2024年西安交通大学
形式语言与编译

绪论

- 《形式语言与编译》是本校自行设计的课程，对标《编译原理》，强调形式语言理论及其在回答编译原理中诸多为什么之问题的作用，有助于化难为易、启发思维这种学习过程中发生的理想状况之呈现。

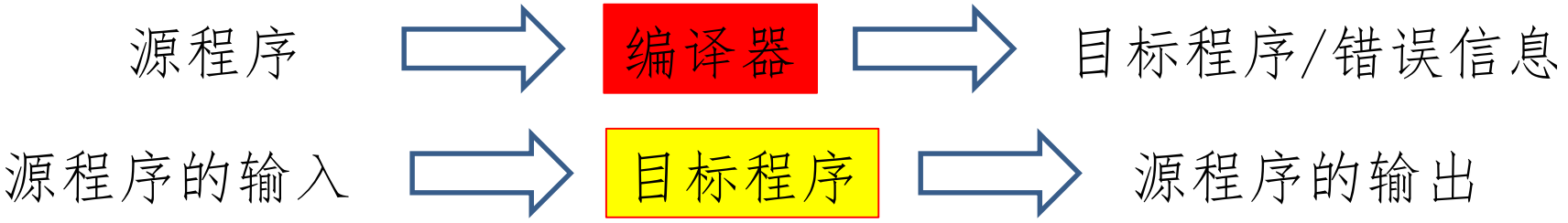
- 从三个角度开始了解本章内容：
  - 什么是编译？
  - 为什么学编译？
  - 本课程是什么？

问题1：什么是编译？

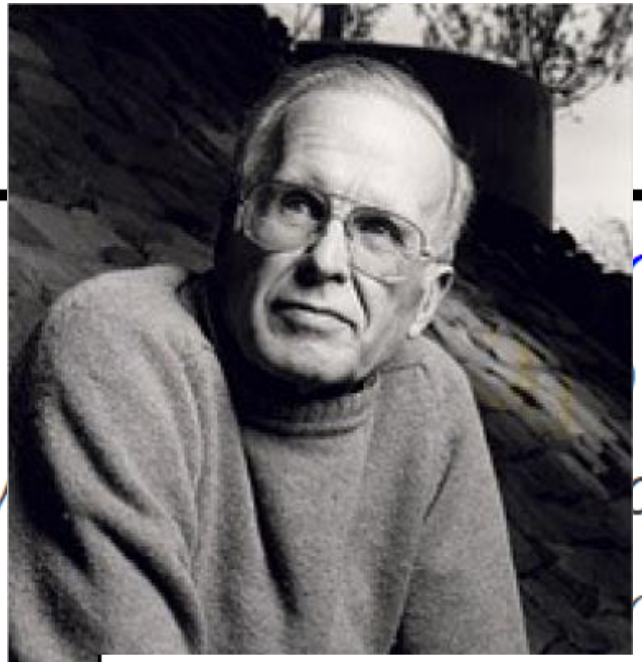
- 所谓编译是指程序（语言处理器）：编译程序（编译器）、编译系统、解释程序（解释器）。
- 所谓编译是指软件框架：前端（多源语言）、后端（多目标机）、中间表示、自动生成工具（词法及语法分析器生成器、代码生成器的生成器）、多优化遍集成等。
- 所谓编译是程序转换过程（含分析与综合）：五阶段编译过程、编译遍等。
- 所谓编译是一套理论和方法：2型和3型形式语言理论（多语言联合DFA、itemPDA等）、LL(k)和LR(k)分析法、属性文法与语法制导翻译、运行时环境、代码优化、流分析、寄存器分配等。

编译程序

- 编译程序是一个程序，该程序将用某个语言写的程序（称为源程序）作为输入，并翻译成为功能上等价的另一个语言程序（称为目标程序）。
- 用来写源程序所用的语言，即源语言，通常是高级语言，如C，FORTRAN等。
- 用来写目标程序所用的语言，即目标语言，通常是低级语言如汇编或机器语言。
- 随着翻译过程的进行，编译程序也报告错误信息以帮助程序员改错，直到翻译通过为止。



高级语言



race

r of

d the

r."

John Backus,
team lead on
FORTRAN.

Image: http://upload.wikimedia.org/wikipedia/commons/thumb/5/55/Grace_Hopper.jpg/300px-Grace_Hopper.jpg
http://www.nytimes.com/2007/03/20/business/20backus.html

- 科技创新与瞄准国家目标

分层认识编程语言

ELF/COFF

```
#as globe var
.data
msg:   .ascii "Hello World"

#as main
.text
la $a0 msg
li $v0 4
syscall
```

```
1  #include <stdio.h>
2
3  main() {
4    printf("hello world\n");
5  }
```

- 低级语言（硬件直接执行）：指令、地址、RAM；
- 高级语言（命令式与陈述式等）（名字、值）。
- 高级语言数千种每年还有近百种被发明。
- 对编译器的影响：要求适应新语言、利用新硬件、提升性能

编译器示意

目标语言是高级/低级语言时？

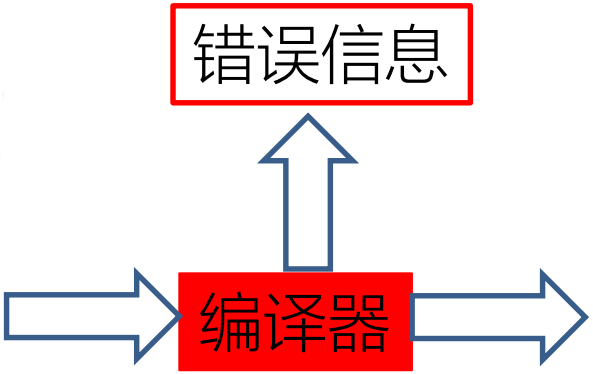
实现语言是高级/低级语言时？

目标机与宿主机相异时？

交叉编译器

```
1  #include <stdio.h>
2
3  main() {
4    printf("hello world\n");
5  }
```

