

04-F1

栈与队列

中缀表达式求值：问题与构思

邓俊辉

deng@tsinghua.edu.cn

知实而不知名，知名而不知实，皆不知也

应用

❖ 给定语法正确的算术表达式s，计算与之对应的数值

❖ UNIX: \$ **echo \$((0 + (1 + 23) / 4 * 5 * 67 - 8 + 9))**

❖ DOS: \> **set /a (!0 ^<^< (1 - 2 + 3 * 4)) - 5 * (6 ^| 7) / (8 ^^ 9)**

❖ PS: GS> 0 1 23 add 4 div 5 mul 67 mul add 8 sub 9 add =

❖ Excel: = COS(0) + 1 - (2 - POWER((FACT(3) - 4), 5)) * 67 - 8 + 9

❖ Word: = NOT(0) + 12 + 34 * 56 + 7 + 89

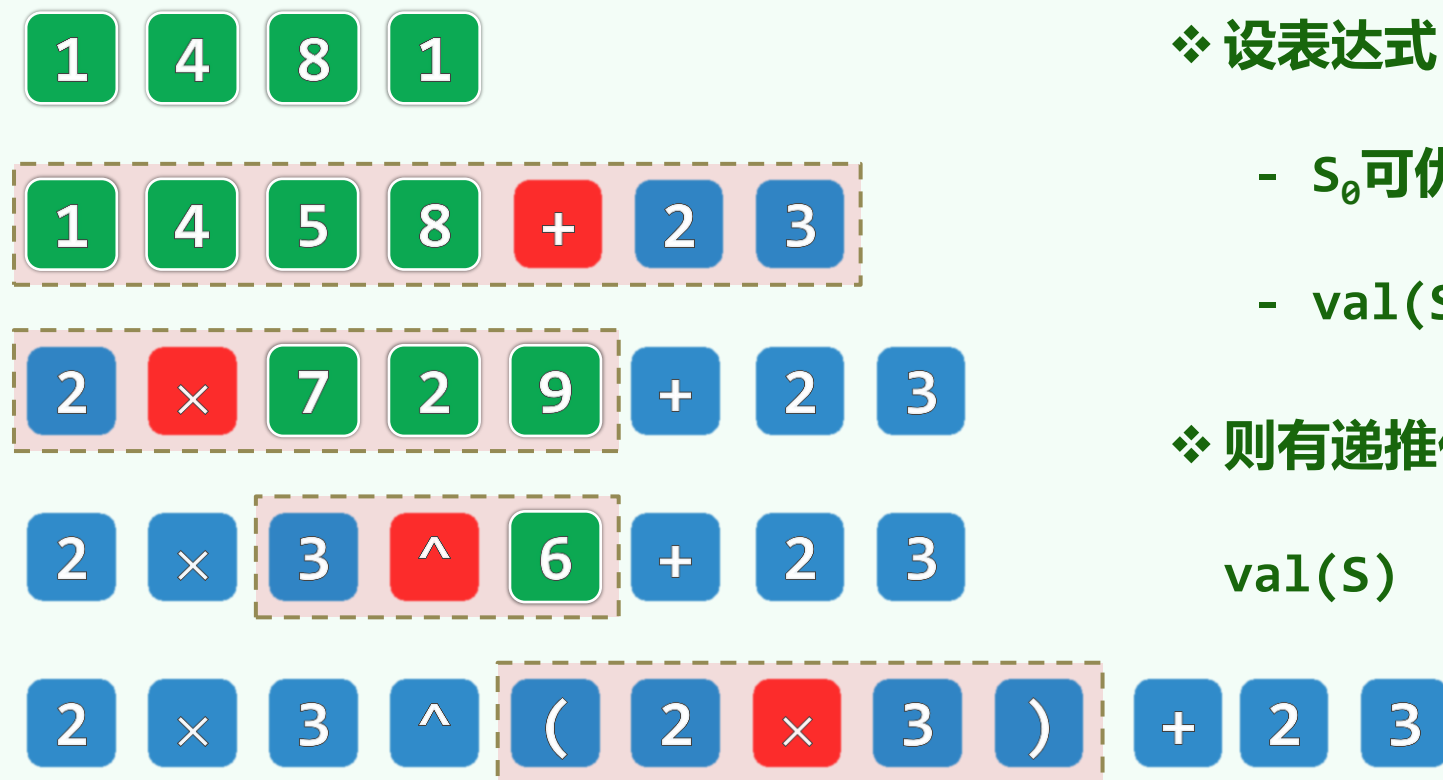
❖ calc: 0 ! + 12 + 34 * 56 + 7 + 89 =

❖ calc: 0 ! + 1 - (2 - (3 ! - 4) y 5) * 67 - 8 + 9 =

减而治之

❖ 优先级高的局部执行计算，并被代以其数值

运算符渐少，直至得到最终结果



❖ $\text{str}(v)$: 数值 v 对应的字符串 (名)

$\text{val}(S)$: 符号串 S 对应的数值 (实)

❖ 设表达式: $S = S_L + S_\theta + S_R$

- S_θ 可优先计算, 且

- $\text{val}(S_\theta) = v_\theta$

❖ 则有递推化简关系

$\text{val}(S) = \text{val}(S_L + \text{str}(v_\theta) + S_R)$

优先级

❖ 难点：如何高效地找到可优先计算的 s_0

（亦即，其对应的运算符）？

1 4 8 1

1 4 5 8 + 2 3

2 × 7 2 9 + 2 3

2 × 3 ^ 6 + 2 3

2 × 3 ^ (2 × 3) + 2 3

❖ 与括号匹配迭代版类似，但亦不尽相同

- 不能简单地按“左先右后”次序处理各运算符
- 此时，需要考虑更多因素...

