中国科学技术大学
University of Science and Technology of China

USTC, CHINA
ADSLAB
先进数据系统实验室

《编译原理与技术》
期末复习

计算机科学与技术学院

李 诚

07/01/2020

考试说明

- 参见gitlab issue：http://210.45.114.30/gbxu/notice_board/issues/281

时间：2020-01-11 08:30~10:30 地点：西区3C201和3C202

内容： 考试形式为闭卷，覆盖内容包括期中考试考过的词法分析和语法分析，但是侧重于期中以后的内容！

座位表在附件中：

编译原理和技术-3C202考场座位表-63人7列.pdf

编译原理和技术-3C201考场座位表-62人7列.pdf

课程加分申报(直接加到总成绩)

- 课堂答题加分老师有记录，每次认可的回答获得0.5分，十次为上限
- 其他加分包括git issue和邮件交流，请大家自行整理后，**于2020年1月18日上午11:59之前**发邮件到sxy799@mail.ustc.edu.cn，并抄送chengli7@ustc.edu.cn
  - 如果是git issue上的tutorial，请附上issue id
  - 如果是git issue上的回帖，请附上issue id和自己回答的截图
  - 如果是邮件交流，请附上邮件内容
  - 助教会按照大家的申报，审核并反馈认定结果。十次为限！

课程加分申报(直接加到总成绩)

中国科学技术大学
University of Science and Technology of China

- 适合加分的内容
  - 明确指出课程实验中的错误、纰漏的
  - 帮助其他同学解决问题（包括教程、建设性的帮助等）
  - 提出有价值的课程内容相关提问的
  - 对于课程的后续开展提出有效改进方案的
- 以下内容不加分
  - 询问或讨论组队问题、课程时间安排、个人得分等
  - 吐槽贴
  - 各种求助类问题

知识点考核

- 词法分析和语法分析会涉及
- 期中考试以后的占主要部分
- 数据流分析不做考察
- 参考资料主要以课后习题为主
- 本学期的实验也会有考核
- 出题指导思想：覆盖全面、难度适中

重要知识点复习思路

中国科学技术大学
University of Science and Technology of China

- 词法分析
  - 能够为字串写正则表达式
  - 掌握正则表达式到NFA和DFA的转换
  - DFA的化简

重要知识点复习思路

- 语法分析
  - 最左和最右推导，判断二义文法
  - LL(1)文法的判定，FIRST和FOLLOW集合的计算
  - 移进归约栈操作
  - 简单的SLR方法

LR分析法总结

		SLR	LALR	LR(1)
初始状态		$[S' \to S]$	$[S' \to S, \$]$	$[S' \to S, \$]$
项目集		LR(0) CLOSURE(I)	合并LR(1)项目集族的同心项目集	LR(1), CLOSURE(I) 搜索符考虑FISRT(βa)
动作	移进	$[A \to \alpha a \beta] \in I_i$ $GOTO(I_i, a) = I_j$ $ACTION[i, a] = sj$	与LR(1) 一致	$[A \to \alpha a \beta, b] \in I_i$ $GOTO(I_i, a) = I_j$ $ACTION[i, a] = sj$
	归约	$[A \to \alpha] \in I_i, A \neq S'$ $a \in FOLLOW(A)$ $ACTION[i, a] = rj$	与LR(1) 一致	$[A \to \alpha, a] \in I_i$ $A \neq S'$ $ACTION[i, a] = rj$
	接受	$[S' \to S \cdot] \in I_i$ $ACTION[i, \$] = acc$	与LR(1) 一致	$[S' \to S \cdot, \$] \in I_i$ $ACTION[i, \$] = acc$
	出错	空白条目	与LR(1) 一致	空白条目
GOTO		$GOTO(I_i, A) = I_j$ $GOTO[i, A] = j$	与LR(1) 一致	$GOTO(I_i, A) = I_j$ $GOTO[i, A] = j$
状态量		少(几百)	与SLR一样	多(几千)

重要知识点复习思路

- 语法制导翻译
  - 掌握语法制导翻译方案
  - 掌握简单的综合属性和继承属性计算
  - S-属性定义与LR分析方法的结合
    - 语义规则到栈操作代码的改写

将S-SDD转换为SDT

- 将一个S-SDD转换为SDT的方法：
  - 将每个语义动作都放在产生式的最后
  - 称为“后缀翻译方案”

S-SDD

产生式	语义规则
(1) $L \rightarrow E$ n	$L.val = E.val$
(2) $E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
(3) $E \rightarrow T$	$E.val = T.val$
(4) $T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
(5) $T \rightarrow F$	$T.val = F.val$
(6) $F \rightarrow (E)$	$F.val = E.val$
(7) $F \rightarrow$ digit	$F.val =$ digit.$lexval$

SDT

(1) $L \rightarrow E$ n { $L.val = E.val$}
(2) $E \rightarrow E_1 + T$ {$E.val = E_1.val + T.val$}
(3) $E \rightarrow T$ { $E.val = T.val$}
(4) $T \rightarrow T_1 * F$ { $T.val = T_1.val \times F.val$}
(5) $T \rightarrow F$ { $T.val = F.val$}
(6) $F \rightarrow (E)$ { $F.val = E.val$}
(7) $F \rightarrow$ digit { $F.val =$ digit.$lexval$}