源程序



源语言

实现语言

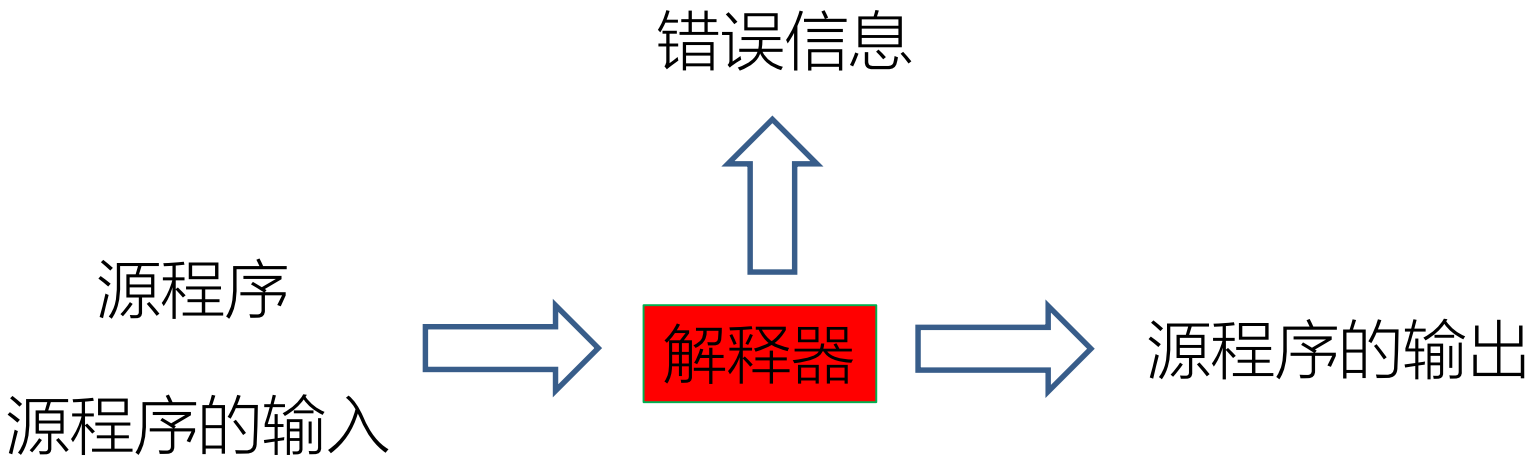
目标程序

```
#as globe var
.data
msg:   .ascii "Hello World"

#as main
.text
la $a0 msg
li $v0 4
syscall
```

目标语言

解释器



解释器和编译器特点

- 概念上的不同（翻译/直接执行）
- 基于解释执行的程序可以动态修改自身，而基于编译执行的程序动态性较弱
- 基于解释方式便于调试
- 基于解释方式有利于人机交互
- 基于解释执行的速度较慢
- 解释器需要保存的信息较多，空间开销大
- 二者实现技术相似

