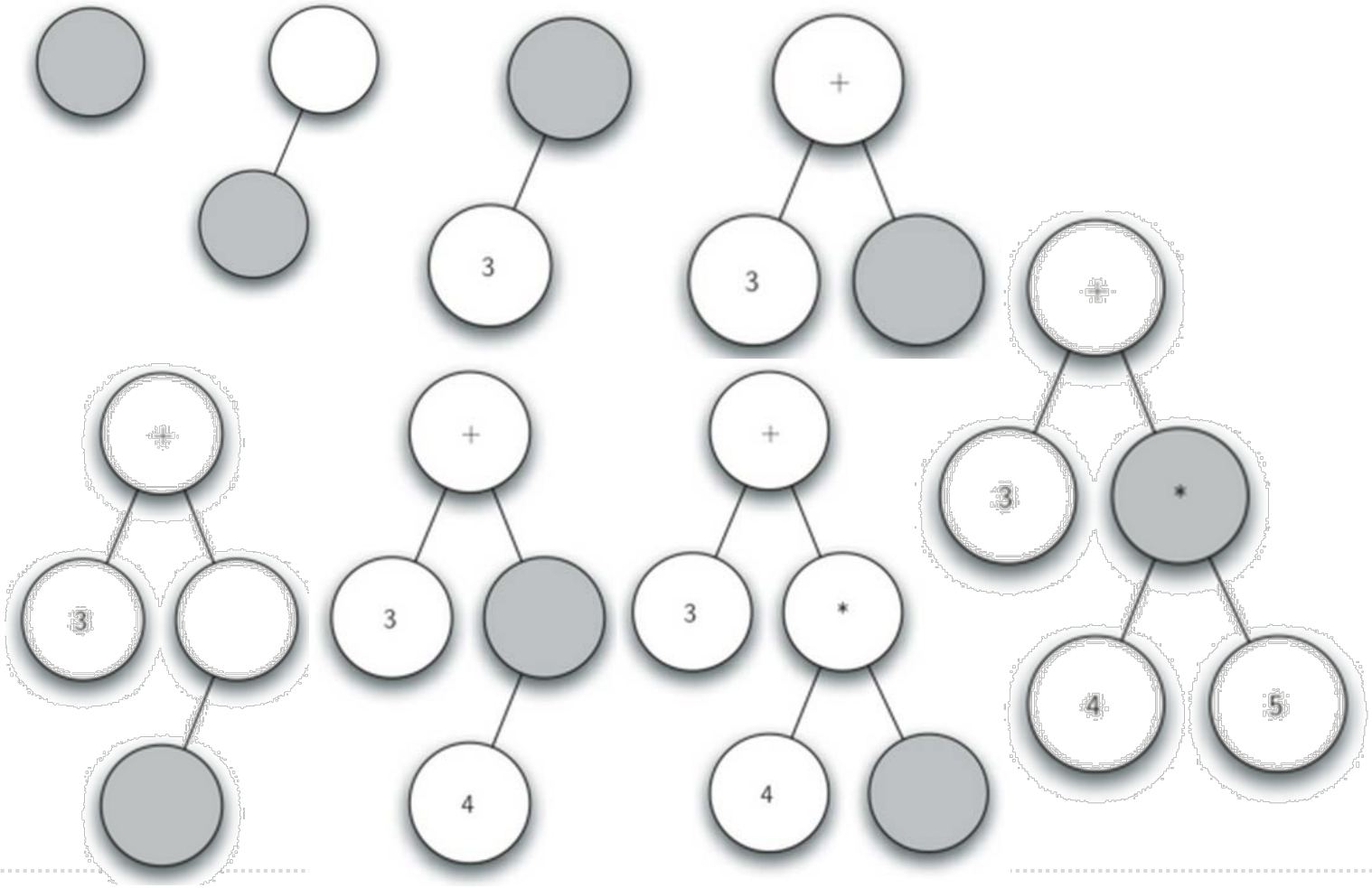
数据结构与算法（Python版）

树的应用：表达式解析（下）

陈斌 北京大学 gischen@pku.edu.cn

建立表达式解析树：实例

- 全括号表达式：（3+(4*5)）



建立表达式解析树：思路

- 从图示过程中我们看到，创建树过程中关键的是对当前节点的跟踪
  - 创建左右子树可调用insertLeft/Right
  - 当前节点设置值，可以调用setRootVal
  - 下降到左右子树可调用getLeft/RightChild
  - 但是，上升到父节点，这个没有方法支持！
- 我们可以用一个栈来记录跟踪父节点
  - 当前节点下降时，将下降前的节点push入栈
  - 当前节点需要上升到父节点时，上升到pop出栈的节点即可！

建立表达式解析树：代码


```
def buildParseTree(fpexp):
    fplist = fpexp.split()
    pStack = Stack()
    eTree = BinaryTree('')
    pStack.push(eTree)
    currentTree = eTree
    for i in fplist:
        if i == '(':
            currentTree.insertLeft('')
            pStack.push(currentTree)
            currentTree = currentTree.getLeftChild()
        elif i not in ['+', '-', '*', '/', ')']:
            currentTree.setRootVal(int(i))
            parent = pStack.pop()
            currentTree = parent
        elif i in ['+', '-', '*', '/']:
            currentTree.setRootVal(i)
            currentTree.insertRight('')
            pStack.push(currentTree)
            currentTree = currentTree.getRightChild()
        elif i == ')':
            currentTree = pStack.pop()
        else:
