



软件分析

程序合成：概率

熊英飞
北京大学



基于概率的方法



很多应用需要概率大的程序

文心大模型3.5

你好！最大子段和问题是指给定一个整数的列表，然后求这个列表中连续的一段，其中这一段的整数的和最大，求这个最大的和。你能写一个程序求解该问题吗？



当然可以，以下是一个Python程序，用于求解最大子段和问题：

```
python 复制代码
1 def maxSubarraySum(nums):
2     if not nums:
3         return 0
4
5     max_sum = nums[0]
6     for i in range(1, len(nums)):
7         max_sum = max(nums[i], max_sum + nums[i])
8
9     return max_sum
10
11 # 示例用法
12 nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
13 print(maxSubarraySum(nums)) # 输出 15
```

在这个程序中，我们定义了一个名为 `maxSubarraySum` 的函数，它接受一个整数列表作为输入，并返回这个列表中连续的一段，其中这一段的整数的和最大。

在函数中，我们首先检查输入列表是否为空，如果是，则返回0。否则，我们初始化一个变量 `max_sum` 为列表中的第一个元素，然后遍历列表中的所有元素。对于每个元素，我们将其与 `max_sum` 进行比较，并更新 `max_sum` 的值。最后，我们返回 `max_sum` 作为函数的结果。

在示例用法中，我们定义了一个整数列表 `nums`，并调用 `maxSubarraySum` 函数来求解最大子段和问题。最后，我



程序估计 Program Estimation

- 输入:
 - 一个程序空间 $Prog$
 - 一条规约 $Spec$
 - 概率模型 P , 用于计算程序的概率
- 输出:
 - 一个程序 $prog$, 满足
 - $prog = \operatorname{argmax}_{prog \in Prog \wedge prog \vdash spec} P(prog)$
- 给定不同的 P , 可以实现不同任务。
 - 如果 P 估计程序满足给定自然语言需求的概率, 那么可以完成从自然语言的代码生成
 - 如果 P 估计程序满足规约的概率, 那么可以用来加速传统程序合成