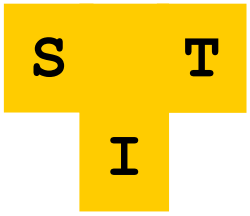
编译过程与编译程序

源语言　　实现语言　　目标语言

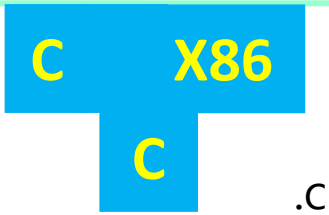
无　　宿主机器　　目标机器

✓ 涉及到三个程序，各程序是用什么语言写的？运行的机器是什么？

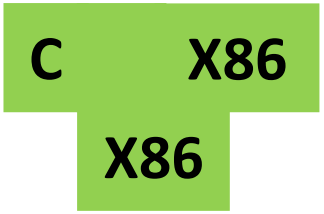
用T型图表示移植编译器的过程



- S – Source Language
- T – Target Language
- I – Implementation Language

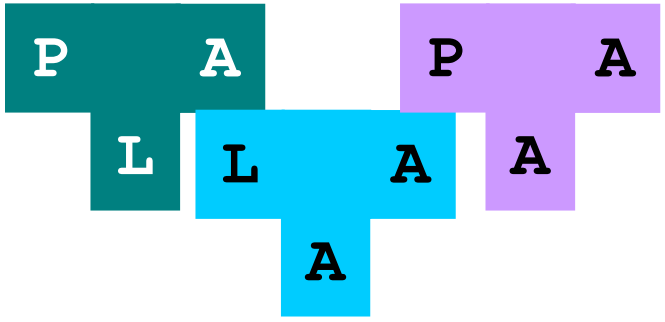


.c

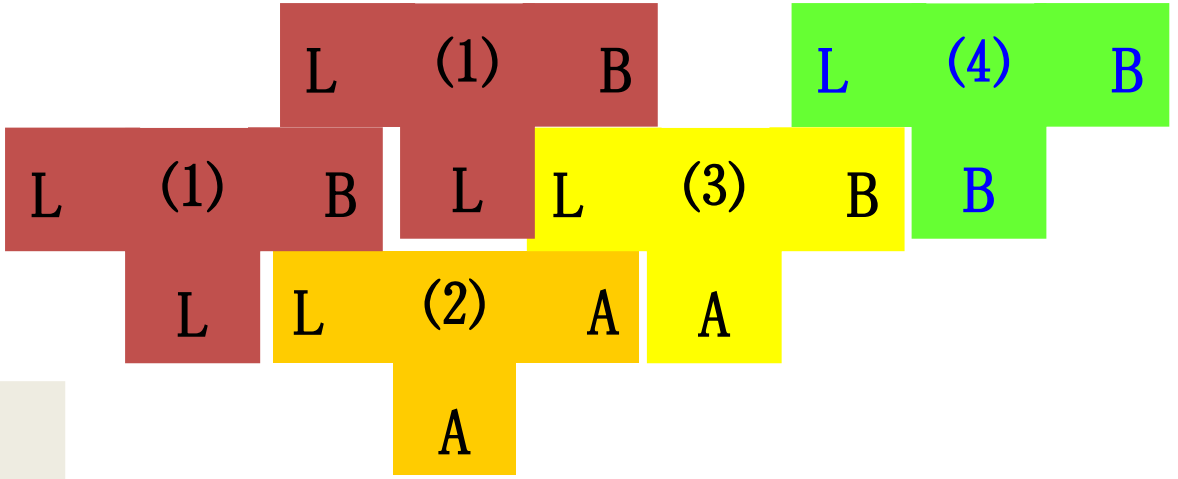


.o

用I语言写的编译器被用于将S语言写的源程序翻译成用T语言写的目标程序



给定用L写的P语言编译器，使用L语言的A机器编译器进行编译，得到A机器的P语言编译器



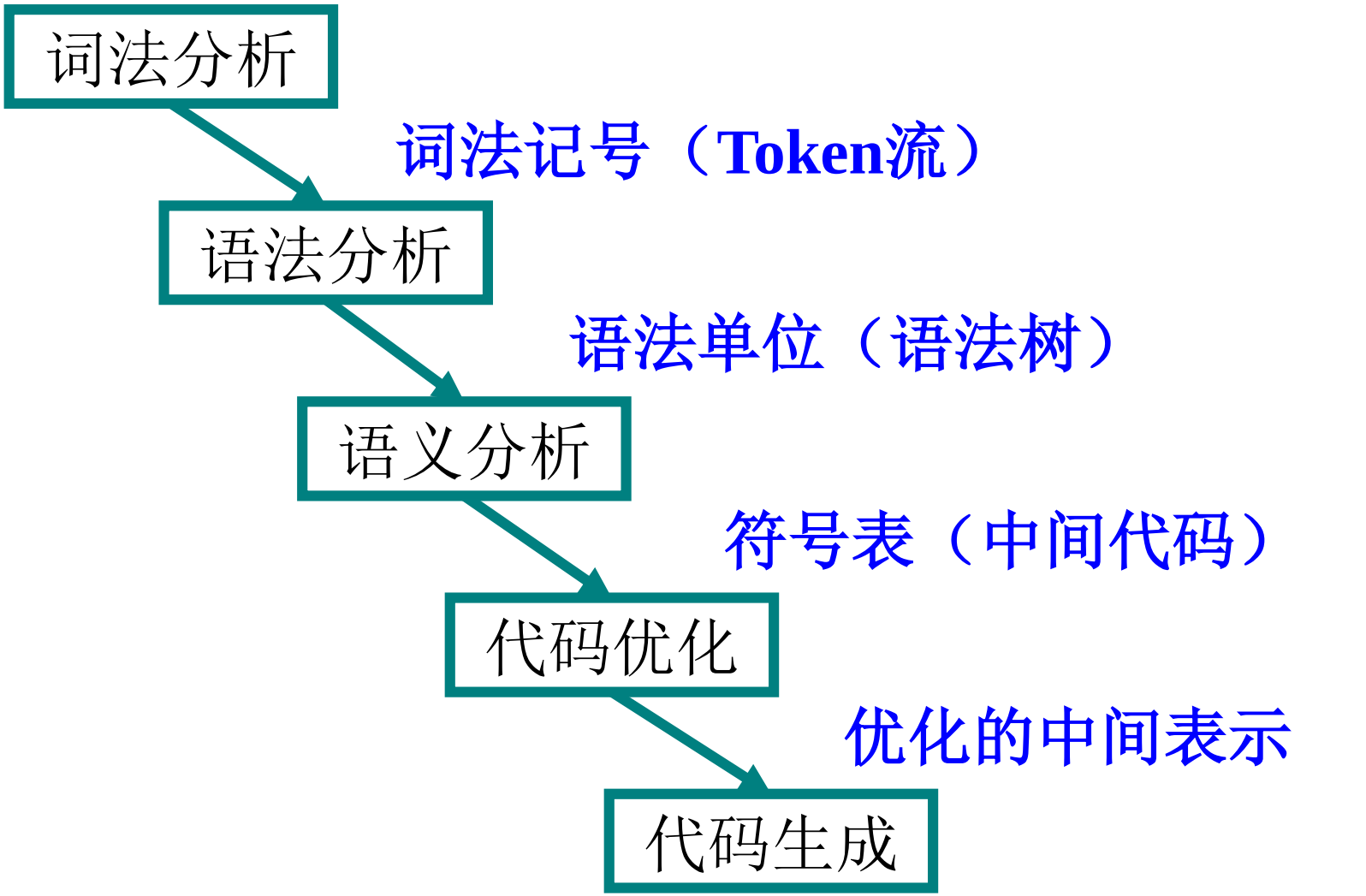
将编译器(1)利用编译器(2)移植为编译器(4)

编译器的种类

- **诊断编译程序**
  - 侧重于错误信息，帮助开发调试
- **优化编译程序**
  - 侧重于提高代码效率
- **交叉编译程序**
  - 一般为嵌入式系统生成可执行代码
- **可重定向编译程序**
  - 仅定制编译器中跟机器有关部分实现针对新目标机的代码生成

编译过程

源程序（字符流）


词法分析
词法记号（Token流）
语法分析
语法单位（语法树）
语义分析
符号表（中间代码）
代码优化
优化的中间表示
代码生成
目标程序（可执行代码）

Step1 词法分析

```
int x;
int fact(int n; int a;){
    if (n==1) return a else return fact (n-1, n*a,)
};
x=123+fact(5,1,);
print x
```

- int fact(int n; int a;){\n\tif (n==1) return a else return fact(n-1, n*a,)\n};

i	n	t		f	a	c	t	(	i	n	t		n	;		i	n	t		a	;	)	{	↲	→	i	f	(	n	=	=	1	)	r	e

词法分析结果：词法记号串

- int fact(int n; int a;){\n\tif (n==1) return a else return fact(n-1, n*a,)\n};

- (INT, _) (ID, fact) (LPA, _) (INT, _) (ID, n) (SCO, _) (INT, _) (ID, a) (SCO, _) (RPA, _) (BLA, _) (IF, _) (LPA, _) (ID, n) (ROP, ==) (NUM,1) (RPA, _) (RETURN, _) (ID, a) (ELSE, _) (RETURN, _) (ID, fact) (LPA, _) (ID, n) (AOP, -) (NUM,1) (CMA, _) (ID, n) (AOP, *) (ID, a) (CMA, _) (RPA, _) (BRA, _) (SCO, _)

词法记号的种类示意

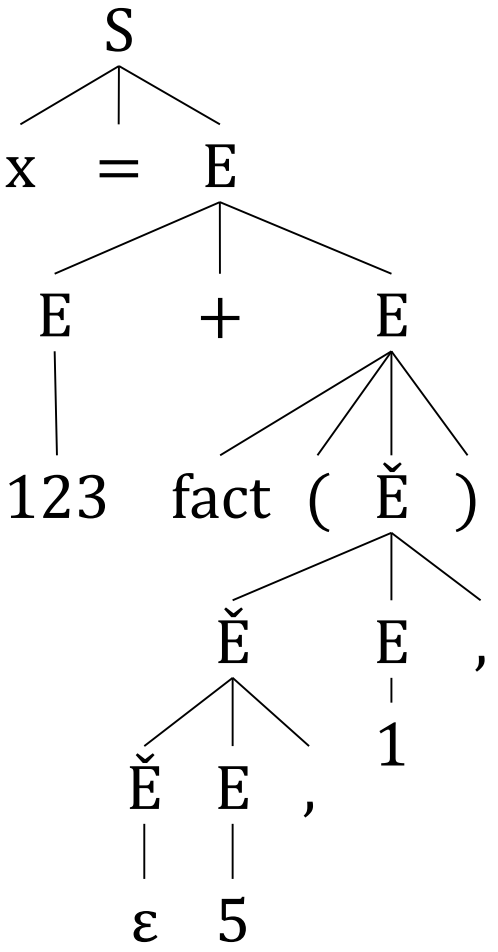
- 常数：NUM，FLO
- 标识符：用户定义的名字：ID
- 关键字/保留字：系统提供的名字：INT，IF，ELSE, RETURN
- 运算符：AOP，ROP
- 分隔符：LPA，SCO，RPA，BLA，CMA，BRA

