求解的需求。

首先，可以使用之前讨论的预处理过程来查找可能“感兴趣”的路径和输入，以便使用符号执行进行探测并精确定位受这些输入影响的指令，这有助于避免为不感兴趣的路径或指令进行约束求解器的调用。符号执行引擎和约束求解器也可以缓存先前计算的（子）公式的结果，从而避免对相同的公式进行两次求解。

因为约束求解是符号执行的关键部分，所以接下来让我们更详细地探讨它是如何工作的。

12.2 使用 Z3 进行约束求解

符号执行以符号公式描述程序的操作，并使用约束求解器自动求解这些公式和解决有关程序的问题。要理解符号执行及其局限性，需要熟悉约束求解的过程。

在本节中使用约束求解器 Z3 来解释约束求解最重要的部分知识。Z3 由 Microsoft 开发，可从 GitHub 免费获得。

Z3 是可满足性模块理论（Statisfiability Modulo Theory，SMT）的约束求解器，专用于解决关于特定数学理论的公式的可满足性问题，如整数算法理论。这与纯布尔可满足性问题的约束求解器不同，后者不具备特定于理论的内置运算知识，如+或<等整数运算。Z3 具有解决涉及整数和位向量（二进制级数据表示）等公式的内置运算知识，这种领域相关的知识在解决符号公式（涉及该类运算）时非常有用。

请注意，Z3 这样的约束求解器是独立于符号执行引擎的程序，它并不仅限于符号执行。一些符号执行引擎甚至能够使用多个不同的约束求解器，具体取决于其偏好哪一种。Z3 是一个受欢迎的选择，因为它的特性非常适合符号执行，且提供了易于使用的 C / C ++和 Python 等语言的 API。它还带有一个命令行工具，可以使用该工具来对公式进行求解，稍后将对其进行介绍。

Z3 并不是万能的。虽然 Z3 和其他约束求解器对解决某些类别的可判定公式很有用，但它们无法对某些公式进行求解，或者需要很长时间才能解出结果，尤其当公式包含大量符号变量时更是如此。这就是保持约束尽可能简单的重要原因。

本书仅介绍 Z3 最重要的功能之一，但如果读者有兴趣，可以在线查看更全面的教程。

12.2.1 证明指令的可达性

让我们首先使用预装在虚拟机上的 Z3 命令行工具来求解一组简单的公式。使用 `z3 -in` 命令启动命令行工具，以从标准输入读取，或者使用 `z3 file` 命令从脚本文件读取输入。

Z3 的输入格式是 SMT-LIB 2.0 的一种扩展形式，而 SMT-LIB 2.0 是 SMT 约束