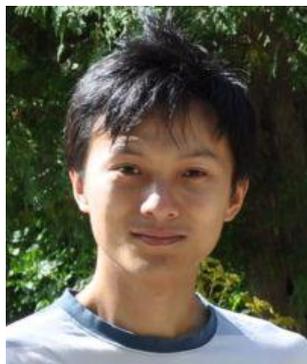
基本算法：穷举

- 用枚举的方法遍历空间中的程序
 - 对每个程序计算概率
 - 返回概率最大的程序
-
- 能否优化这个过程？



扩展枚举算法求解程序估计问题

玲珑框架L2S（包括本部分内容+语法上的静态预分析）



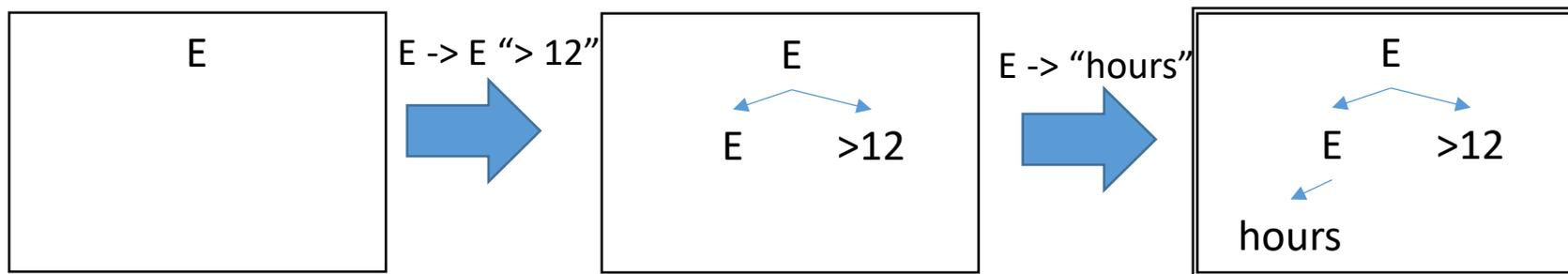
熊英飞
北京大学副教授



王博
北京交通大学讲师
北京大学博士

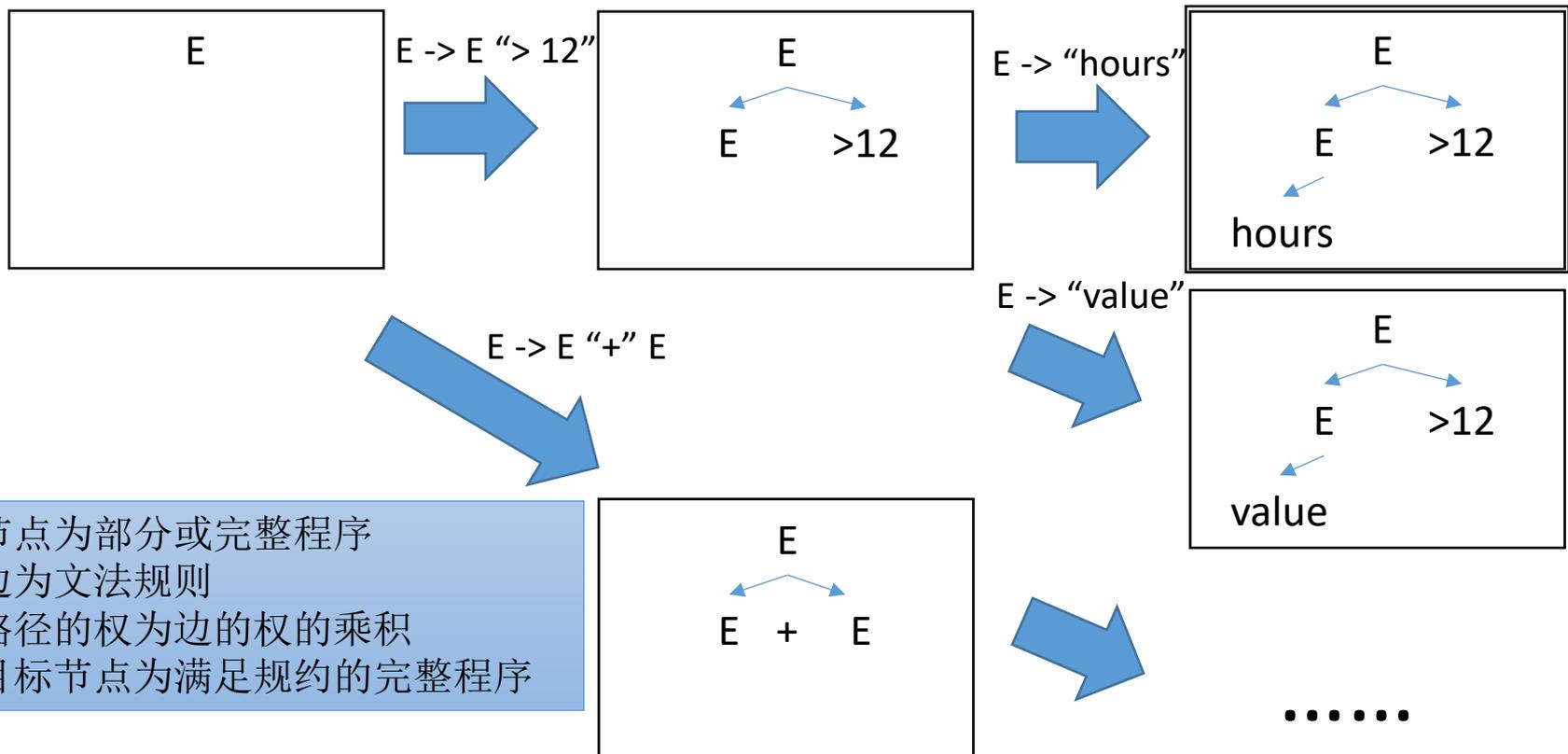


规则展开概率模型



- 按某种固定顺序选择节点展开
- $P(\text{prog}) = \prod_i P(\text{rule}_i \mid \text{rule}_1, \dots, \text{rule}_{i-1})$
 - rule_i : 展开程序所用的文法规则
- 可以用任意统计模型或机器学习模型实现

程序估计问题作为路径查找问题



- 节点为部分或完整程序
- 边为语法规则
- 路径的权为边的权的乘积
- 目标节点为满足规约的完整程序



如何求解概率最大的程序？

- 采用求解路径查找问题的标准算法
 - 迪杰斯特拉算法
 - 定向搜索 (Beam Search)
 - A*算法
-
- 当概率模型预测程序满足约束的概率时，这些算法帮助避免探索概率低的程序，达到加速效果



迪杰斯特拉算法

- 定义节点的权为到达该节点的路径的最大权
- 维护一个可达节点列表，并记录每个节点的权
- 选择权最大的节点，把该节点直接关联的新节点加入列表
- 如果某个节点已经没有未探索出边，则从列表中删除
- 反复上一步直到找到目标节点