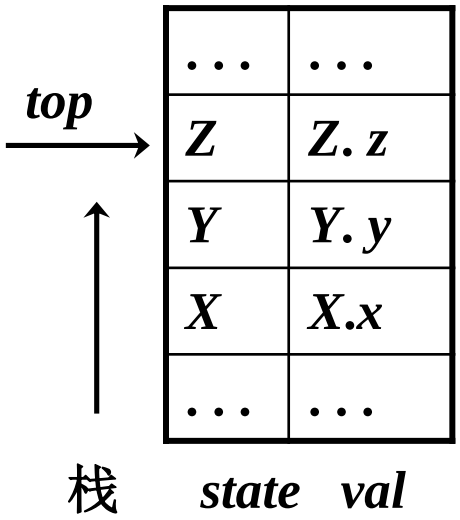
S-属性定义的SDT实现-2

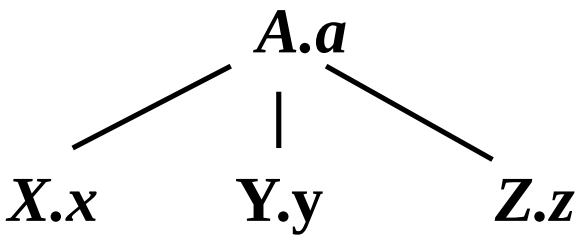
- 可以通过扩展的LR语法分析栈来实现
  - 在分析栈中使用一个附加的域来存放综合属性值。若支持多个属性，那么可以在栈中存放指针
  - 每一个栈元素包含状态、文法符号、综合属性三个域
    - 也可以将分析栈看成三个平行的栈，分别是状态栈、文法符号栈、综合属性栈，分开看的理由是，入栈出栈并不完全同步
  - 语义动作将修改为对栈中文法符号属性的计算

S-属性定义的SDT实现-2

❑可以通过扩展的LR语法分析栈来实现

❖考虑产生式 $A \rightarrow XYZ$



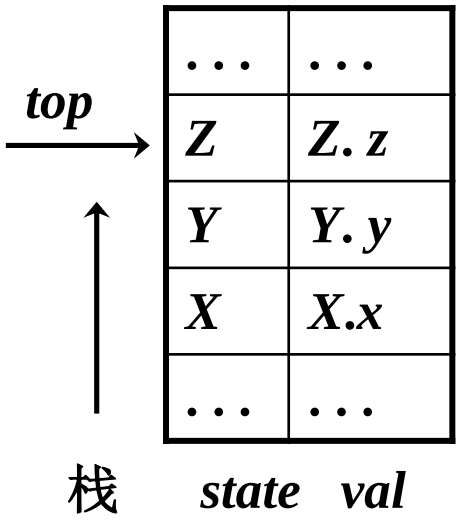


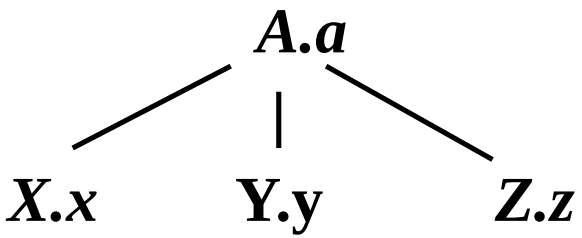
$A \rightarrow XYZ \{A.a = f(X.x, Y.y, Z.z)\}$

S-属性定义的SDT实现-2

❑可以通过扩展的LR语法分析栈来实现

❖考虑产生式 $A \rightarrow XYZ$





$A \rightarrow XYZ \{A.a = f(X.x, Y.y, Z.z)\}$

语义动作

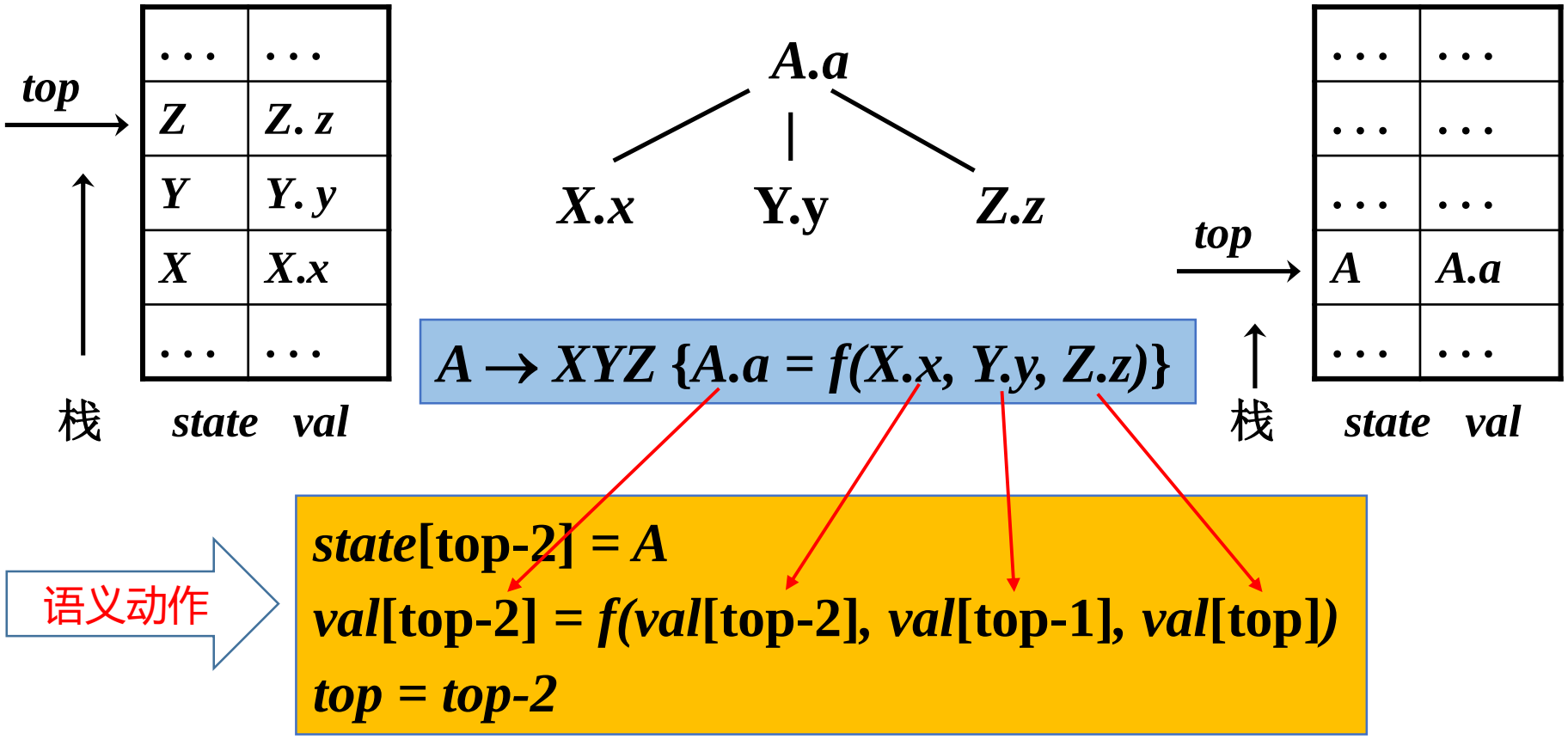
state[top-2] = *A*

val[top-2] = *f*(*val*[top-2], *val*[top-1], *val*[top])

