

# 1. Introduction

**Satisfiability Modulo Theories (SMT)** 문제는 산술(arithmetic), 비트벡터(bit-vectors), 배열(arrays), 미해석 함수(uninterpreted functions) 등과 같은 **배경 이론(background theories)**의 조합에 따라 논리식의 만족 가능성을 결정하는 **의사결정 문제(decision problem)**이다.

**Z3**는 이러한 배경 이론을 해결하기 위한 특화된 알고리즘을 갖춘 **효율적인 SMT 솔버(SMT solver)**이다.

SMT 해결(SMT solving)은 **소프트웨어 분석(software analysis)**, **검증(verification)**, **기호적 실행(symbolic execution)** 도구들과 **상호보완적인 관계(synergetic relationship)**를 맺고 있다.

이는 프로그램과 명세(specification)에서 흔히 발견되는 도메인을 지원하는 데 초점을 두고 있기 때문이다.

이러한 도구들이 제시하는 쿼리 중 일부는 **지원되는 논리(logic)**로 표현될 수 있으며, 도구 개발자는 어떤 논리가 지원되는지, 그리고 그 논리식이 어떻게 해결되는지를 파악할 필요가 있다.

하지만 SMT 솔버와의 상호작용은 단일 공식(query)을 제시하는 데 그치지 않는다.

**사용 가능한 답변(usable answer)**을 얻기 위해서는 일련의 상호작용이 필요할 수 있으며, 이를 위해서는 어떤 **메서드(methods)**와 **조정 가능한 옵션(knobs)**이 제공되는지를 이해해야 한다.

요약하자면, 이 튜토리얼은 예제와 간단한 이론적 배경을 통해 다음과 같은 질문들에 답하고자 한다:

- Z3에는 어떤 기능들이 있으며, 그것들은 어떤 용도로 설계되었는가?
- Z3에서 사용되는 기본 알고리즘은 무엇인가?
- Z3를 기반으로 애플리케이션을 어떻게 프로그래밍할 수 있는가?

**그림 1(Figure 1)**은 **Z3 버전 4.8** 기준의 전체 시스템 다이어그램을 보여준다.

좌측 상단은 Z3의 인터페이스를 요약한 부분이다.

Z3와의 상호작용은 다음과 같은 방법으로 이루어진다:

- 텍스트 파일이나 파이프를 통해 **SMT-LIB2 스크립트**를 Z3로 전달하거나,
- 고급 프로그래밍 언어에서 **C 기반 API 호출을 프록시하는 함수(API calls)**를 이용하는 방식이다.

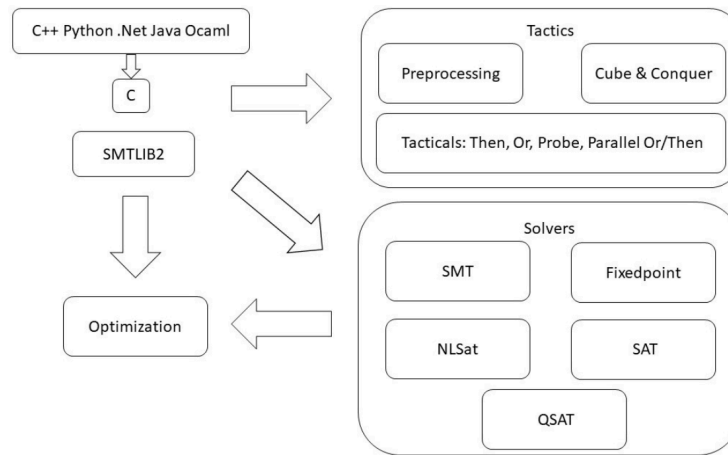
이 튜토리얼은 **Python 프론트엔드(front-end)**를 이용한 Z3 인터페이스를 중심으로 설명하며, 2장에서 Z3가 허용하는 **항(term)**과 **논리식(formula)**의 **추상 구문(abstract syntax)**을 설명한다.