注：在本问题中只能被一条路径到达，而在一般路径查找问题中，每个节点可以被多条路径达到，所以通用算法还需到达了旧节点时更新最大权。



迪杰斯特拉算法求解的例子

- $\langle E, 1 \rangle$
- $\langle E+E, 0.5 \rangle, \langle E-E, 0.4 \rangle, \langle x, 0.05 \rangle, \langle y, 0.05 \rangle$
- $\langle E-E, 0.4 \rangle, \langle x+E, 0.3 \rangle, \langle (E+E)+E, 0.1 \rangle, \langle y+E, 0.1 \rangle, \langle x, 0.05 \rangle, \langle y, 0.05 \rangle$
- $\langle x+E, 0.3 \rangle, \langle x-E, 0.2 \rangle, \langle y-E, 0.1 \rangle, \langle (E+E)+E, 0.1 \rangle, \langle y+E, 0.1 \rangle, \langle x, 0.05 \rangle, \langle y, 0.05 \rangle, \langle (E+E)-E, 0.05 \rangle, \langle (E-E)-E, 0.05 \rangle$
-



定向搜索 (Beam Search)

- 在迪杰斯特拉算法中不保留所有节点，只保留概率最大的k个
- 近似算法，不保证最优，也不保证找到结果



A*算法

- 节点n的权=到达该节点的权*h(n)
 - $h(n)$ =剩余路径权的上界
- 其他同迪杰斯特拉算法
- 如何知道剩余路径权的上界?
 - 假设存在函数 $\hat{P}(rule)$ ，满足
 - $\forall prog: \hat{P}(rule) \geq P(rule | prog)$
 - 在概率模型简单的时候可以遍历获得上界
 - 在语法展开式上做静态分析，分析出每个非终结符的概率上界
 - 从 $E \rightarrow E+E \mid x \mid y \mid \dots$
 - 得到方程 $\hat{P}(E) = \max(\hat{P}(E \rightarrow E + E)\hat{P}(E)\hat{P}(E), \hat{P}(E \rightarrow x), \hat{P}(E \rightarrow y), \dots)$
 - 剩余路径权的上界为所有未展开非终结符概率上界的积



剪枝

- 之前描述的剪枝过程仍然可以用于求解程序估计问题
- 判断出一个部分程序无法满足规约时，从列表中移除对应节点



和大模型的关系

- 规则展开概率模型等价于将代码建模为文法规则编码的序列
 - 保证文法正确性
 - 生成过程保留文法结构，可以应用程序分析剪枝
- 大模型将代码和文本统一建模为Token序列
 - 不保证文法正确性
 - 生成过程不保留文法结构，程序分析需要先实现 partial parser
- 大模型也能采用之前介绍的所有搜索算法，但由于计算资源消耗多，所以通常采用随机采样



扩展空间表示法求解 程序估计问题

空间表示法 vs 程序估计问题



- 能否采用空间表示法求解程序估计问题?
- 方案:
 - 套入CEGIS框架得到输入输出样例
 - 首先根据输入输出样例建文法
 - 然后将文法作为程序空间，用玲珑框架求解概率最大的程序
 - 为便于统计，计算规则概率时忽略返回值约束
- 问题：建文法没有被概率引导，无法加速



MaxFlash

- MaxFlash
 - 2020年由北京大学吉如一等人提出
 - 采用概率引导语法构建
 - 效率超过FlashMeta达400-2000倍



吉如一
北京大学博士生



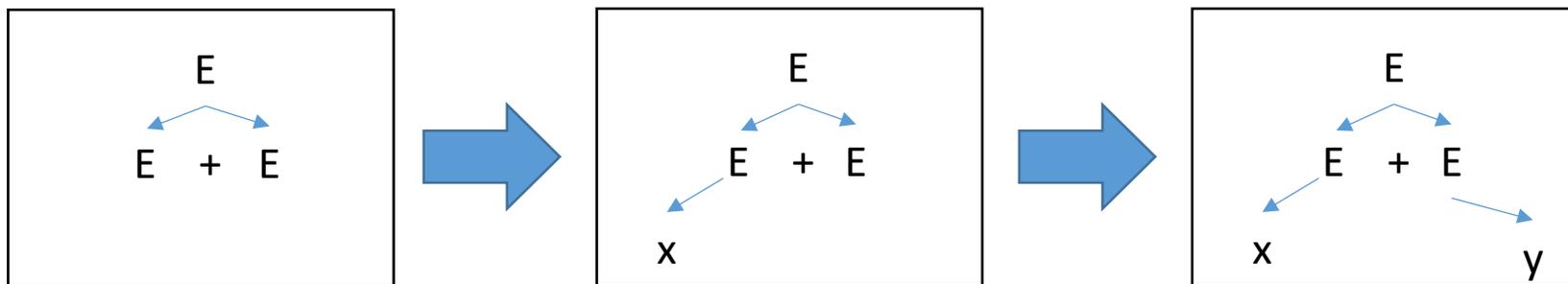
概率计算和文法构建的矛盾

- 假如我们获得如下文法展开式
 - $[acc]S \rightarrow [a]S + [cc]S$
- $[a]S$ 的展开式和 $[cc]S$ 的展开式是两个独立问题，可以分别求解，形成动态规划算法
- 但 $[cc]S$ 的展开式的概率取决于 $[a]S$ 是如何展开的，无法分治
- 导致在创建文法的时候无法应用概率引导