❖ 约定俗成的优先级：

1 + 2 * 3 ^ 4 !

可强行改变次序的括号：

(((1 + 2) * 3) ^ 4) !

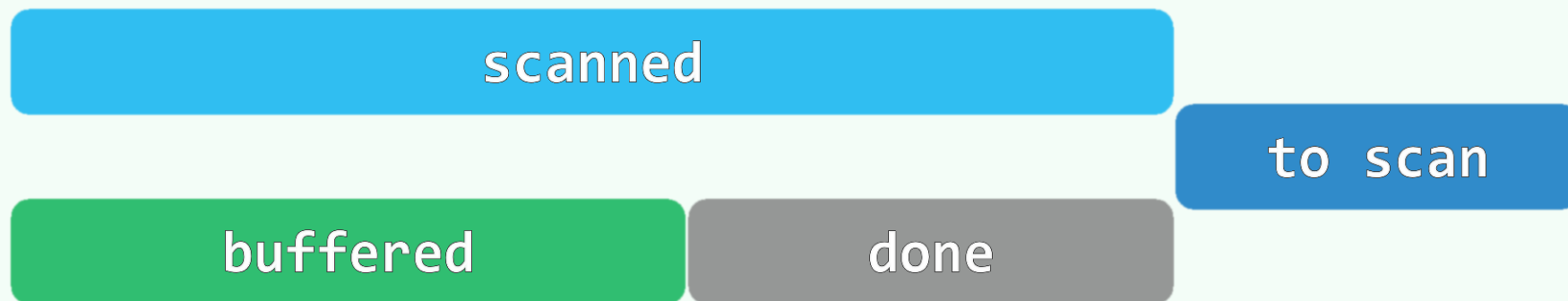
延迟缓冲

❖ 仅根据表达式的前缀，不足以确定各运算符的计算次序

只有获得足够的后续信息，才能确定其中哪些运算符可以执行

❖ 体现在求值算法的流程上

为处理某一前缀，必须提前预读并分析更长的前缀



❖ 为此，需借助某种支持延迟缓冲的机制...

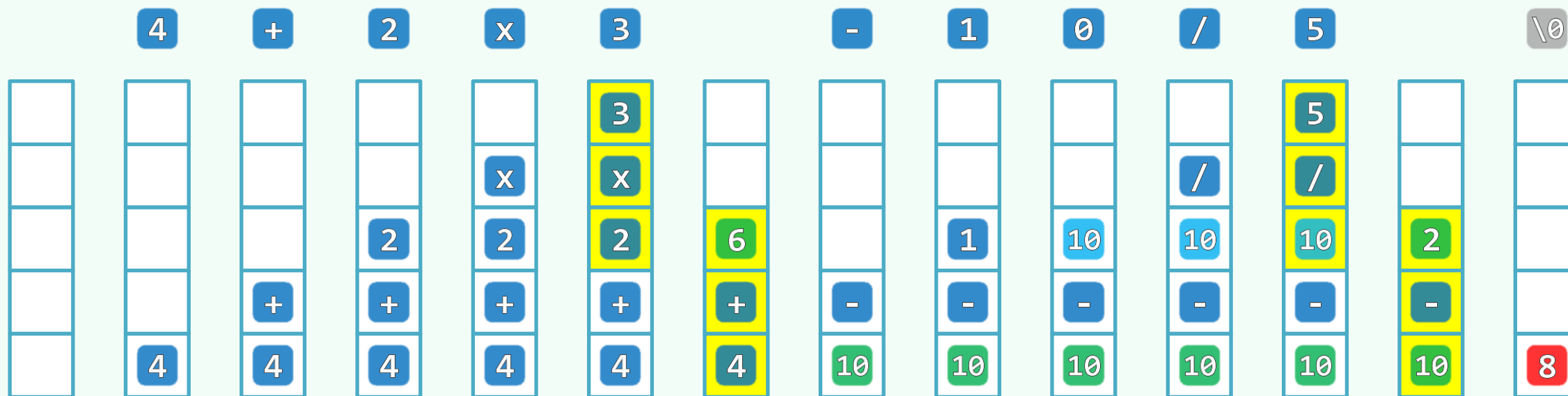
求值算法 = 栈 + 线性扫描

❖ 自左向右扫描表达式，用栈记录已扫描的部分（含已执行运算的结果）

栈的**顶部**存在**可优先计算**的子表达式

？ 该子表达式退栈；计算其数值；计算结果进栈

： 当前字符进栈，转入下一字符



❖ 只要语法正确，则栈内最终应只剩一个元素 //即表达式对应的数值