、Java 等编程语言，便于嵌入现有系统。

展望未来，随着我们构建的系统越来越复杂、越来越智能，对自动化、可证明的正确性的需求只会越来越强烈。

随着形式化方法与 AI、安全、医疗等领域的深度融合，Z3 的应用边界将持续扩展，成为构建更可靠、更安全的软件与硬件系统的关键支撑技术。

感谢老师批评指正

THANK YOU FOR WATCHING

小组成员：郝仕达 肖宇 张子豪