解决方案：自顶向下预测模型

- 节点展开规则概率只取决于其祖先，即兄弟节点的展开规则相互独立
 - 通常定义为依赖最近k层祖先节点



$$P(x + y) = P(E \rightarrow E + E \mid \perp)P(E \rightarrow x \mid E)P(E \rightarrow y \mid E)$$



统一概率计算和文法构建

带祖先的文法	产生式概率	最优程序和概率
$[acc \perp]S \rightarrow [a S]S + [cc S]S$	0.9	
$ [ac S]S + [c S]S$	0.9	

可采用动态规划算法独立求解每个子问题



基于概率的剪枝：分支限界

- 假设我们认为最优程序的概率应大于0.3

带祖先和概率下界的文法	概率	说明
$[acc \perp 0.3]S \rightarrow [a S 0.33]S + [cc S 0.33]S$	0.9	$0.3/0.9=0.33$
$[cc S 0.33]S \rightarrow [c S 3.3]S + [c S 3.3]S$	0.1	$0.33/0.1=3.3$
$ \psi$	0.3	$0.33/0.3=1.1$



基于概率的剪枝：估价函数

- 静态分析非终结符的概率上界

祖先	非终结符	概率上界
L	S	0.081
S	S	0.3

- 假设我们认为最优程序的概率应大于0.3

带祖先和概率下界的文法	概率	说明
$[acc L 0.3]S \rightarrow [a S 1.11]S + [cc S 1.11]S$	0.9	$0.3/0.9/0.3=1.11 > 0.3$



基于概率的剪枝：迭代加深

- 如何知道最优程序的概率应大于多少？
 - 设置一个概率下界，并逐步放宽
 - 如，一开始是0.1，然后每次除以10

基于概率的剪枝：复用于子问题



- 概率下界基本不可能相同
 - 动态规划退化成分治
- 需要复用概率下界不同的子问题
 - 考虑两个除了概率下界不同以外，其他都一样的子问题 $(P, 0.2)$, $(P, 0.1)$
 - Case 1: $(P, 0.2)$ 先于 $(P, 0.1)$
 - $(P, 0.2)$ 有解，则同样是 $(P, 0.1)$ 的解；
 - $(P, 0.2)$ 无解，则可以更新 P 的估价函数
 - Case 2: $(P, 0.1)$ 先于 $(P, 0.2)$
 - $(P, 0.1)$ 有解，则可以直接得到 $(P, 0.2)$ 的解（因为总是搜索概率最大的结果）
 - 无解， $(P, 0.2)$ 同样无解



参考文献

- Yingfei Xiong, Bo Wang, Guirong Fu, Linfei Zang. Learning to Synthesize. GI'18: Genetic Improvement Workshop, May 2018.
- Yingfei Xiong, Bo Wang. L2S: a Framework for Synthesizing the Most Probable Program under a Specification. ACM Transactions on Software Engineering Methodology, Online First, Dec 2021.
- Ruyi Ji, Yican Sun, Yingfei Xiong, Zhenjiang Hu. Guiding Dynamic Programming via Structural Probability for Accelerating Programming by Example. OOPSLA'20: Object-Oriented Programming, Systems, Languages, and Applications 2020, November 2020.



参考文献

- Zeyu Sun, Qihao Zhu, Lili Mou, Yingfei Xiong, Ge Li, Lu Zhang. A Grammar-Based Structural CNN Decoder for Code Generation. AACL'19: Thirty-Third AACL Conference on Artificial Intelligence, January 2019.
- Zeyu Sun, Qihao Zhu, Yingfei Xiong, Yican Sun, Lili Mou, Lu Zhang. TreeGen: A Tree-Based Transformer Architecture for Code Generation. AACL'20: Thirty-Fourth AACL Conference on Artificial Intelligence, January 2020.
- Qihao Zhu, Zeyu Sun, Yuanan Xiao, Wenjie Zhang, Kang Yuan, Yingfei Xiong, Lu Zhang. A Syntax-Guided Edit Decoder for Neural Program Repair. ESEC/FSE'21: ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, August 2021.