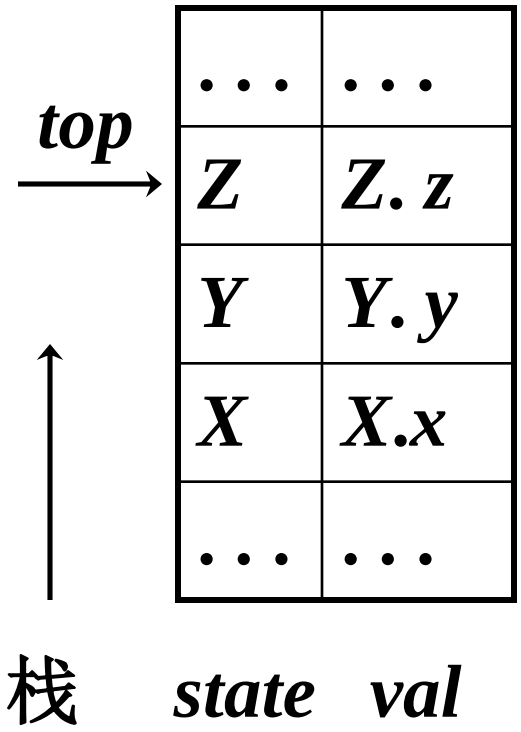
top = *top*-2

❑可以通过扩展的LR语法分析栈来实现

❖考虑产生式 $A \rightarrow XYZ$



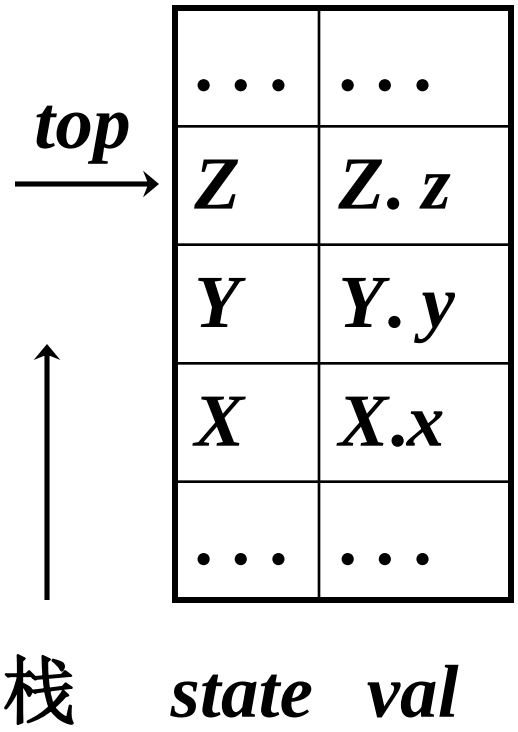
❑简单计算器的语法制导定义改成栈操作代码



产 生 式	语 义 规 则
$L \rightarrow E$ n	*print* (*E.val*)
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	*E.val* = *T.val*
$T \rightarrow T_1 * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	*T.val* = *F.val*
$F \rightarrow (E)$	*F.val* = *E.val*
$F \rightarrow$ digit	*F.val* = digit.*lexval*

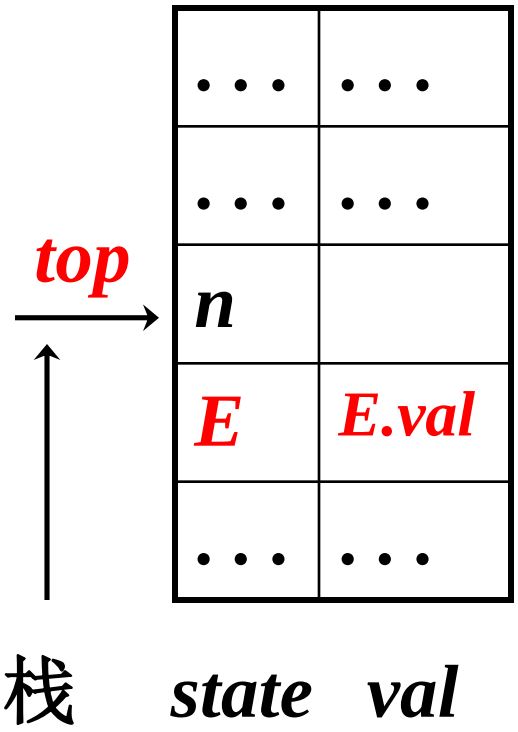
SLR分析栈中实现计算器

❑简单计算器的语法制导定义改成栈操作代码



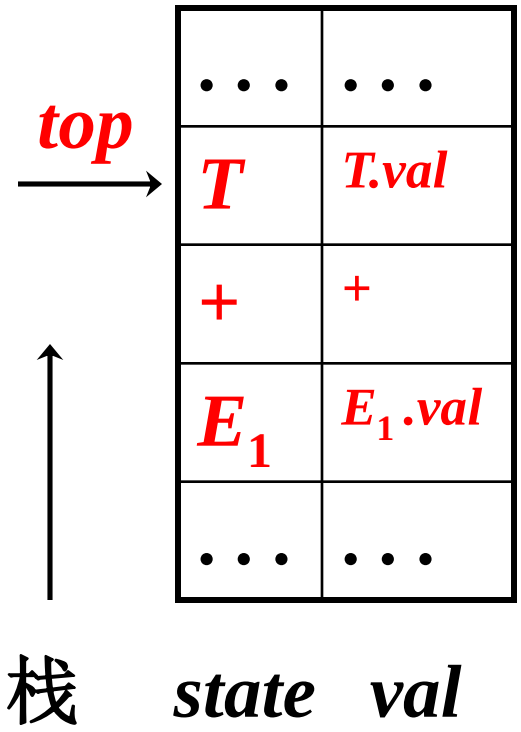
产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*E.val*)
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	*E.val* = *T.val*
$T \rightarrow T_1 * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	*T.val* = *F.val*
$F \rightarrow (E)$	*F.val* = *E.val*
$F \rightarrow$ digit	*F.val* = digit.*lexval*

❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*val* [*top*–1])
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
$E \rightarrow T$	$E.val = T.val$
$T \rightarrow T_1 * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow (E)$	$F.val = E.val$
$F \rightarrow$ digit	$F.val =$ digit.*lexval*

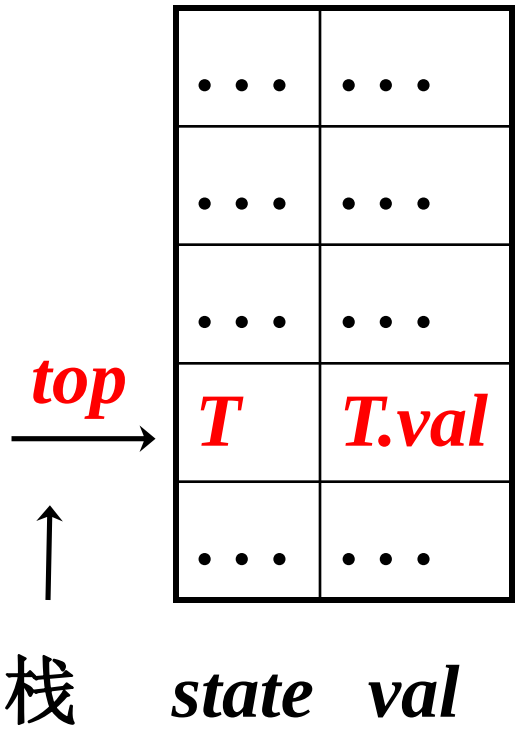
❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*val* [*top*–1])
$E \rightarrow E_1 + T$	*val* [*top* –2] = *val* [*top* –2]+*val* [*top*]
$E \rightarrow T$	*E.val* = *T.val*
$T \rightarrow T_1 * F$	*T.val* = T_1.*val* * *F.val*
$T \rightarrow F$	*T.val* = *F.val*
$F \rightarrow (E)$	*F.val* = *E.val*
$F \rightarrow$ digit	*F.val* = digit.*lexval*

SLR分析栈中实现计算器

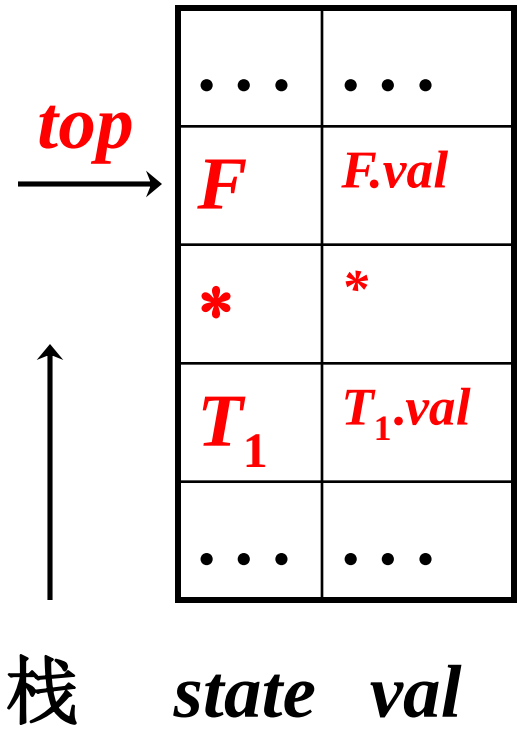
❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*val* [*top*–1])
$E \rightarrow E_1 + T$	*val* [*top* –2] = *val* [*top* –2]+*val* [*top*]
$E \rightarrow T$	值不变，无动作
$T \rightarrow T_1 * F$	$T.val = T_1.val * F.val$
$T \rightarrow F$	$T.val = F.val$
$F \rightarrow (E)$	$F.val = E.val$
$F \rightarrow$ digit	$F.val =$ digit.*lexval*

SLR分析栈中实现计算器

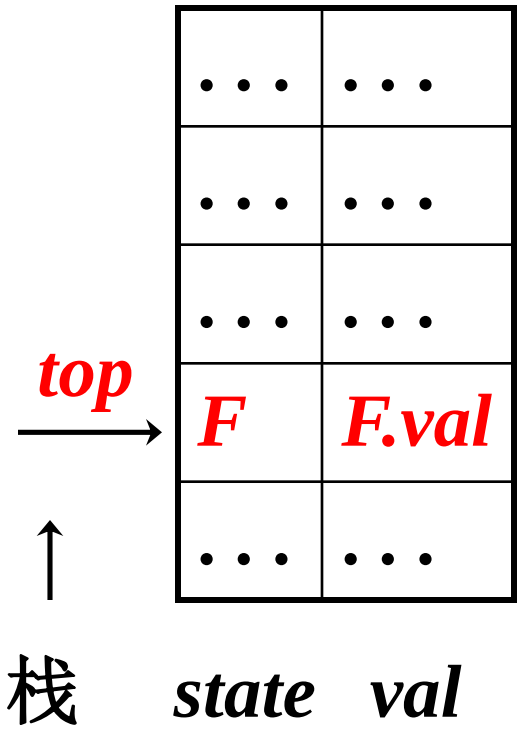
❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \to E$ n	*print* (*val* [*top*–1])
$E \to E_1 + T$	*val* [*top* –2] = *val* [*top* –2]+*val* [*top*]
$E \to T$	
$T \to T_1 * F$	*val* [*top* –2] = *val* [*top* –2]×*val* [*top*]
$T \to F$	*T.val* = *F.val*
$F \to (E)$	*F.val* = *E.val*
$F \to$ digit	*F.val* = digit.*lexval*