- 词法错误?
- x=12..3+fa#ct(5; 1);
- 正则语言识别器

Step2 语法分析

- 依据语言的语法规则，将词法记号序列转化为一些语法单位（短语、子句、句子、程序段、程序）及其之间的关系，并确定整个输入串是否构成一个语法上正确的程序。

x=123+fact(5,1,)



列表1-1的78~98

语法分析发现语法错误

```
int * foo(i, j, k))
   int i;
   int j;
{
   for(i=0; i j) {
   fi(i>j)
   return j;
}
```

多余的括号

缺少步长

不是保留字

不是表达式

Step3 语义分析

- 对语法分析所识别出的各类语法范畴，分析其含义。
- 从程序声明部分提取语义信息登记在符号表。
- 对程序语句部分生成中间代码。
- 发现语法错误。

属性文法、语法制导的语义分析

fact程序的声明部分与语句部分

```
int x;
int fact(int n; int a;){
    if (n==1) return a else return fact (n-1, n*a,)};
x=123+fact(5,1,);
print x
```

声明2个函数、1个全局变量x、2个形参变量

语句：最外层函数的2个语句、fact函数的if语句

```
@table: (
    outer: NULL width: 28 argc: 0 arglist: NIL rtype: INT level: 0
    code: [t6=123; t7=5; t8=1; PAR t7; PAR t8; t9=CALL fact, 2; t10=t6+t9; x=t10; PRINT x]
    entry: (name: x type: INT offset:4)
    entry: (name: fact type: FUNC offset: 12 mytab: fact@table)
    entry: (name: t6 type: INT offset: 16) entry: (name: t7 type: INT offset: 20)
    entry: (name: t8 type: INT offset: 24) entry: (name: t9 type: INT offset: 28))
fact@table: (
    outer: @table width: 30 argc: 2 arglist: (n a) rtype: INT level: 1
    code: [t1=1; IF n<=t1 THEN l1 ELSE l2; LABLE l1; RETURN a; GOTO l3; LABEL l2;
    t2=1; t3=n-t2; t4=n*a; PAR t3; PAR t4; t5=CALL fact, 2; RETURN t5; LABEL l3]
    entry: (name: n type: INT offset: 4) entry: (name: a type: INT offset: 8)
    entry: (name: t1 type: INT offset: 12) entry: (name: t2 type: INT offset: 16)
    entry: (name: t3 type: INT offset: 20) entry: (name: t4 type: INT offset: 24)
    entry: (name: t5 type: INT offset: 30))
```

语义分析发现语义错误

```
int * foo(i, j, k)
   int i;
   int j;
{
   int x;
   x = x + j + N;
   return j;
}
```

类型未说明

变量未说明

返回结果类型不匹配

使用了没有初始化的变量

Step4优化

int fact(int n; int a;){if (n==1) return a else return fact(n-1, n*a,)};

- 消除尾递归

[t1=1; IF n=t1 THEN l1 ELSE l2; LABLE l1; RETURN a; GOTO l3; LABEL l2; t2=1; t3=n-t2; t4=n*a; PAR t3; PAR t4; t5=CALL fact, 2; RETURN t5; LABEL l3]

[LABEL l0; t1=1; IF n=t1 THEN l1 ELSE l2; LABLE l1; RETURN a; GOTO l3; LABEL l2; t2=1; t3=n-t2; t4=n*a; ~~PAR t3; PAR t4; t5=CALL fact, 2; RETURN t5~~ n=t3; a=t4; GOTO l0; LABEL l3]

Step5 可执行代码

- 运行时环境
- 寄存器分配、指令选择等等。

编译程序的错误处理

- 词法错误　不符合词法规则的错误。
  单词构成有错
- 语法错误　不符合语法规则的错误。
  括号不匹配、缺失语法单位、语法单位间的错误关系等
- 语义错误　不符合语义规则的错误。声明错误、
  作用域错误、类型不匹配等

- 全　　最大限度发现错误
- 准　　准确指出错误的性质和发生地点
- 局部化　将错误的影响限制在尽可能小的范围内

编译程序的符号表管理

- 符号表的作用和种类
  - 保存编译各阶段的有用信息
  - 名字表（变量名、过程名、标号名、保留字）
  - 常数表
  - 名字表也可匀质拆分（不是关键）
  - 符号表默认为符号表
- 符号表的操作
  - 创建函数符号表、添加和查找表的登记项、修改表项信息
  - 关注方便性和时空效率

一遍与多遍编译

- 是对源程序或编译中间结果从头到尾扫描一次，并做有关的加工处理，生成新的中间结果或目标程序。

- 注意各遍间的中间结果（内部形式、外部形式）

One-pass compiler: Type of software compiler that passes through the source code only once. One-pass compilers are faster, but may not generate an as efficient program. In addition, one-pass compilers cannot compile all types of source codes.
http://www.computerhope.com/jargon/o/onepassc.htm