            raise ValueError
    return eTree
```

入栈下降

表达式开始

入栈下降

操作数

出栈上升

操作符

表达式结束

出栈上升

利用表达式解析树求值：思路

- 创建了表达式解析树，可用来进行求值
- 由于二叉树BinaryTree是一个递归数据结构，自然可以用递归算法来处理
- 求值递归函数evaluate

  由前述对子表达式的描述，可从树的底层子树开始，逐步向上层求值，最终得到整个表达式的值

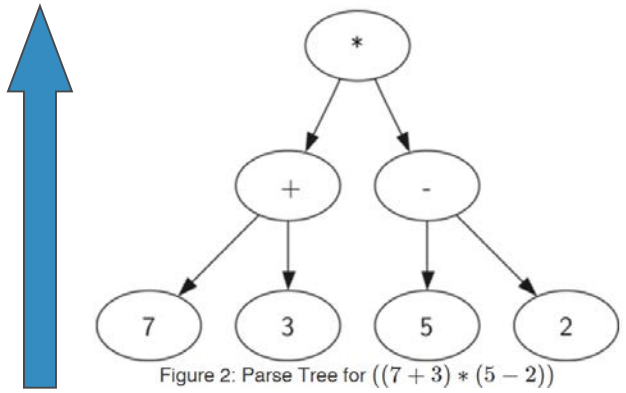


Figure 2: Parse Tree for $((7+3)*(5-2))$

利用表达式解析树求值：思路

- **求值函数evaluate的递归三要素：**

  基本结束条件：叶节点是最简单的子树，没有左右子节点，其根节点的数据项即为子表达式树的值

  缩小规模：将表达式树分为左子树、右子树，即为缩小规模

  调用自身：分别调用evaluate计算左子树和右子树的值，然后将左右子树的值依根节点的操作符进行计算，从而得到表达式的值

利用表达式解析树求值：思路

❖ 一个增加程序可读性的技巧：函数引用

```
import operator

op= operator.add
```

```
>>> import operator
>>> operator.add
<built-in function add>
>>> operator.add(1,2)
3
>>> op= operator.add
>>> n= op(1,2)
>>> n
3
```

利用表达式解析树求值：代码


```
import operator
def evaluate(parseTree):
    opers = {'+':operator.add, '-':operator.sub, \
             '*':operator.mul, '/':operator.truediv}

    leftC = parseTree.getLeftChild()
    rightC = parseTree.getRightChild()

    if leftC and rightC:
        fn = opers[parseTree.getRootVal()]
        return fn(evaluate(leftC),evaluate(rightC))
    else:
        return parseTree.getRootVal()
```

缩小规模

递归调用

基本结束条件