SLR分析栈中实现计算器

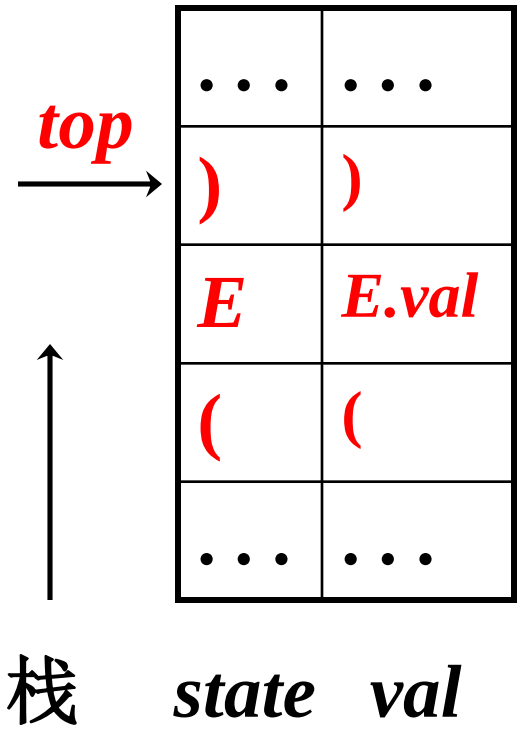
❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*val* [*top*–1])
$E \rightarrow E_1 + T$	*val* [*top* –2] = *val* [*top* –2]+*val* [*top*]
$E \rightarrow T$	值不变，无动作
$T \rightarrow T_1 * F$	*val* [*top* –2] = *val* [*top* –2]×*val* [*top*]
$T \rightarrow F$	值不变，无动作
$F \rightarrow (E)$	*F.val* = *E.val*
$F \rightarrow$ digit	*F.val* = digit.*lexval*

SLR分析栈中实现计算器

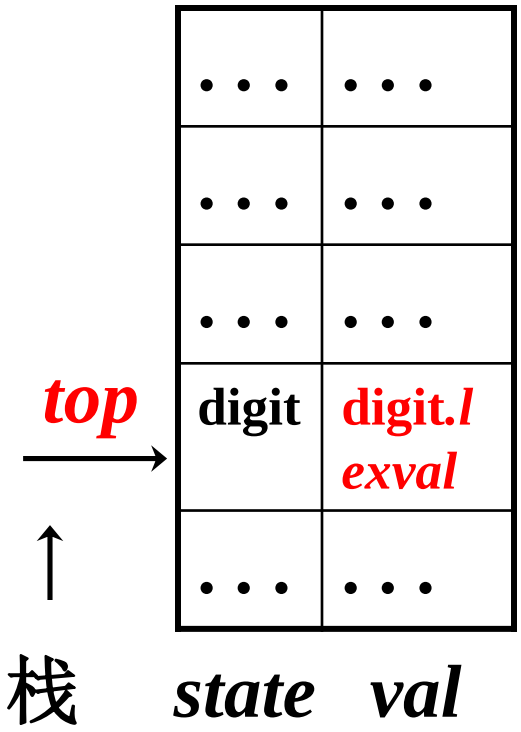
❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*val* [*top*–1])
$E \rightarrow E_1 + T$	*val* [*top* –2] = *val* [*top* –2]+*val* [*top*]
$E \rightarrow T$	值不变，无动作
$T \rightarrow T_1 * F$	*val* [*top* –2] = *val* [*top* –2]×*val* [*top*]
$T \rightarrow F$	值不变，无动作
$F \rightarrow (E)$	*val* [top –2] =*val* [top –1]
$F \rightarrow$ digit	*F.val* = digit.*lexval*

SLR分析栈中实现计算器

❑简单计算器的语法制导定义改成栈操作代码



产 生 式	代 码 段
$L \rightarrow E$ n	*print* (*val* [*top*–1])
$E \rightarrow E_1 + T$	*val* [*top* –2] = *val* [*top* –2]+*val* [*top*]
$E \rightarrow T$	值不变，无动作
$T \rightarrow T_1 * F$	*val* [*top* –2] = *val* [*top* –2]×*val* [*top*]
$T \rightarrow F$	值不变，无动作
$F \rightarrow (E)$	*val* [top –2] =*val* [top –1]
$F \rightarrow$ digit	值不变，无动作

重要知识点复习思路

中国科学技术大学
University of Science and Technology of China

❑类型检查

- ❖掌握类型表达式书写
  - ➢指针、数组、结构体、函数等
- ❖掌握使用语法制导翻译进行类型检查

类型表达式 (Type expression)