编译前端与后端

- 前端指跟目标机无关的部分，是面向源语言的；
- 后端指跟目标机有关的部分，是面向目标机的；
- 前后端之间的接口：
  - 中间表示
  - 符号表

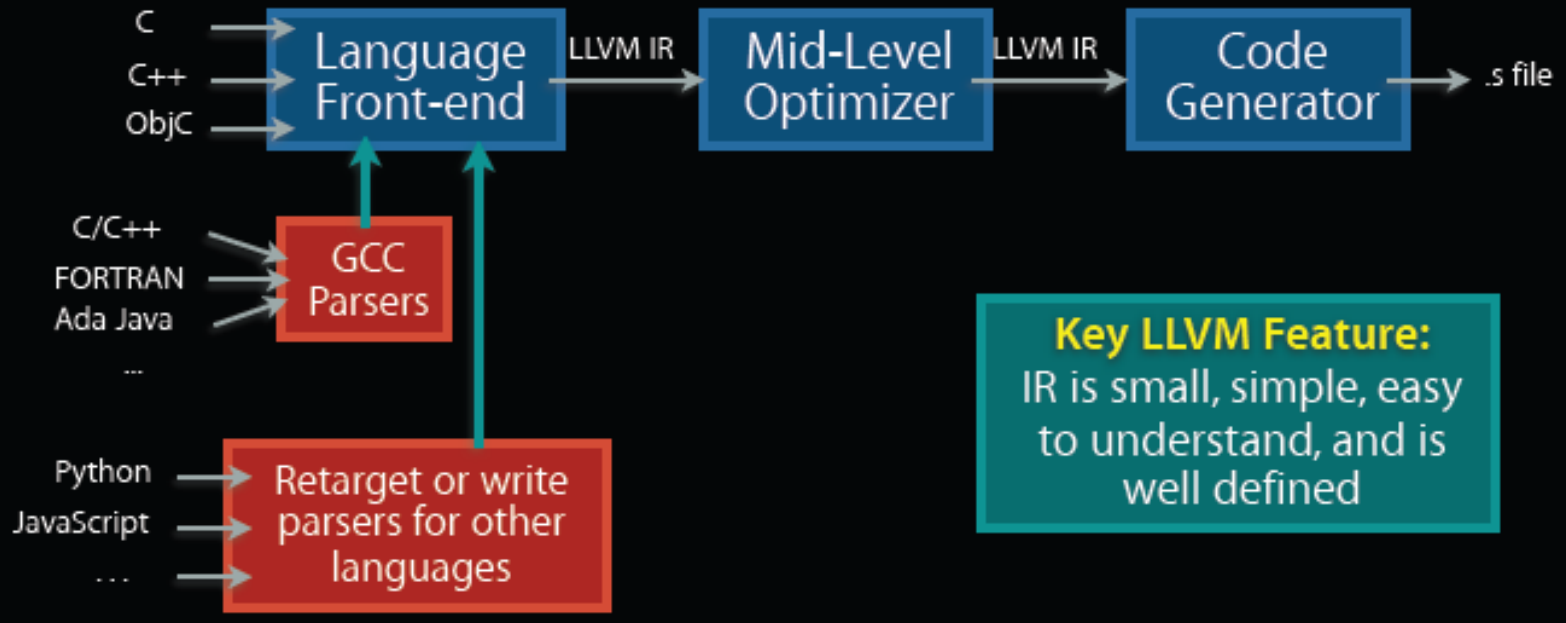
Example of a Simple Static Compiler

- Standard compiler organization, which uses LLVM as midlevel IR:
  - Language specific front-end lowers code to LLVM IR
  - Language/target independent optimizers improve code
  - Code generator converts LLVM code to target (e.g. IA64) code

Many compilers (e.g. GCC) follow this model.

Front-end options for this compiler

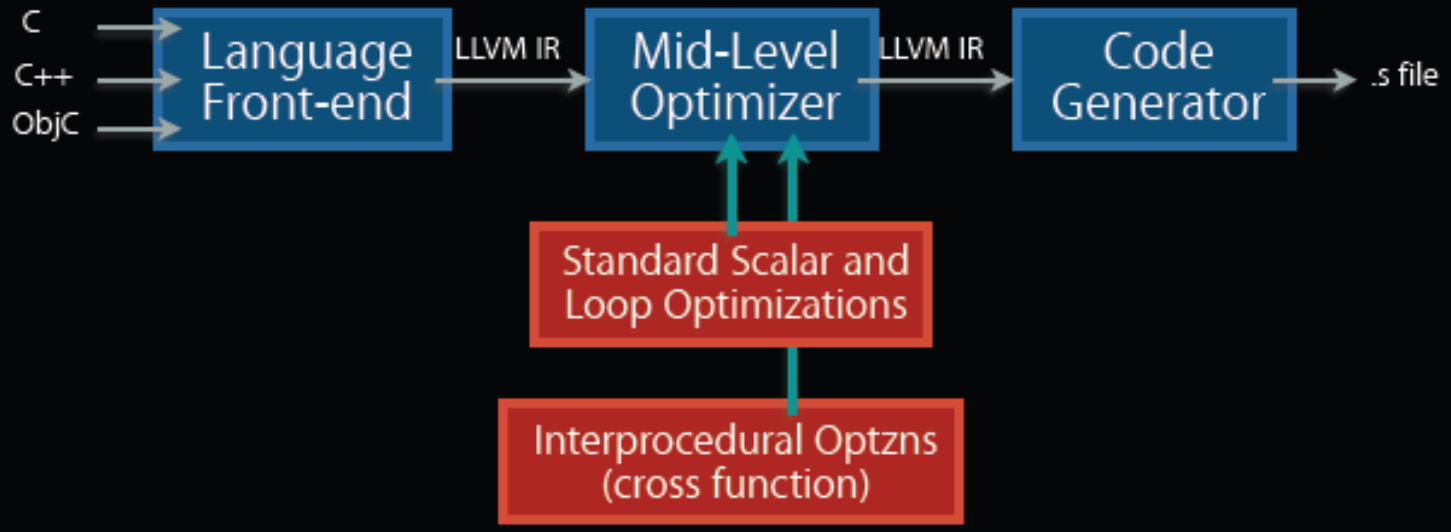
- Front-ends are truly separate from optimizer & codegen
  - Can use front-end AST's that are tailored to the source language
  - Optimizer & Codegen improvements benefit all front-ends
  - Front-ends generate debug info and include it in the IR



llvm-gcc currently uses the GCC 4.0.1 parsers

Optimizer options for this compiler

- Optimizer is solely concerned with semantics of LLVM IR
  - Optimizer & Codegen only know LLVM, not all source languages
  - LLVM includes IP framework and aggressive IP optimizations
  - LLVM uses a modern and light-weight (fast) SSA-based optimizer



编译器构造工具

- 语法分析器的生成器 YACC BISON
- 词法扫描器的生成器 LEX FLEX
- 语法制导的翻译引擎
- 代码生成器的生成器
- 数据流分析引擎
- 编译器构造工具集如Java编译器的编译器JastAdd