- **이론(Theory)**: 3장에서 다룸
- **솔버(Solver)**: 4, 5, 6장에서 설명되며, 논리식의 만족 가능성을 결정함
- **전략(Tactics)**: 7장에서 다루며, 전처리 단순화(pre-processing simplification) 및 부분 목표(sub-goals) 생성을 지원함

Z3는 단순한 만족도 검사 외에도 **최적화(Optimization)** 기능(8장)을 제공한다.

이를 통해 **목적 함수(objective function)**에 따라 값을 최대화하거나 최소화하는 만족도 문제를 해결할 수 있다.

또한, **결과(enumerating consequences, backbone literals)**를 나열하는 특수한 절차도 제공하며, 이는 **섹션 4.6.6**에서 자세히 설명된다.



## 1.1. 자료(References / Resources)

Z3의 주요 참고 자료는 GitHub 저장소이다.

<https://github.com/z3prover/z3>

이 튜토리얼에서 실행 가능한 예제들은 다음 링크에서 확인할 수 있다.

<https://github.com/Z3Prover/doc/tree/master/programmingz3/code>

Z3를 사용하는 여러 시스템이 존재한다.

이들은 다양한 **프론트엔드(front-end)**를 사용하며, 어떤 것은 **OCaml**, 어떤 것은 **C++**, 또 다른 것은 **SMT-LIB2 텍스트 인터페이스**를 사용한다.

그중 Python 프론트엔드를 사용하는 대표적인 사례들은 다음과 같다:

- **Dennis Yurichev**은 퍼즐과 코드 분석 사례들을 모아 다수의 예제를 Python 프론트엔드를 이용해 제시하였다.

[SAT/SMT by Example \(PDF\)](#)

- **Ivy 시스템**은 Python으로 작성되었으며 Z3를 사용한다.

<https://github.com/Microsoft/ivy>

- **Angr 시스템**은 바이너리 분석 도구로, Python으로 작성되어 있으며 Z3를 사용한다.

<https://docs.angr.io/>

- 또한 **온라인 Z3 튜토리얼**도 존재한다.

<https://microsoft.github.io/z3guide>

## 1.2. 출처(Sources)

이 튜토리얼의 내용은 여러 출처에서 종합된 것이다.

일부 실행 예제는 **SAT 및 SMT 커뮤니티** 내에서 공유된 슬라이드 자료에서 유래했다.

첫 번째 SAT 예제는 **Armin Biere**의 SAT 튜토리얼에서 가져온 것이며,

그 외의 예제들은 **Natarajan Shankar**의 슬라이드에서 발췌되었다.



## Summary

### 무엇에 대한 문서인가?

- 이 튜토리얼은 **Z3**라는 **SMT(Satisfiability Modulo Theories)** 솔버(논리식 자동판단기)에 대해 프로그래머가 쉽게 이해하도록 설명한 문서임
- Z3는 논리식이 참인지, 거짓인지를 수학적으로 판단해주는 도구임
  - 주로 프로그램 검증, 분석, 자동화된 논리 추론 등에 쓰임

### Z3 튜토리얼의 목표

이 문서는 세 가지 질문에 답하는 걸 목표로 함.

1. Z3에는 어떤 기능이 있고, 어디에 쓰이나?
2. Z3는 내부적으로 어떤 알고리즘을 쓰나?
3. Z3를 이용해 직접 프로그램을 만들려면 어떻게 해야 하나?

### Z3의 주요 구성

- Z3는 **Python, C++, OCaml, SMT-LIB2** 등 여러 방식으로 사용할 수 있음
- 이 튜토리얼은 **Python API(파이썬 코드)**를 중심으로 설명함
- 내부 구조는 다음과 같이 나뉨:
  - **Theories** (이론들 - 산술, 배열, 비트벡터 등)
  - **Solvers** (논리식의 만족 가능성을 판단)
  - **Tactics** (공식 단순화, 전처리)
  - **Optimization** (값의 최대화/최소화 문제 해결)