中国科学技术大学
University of Science and Technology of China

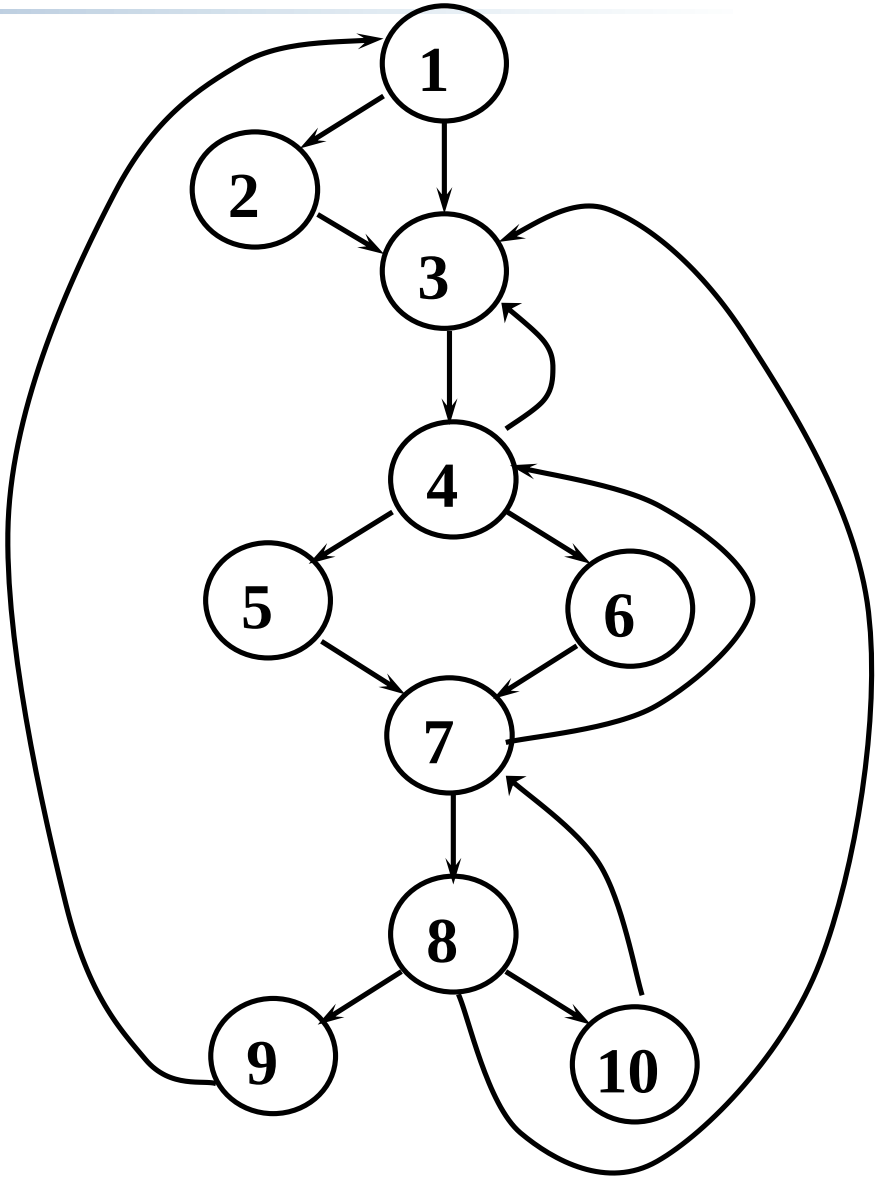
- 基本类型是类型表达式
- 可为类型表达式命名，类名也是类型表达式
- 将类型构造算子(type constructor)作用于类型表达式可以构成新的类型表达式
  - 数组类型构造算子*array*
  - 指针类型构造算子*pointer*
  - 笛卡尔乘积类型构造算子×
  - 函数类型构造算子→
  - 记录类型构造算子*record*

重要知识点复习思路

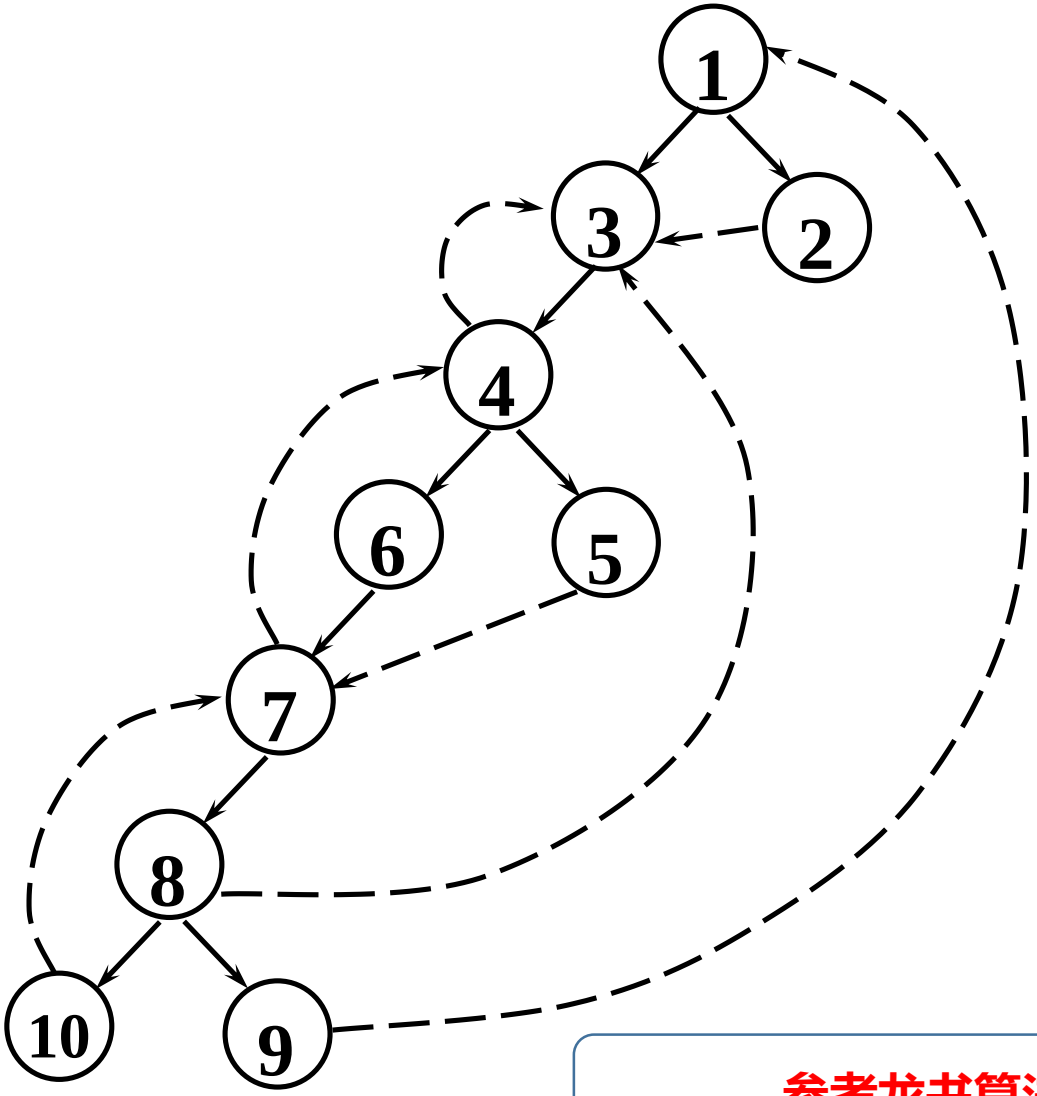
- 中间代码
  - 掌握三地址码格式
  - 为简单的高级语言程序写三地址代码
    - 表达式、数组访问、布尔表达式等
  - 掌握基本块、流图、循环
    - 给定三地址码，如何划分基本块、画出流图、找出循环、计算回边等
    - 参考ppt和助教习题课

