1，停机问题通常是针对带输入的函数而言的。停机问题的定义是：给定一个程序 p 和一个输入 i，判断是否程序 p 在输入 i 上会停机（即在有限步骤内结束执行）。

形式化定义的停机问题通常是不可解的，这意味着没有通用算法可以解决这个问题。这是图灵的停机定理的一部分，它证明了不存在一个程序，能够对于所有可能的程序 p 和输入 i，判断程序是否会停机。

所以，停机问题的不可解性适用于带输入的函数 Halt(p, i)。这个问题的证明并不依赖于是否有输入，因为输入是问题的一部分，它用来确定程序的行为。

我们也可以认为带输入的函数 p 实际上是不带输入的函数，但他有一个值为 i 的常量

2，

+	自然数	负	踩
自然数	自然数		
负	踩	负	
踩	踩	踩	踩

/	自然数	负	踩
自然数	踩	踩	踩
负	踩	自然数	踩
踩	踩	踩	踩

可以看出该抽象域的运算十分粗糙。加法中和原始分析精确度差别尚不明显，但除法中可以看出该抽象域的运算很粗糙，大部分的运算场景都不能给出准确的答案，而原始分析中针对正数和负数的除法操作是可以给出准确答案的 如 6/(-3)在此情景为踩，而原始分析为负