编译技术的应用

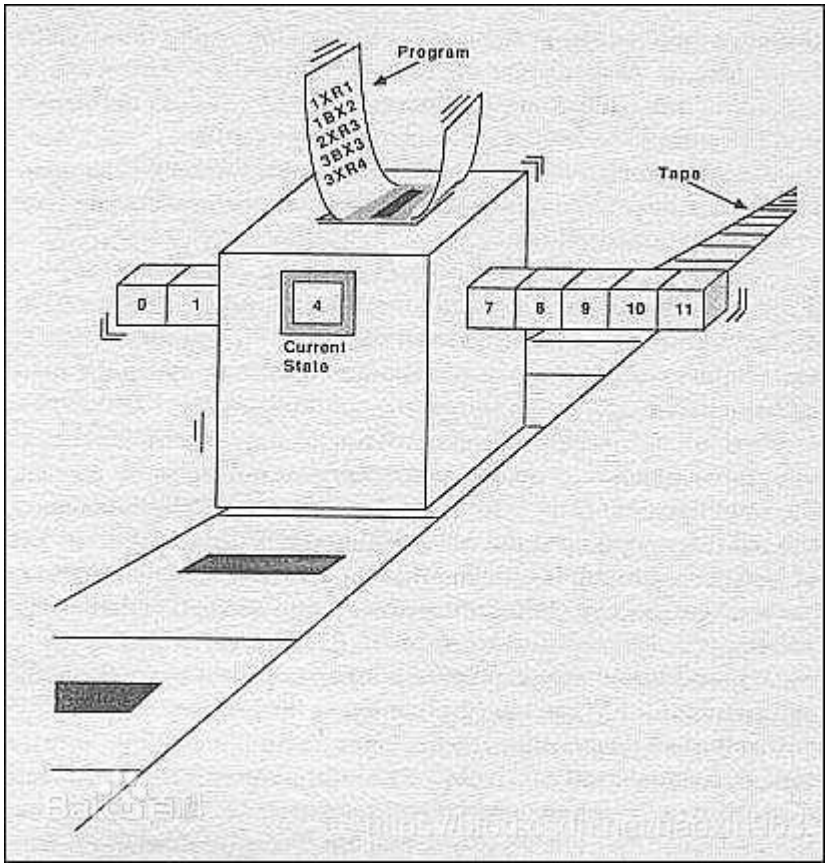
- 高级程序设计语言的实现
  - 语言特征与编译优化
  - 动态编译
- 针对计算机体系结构的优化
  - 并行性
  - 内存层次结构
- 新计算机体系结构的设计：RISC、专用体系结构
- 程序翻译：二进制翻译、硬件合成、数据查询解释器、编译然后模拟
- 软件生产率工具：类型检查、边界检查、内存管理工具

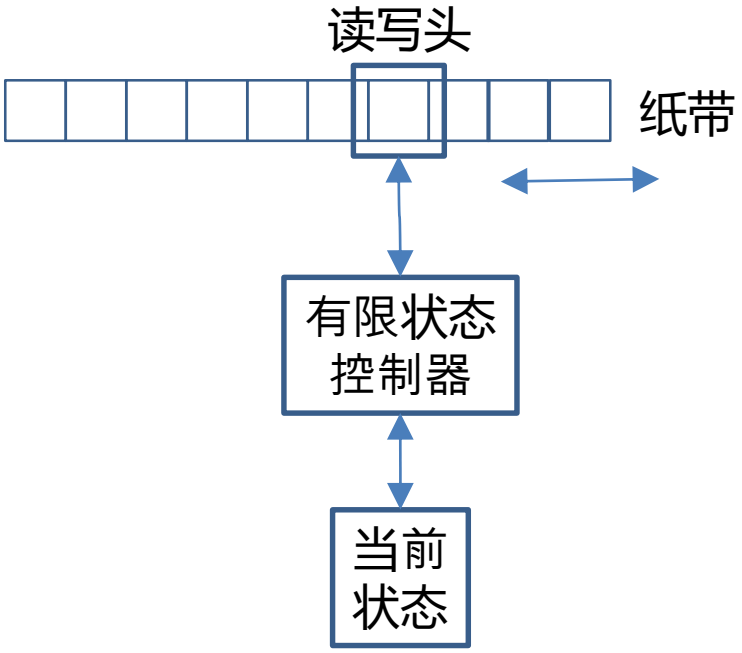
问题2：为什么学编译？

- 为了掌握高级语言翻译为机器语言的原理和方法？
- 为了会编写编译器？
- 为了对程序有更深的理解？

从图灵机认识抽象计算模型

图灵机是由英国数学家A.M.图灵1936年提出的一种抽象计算模型，即将人们使用纸笔进行数学运算的过程进行抽象，由一个虚拟的机器替代人们进行数学运算。





初始状态；
接受状态；
当前状态；
状态转换规则集。

图灵机例子

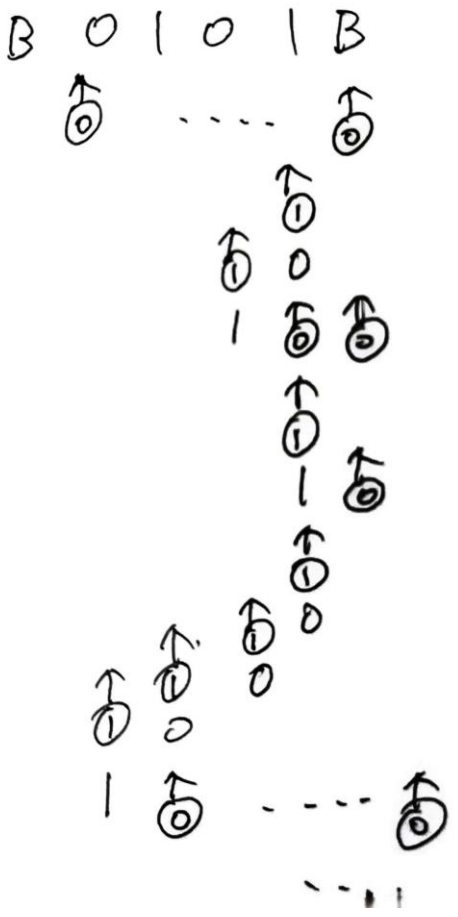
- (State Number, Symbol Read) -> (Next State Number, Symbol To Write) Next Cell

- Symbol Read：(0, 1, 空格)
- (1,0) -> (0,1) Left
- 若机器处于状态 1且单元内容为0，则修改为1、状态转为0、向左移动一个单元
- (1,1) -> (1,0) Left　在状态1若单元内为1则改为0、停留状态1并向左移动一格
- (1,B) -> (0,1) Right　若单元内为空白改为1、状态转为0并右移动一格
- (1,B) -> (0,1) Halt　停机指令，若当前单元为空白则改为1状态转0并停机

计数器图灵机示例

```
(0, 1) -> (0, 1) Right
(0, 0) -> (0, 0) Right
(0, B) -> (1, B) Left

(1, 0) -> (0, 1) Right
(1, 1) -> (1, 0) Left
(1, B) -> (0, 1) Right
```



无符号数加1图灵机：第4、第6条规则中的Right改为Halt，其它不变。（接受状态隐含停机）

通用图灵机

- In computer science, a universal Turing machine (UTM) is a Turing machine that can simulate an arbitrary Turing machine on arbitrary input. The universal machine essentially achieves this by reading both the description of the machine to be simulated as well as the input thereof from its own tape. Alan Turing introduced this machine in 1936–1937. This model is considered by some (for example, Martin Davis (2000)) to be the origin of the stored program computer—used by John von Neumann (1946) for the "Electronic Computing Instrument" that now bears von Neumann's name: the von Neumann architecture. It is also known as universal computing machine, universal machine (UM), machine U, U.