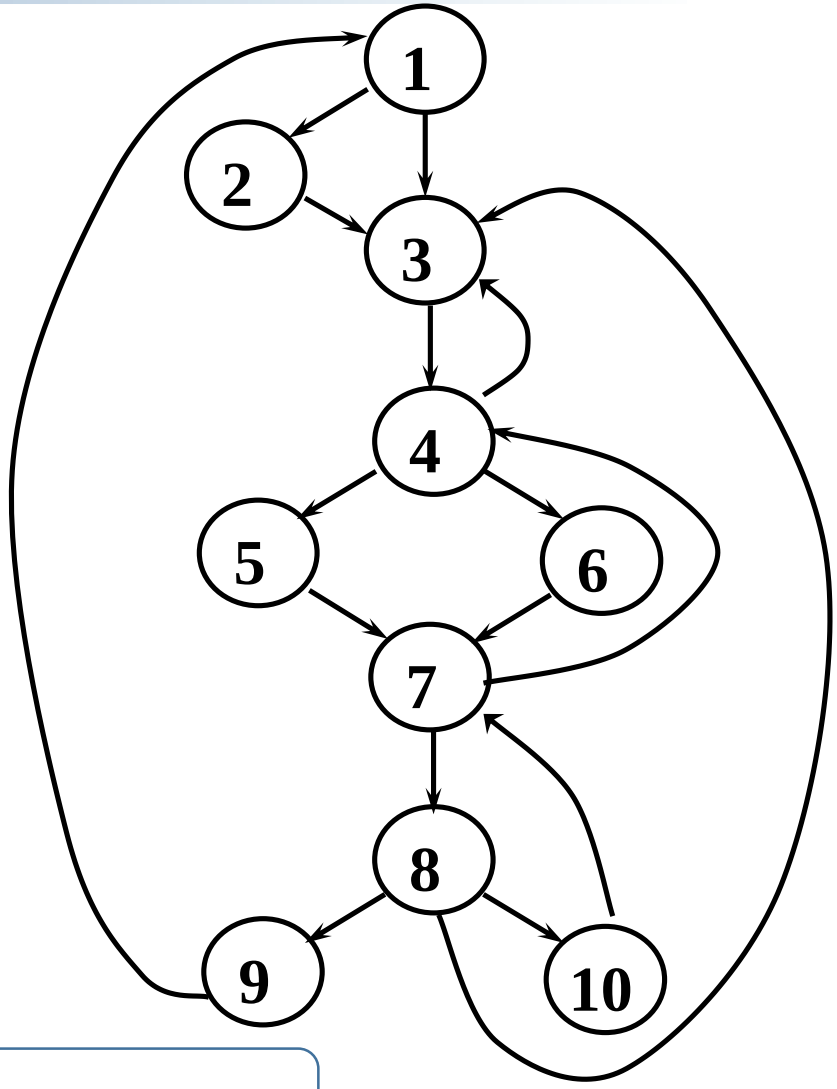
深度优先排序

- 后序遍历(先右孩子)
  - 10,9,8,7,6,5,4,3,2,1
- 深度优先排序正好与后序遍历相反
  - 1,2,3,4,5,6,7,8,9,10



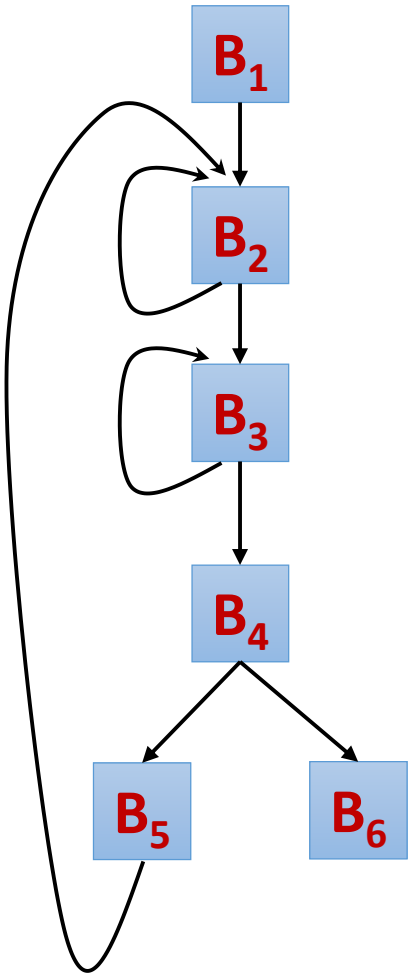
深度优先表示





参考龙书算法9.41

- 流图中的一个结点集合L是一个循环，如果它满足：
  - 该集合有唯一的入口结点
  - 任意结点都有一个到达入口结点的非空路径，且该路径全部在L中
- 不包含其他循环的循环叫做内循环
- 右图中的循环
  - B_2 自身
  - B_3 自身
  - $\{B_2, B_3, B_4, B_5\}$



