软件分析技术报告

杨晨阳 袁野 关智超

2019/12/22

1 目标

1. 给定文法 G 和约束 C ，生成程序 P 使得 P 属于 $L(G)$ 并且 P 满足约束 C。
2. 文法 G 的形式为 Synth-lib。
3. 约束 C、输出 P 的形式为 SMTlib。
4. 程序基于课程上给的 Python 框架。
5. 使用 z3 作为 checker。

对于 SyGuS 问题给定的输入：文法 G、约束 C。我们分别从文法 G 和约束 C 开始考虑该问题。

2 基于文法 G 的枚举算法

与传统的 SyGuS 枚举算法类似，通过自顶向下的遍历方式按语法依次展开。当搜索到的句子满足约束要求时，搜索结束。但考虑到我们无法预估满足约束的句子所包含的终结符的个数，即无法估计搜索的深度。因此需要优先使用*广度优先级搜索*，同时我们需要进行剪枝缩小搜索空间。

2.1 检查等价类

一个显然的剪枝方式是*检查等价类*。这样做的动机是，对于任何两个等效项 t 和 s，只需验证其中一项即可，因为 t 和 s 或者都是合理的解或者都不是。给定术语 t，我们写出 $t\downarrow$ 来表示其等价类。在搜索过程中，如果发现等价类，仅需要搜索其中最简形式。以下是我们在程序中指定的等价类：

1. $(x+y)\downarrow = (y+x)\downarrow$
2. $(x \geq y)\downarrow = (y \leq x)\downarrow$
3. $ite(x = 0 \wedge \omega, 0, x)\downarrow = x\downarrow$

2.2 忽略确定性条件

考虑在表达式 $ite(B\ I\ I)$ 中，如果出现在 ite 中的条件 B 不包含任何变元，则该条件可以被直接忽略。类似的，在条件中出现一些恒成立、恒不成立的式子也同样不会被继续展开，例如 $x \geq x$、$x \neq x$ 等。

此剪枝的依据是，对于一个正常的文法 G，当此条件恒成立或恒不成立时，$I \to ite(true, I_1, I_2)$ 等价于 $I \to I_1$，（我们期望）也可以由 I 直接生成。对 B 为 $false$ 时同理。

2.3 更多剪枝

- 考虑到如果在约束中没有出现任何数字，则在搜索的过程中不会展开出任何数字的形式，例如 $S \to 1$。
- 如果在约束中没有出现任何算术运算符，那么在搜索过程中不展开算术运算符的形式。
- 如果在 constrain 中出现了 k 次函数调用，那么在搜索到 ite 时直接将其展开 k 次的结果加入队列中，减小搜索深度。

3 基于约束 C 的搜索算法

考虑到测试集中的样例相对较为简单，因此我们可以通过尝试分析约束条件来直接生成目标句子 S'。如果分析失败，将会返回使用基于文法 G 的枚举算法。

考虑到一般给出的文法可以生成全部所需的句子形式。且由于在测试样例 *s1* 中，我们没有找到可以使用给定文法表达目标条件的句子，所以以下并未完全进行实现。

需要注意的是，我们在此过程中生成的代码是允许使用任何合法的字符（包括文法中未给出的终结符）。如果分析成功，则需要使用给定的文法 G，表达出所生成的句子 S'。

举例说明如下，如果文法中仅提供 and、not、$<$ 运算符，但三者是逻辑运算的完全集，可以通过他们表示 or、$=$、$\neq$ 等等。通过其等价形式替换

S' 中对应符号得到结果 S。除逻辑运算外，我们尝试给出了一些常见的关于数学运算的解决方案。

如果输入的文法不便于直接进行分析，理论上则可以进行搜索相应的符号或运算，例如搜索 add，满足约束 $(constraint\ (=\ (add\ x\ y)\ (+\ x\ y)))$。这种方法要求输入是一个完全集，例如 $\{-, /, and, not, <, 1, x\}$，一个典型的反例是 s1，因为 1 不可以通过文法生成。