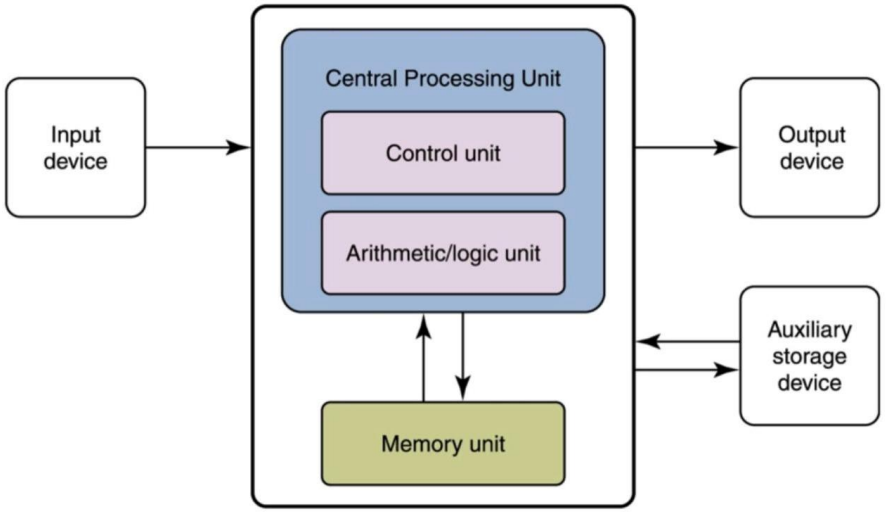
具体地说，如果把任意一个 Turing 机的指令集用 Turing 自己提出的一种规范方式编码并预存在纸条上，那么通用 Turing 机就能够根据纸条上已有的信息，在纸条的空白处模拟那台 Turing 机的运作，输出那台 Turing 机应该输出的东西。

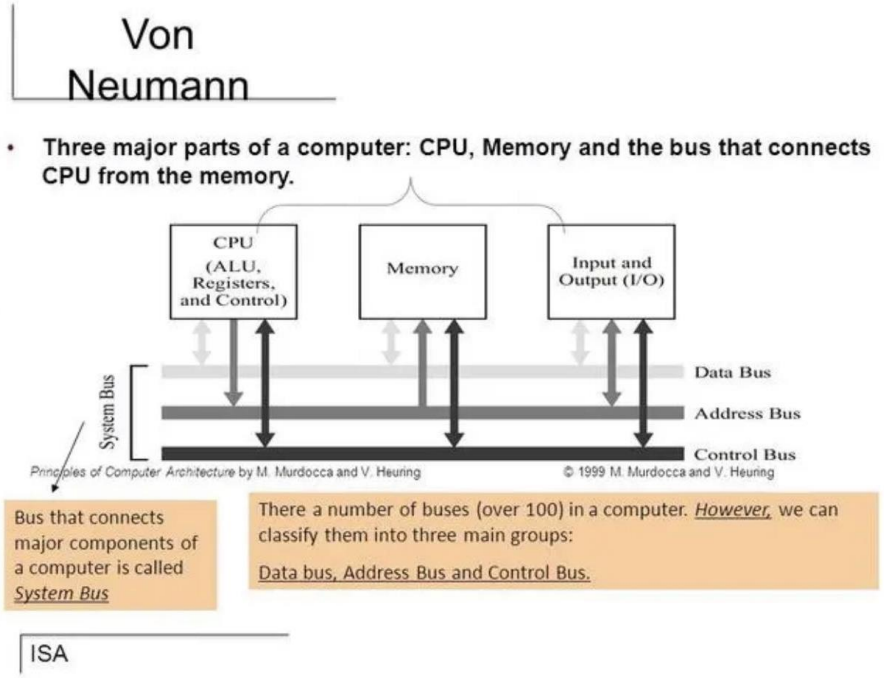
ENIAC 电子数字积分与算术计算1946

UNIVAC 通用自动计算机1951

冯诺依曼结构

- 采用二进制形式表示数据和指令
- 采用存储程序方式





意味着事先编制程序，事先将程序(包含指令和数据)存入内存，运行时自动地、连续地从存储器中依次取出指令且执行。这是计算机能高速自动运行的基础。计算机的工作体现为执行程序，计算机功能的扩展在很大程度上也体现为所存储程序的扩展。计算机的许多具体工作方式也是由此派生的.

编程语言用于构建计算系统

- 计算系统的基础是图灵机和冯诺依曼体系结构；
  - 图灵机▶通用图灵机▶冯诺依曼结构
- 编程语言是专门设计用来表达计算过程的语言，分层适应于计算系统的构建；
- 编译程序的作用就是降低计算系统构建的复杂度。

问题3：本课程是什么？

- 可参看思源学堂课程介绍。

课程名称的由来

- 2016-2017第二学期起采用现名《形式语言与编译》，之前为《编译原理》。
- 该名称为本校自提，其合理性在于编程语言是唯一有学科代表性的形式语言
- 该课程的设计目标是在兼顾课程体系、课程内容的基础上强调编译相关理论、原理和方法，而将代码级实现技术作为专题实验课与之配套来实现。
- 该课程的教学设计主要采用关注知识等价性的教学方法论。
  - [1]赵银亮. 关注编译原理的等价性[J]. 计算机教育,2011(11).
  - [2]赵银亮,戴慧珺,等. 谈谈形式语言与编译课程[J]. 计算机教育,2018(4).
  - [3]赵银亮. 浅谈编译原理课程的定位[J]. 西安邮电学院学报,2010(4).
- 本课程特色：保持形式语言理论完整性；强调编译的理论性和方法论；具有化难为易的效果。配套实验课采用了国际一流大学的编译实验。
- 完成同名教材。

课程简介

- 介绍基本的形式语言与自动机理论、编译前端基本原理和构建方法。
- 内容包括基于正则语言、有穷自动机理论的词法分析方法，基于上下文无关语言、下推自动机理论的语法分析方法，基于属性文法的高级语言语句到中间表示的翻译方法，基于栈的面向过程语言的运行时环境分析与设计。
- 本课程兼顾形式语言理论完整性传授的同时，兼顾词法、语法、语义分析方法的典型性，并联系到运行时环境、寄存器分配等高级技术。
- 本课程与不可或缺的编译器设计专题实验课提供了同学理论层面和实现层面完整知识。
- 参加本课程的学习有助于提高抽象思维能力、编程动手能力、以及形式化表达能力。