❑自然循环的性质

- ❖ 有唯一的入口结点，叫做首结点，首结点支配该循环中所有结点
- ❖ 至少存在一条回边进入该循环首结点

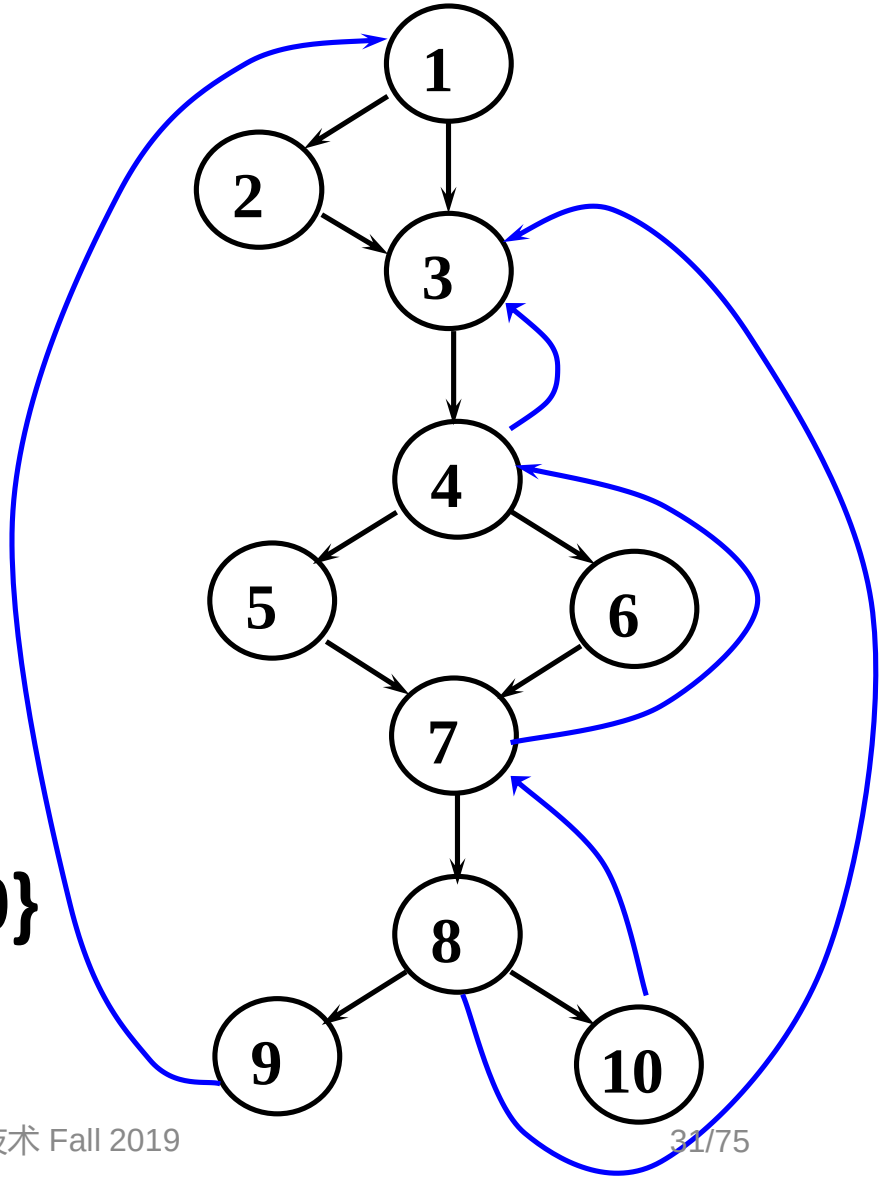
❑回边$n \rightarrow d$确定的自然循环

- ❖*d*加上不经过*d*能到达*n*的所有结点
- ❖ 结点*d*是该循环的首结点

自然循环

- 回边10 → 7
  - 循环{7, 8, 10}
- 回边7 → 4
  - 循环{4, 5, 6, 7, 8, 10}
- 回边4 → 3和8 → 3
  - 循环{3, 4, 5, 6, 7, 8, 10}
- 回边9 → 1

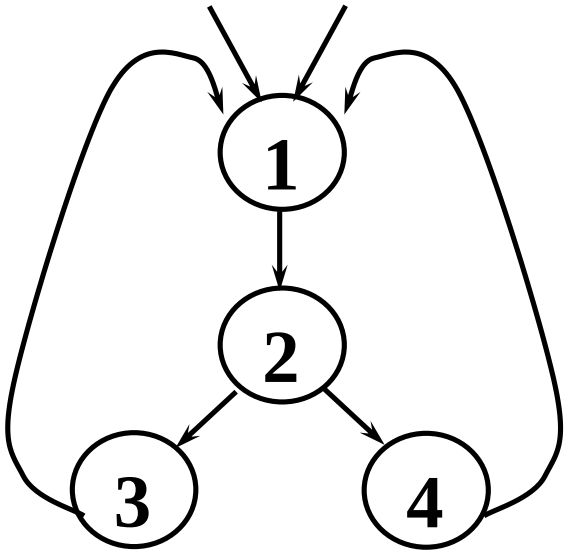
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}



参考龙书算法9.46

自然循环的合并

- 两个循环有相同的首结点，但并非一个结点集是另一个的子集，则看成一个循环



❑机器无关优化

- ❖理解相关优化的含义
  - ➢公共子表达式删除、复制传播、 常量合并、死代码删除、代码移动、强度削弱、删除归纳变量
- ❖给定三地址码，给出优化意见
  - ➢如何识别优化的可能：循环不变量等

- 运行时与代码生成
  - 掌握活动记录
  - 掌握存储栈式分配
    - 结合C语言例子
      - 掌握C源程序，汇编代码，活动记录三者内在联系
  - 掌握简单的汇编代码生成
    - 给定三地址码，生成汇编代码
    - 掌握寄存器分配，地址和寄存器描述符

最后的注意事项

- 利用好issue来复习
- 温习好课后习题
- 温习做过的实验，自己写的report

中国科学技术大学
University of Science and Technology of China

USTC, CHINA
ADSLAB
先进数据系统实验室

《编译原理与技术》
期末复习

The real end & Have fun!