课程目标及同学应达到的能力

- 掌握形式语言理论、基本概念和基本方法，包括正则语言、有穷自动机、正则表达式、上下文无关语言、上下文无关文法、下推自动机等，综合应用于形式语言有关判定与分析。
- 掌握程序设计语言的编译原理和主要方法，包括词法分析、语法分析和语法制导的语义分析方法，中间代码生成方法及运行时环境设计，能够解决有关编译器前端设计方面的基本问题。
- 表现在对所学理论和方法达到了一定程度的融汇贯通，对于所学知识点取得一定的举一反三效果。

课程结构

- 正则语言与词法分析
  - 有穷自动机、正则表达式与正则语言
  - 多语言联合DFA与词法分析原理
  - 词法分析器的设计与生成
- 上下文无关语言与语法分析
  - 上下文无关文法与自上而下语法分析
  - 下推自动机与自下而上语法分析
  - 语法分析原理与分析器的设计、生成
- 语法制导的语义分析
  - 属性文法与SLR(1)引导的语义分析
  - 符号表、中间表示、中间代码生成
  - 运行时环境、可执行代码生成

课程知识点汇总

Listing4.1：课程知识结构$\mathcal{K}=\{\cdot\}$

C 形式语言{

中心概念{符号,字母表,符号串,语言,问题,图灵机},

RL{121 识别器{DFA[11],NFA[11],RE[11]},122 判定性{$\tilde{\upsilon}_D$[1211],$\tilde{\upsilon}_N$[1212],匹配[1213]},

123 等价性[121-2]{NFA~DFA,DFA 最小化,RE~NFA},124 封闭性},

CFL{131 识别器{CFG[11],PDA[11]},132 判定性{推导[1311],归约[1311],语法树[1311],移动[1312]},

133 等价性[131-2]{CFG 化简[1311,1321],PDA_N~PDA_F[1312,1324], UPDA~CFG },134 歧义性},

C 编译{

词法分析{联合 DFA[121-2],记号,扫描框架,扫描器,扫描器生成器},

语法分析{221 文法修剪[1311]{首符集,FOLLOW 集,左递归消除,回溯消除,增广文法}

222 自上而下{LL(1)文法[221],LL(1)框架,分析器{递归下降,预测}},

223 自下而上{LR(0)规范簇[1321],活前缀 DFA[1211-2],SLR(1)文法,冲突消解,SLR(1)框架,

ULR(1)分析,ULALR 分析,U 生成器},

语义分析{231 语法制导{属性文法[1322],符号表{作用域},中间语言{三地址码,U 四元式,

UAST,U 逆波兰}},

232 声明翻译[231,1322-3]{变量,数组,函数,数组原型,函数原型},

233 语句翻译[231,1322-3]{赋值,分支,循环,复合},

可执行程序{241 运行时环境{内存映像,活动树,栈帧,栈快照},242 参数传递,

243 名字寻址[2413,242,2631]{局部名,非局部名,形参},

244 代码扩展[2413,242,252,2631]{序言,尾声,调用序列,返回序列,寻址序列}},

U 目标代码生成{指令系统,252C 指令模板,U 库{堆区管理,预定义函数},其它}

U 优化{寄存器分配,代码优化,栈帧优化{2631CD 表,参数传递优化},其它},

R 离散数学,R 数据结构,R 程序设计 C,R 汇编语言{MIPS}

U 基础专业知识

教材及参考书

- 赵银亮 形式语言与编译，西安交通大学（印制），2024
- 陈火旺等. 编译原理. 国防工业出版社，2004

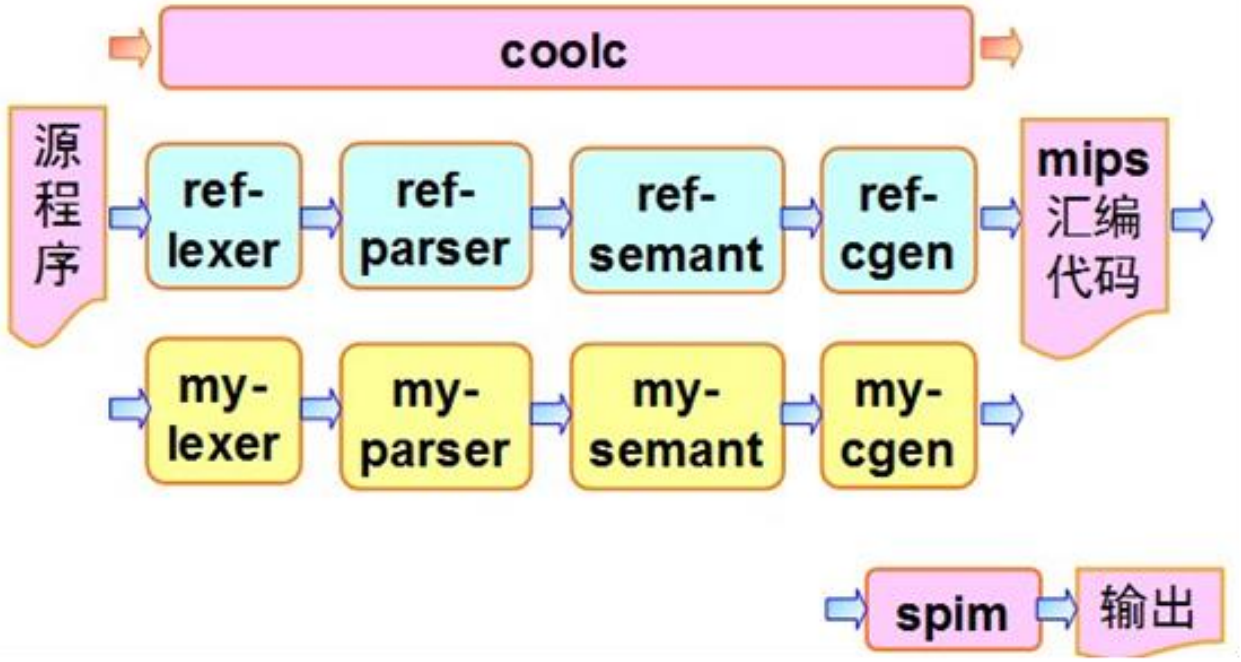
- John E. Hopcroft等. Introduction to Automata Theory, Languages, and Computation (3rd Ed.中文版，孙家骕等译)，机械工业出版社，2008
- Torben Ægidius Mogensen. Introduction to Compiler Design (2nd edition). Springer International Publishing AG, 2017
- 龙书： Alfred V. Aho,et.al.,赵建华等译. 编译原理（第二版）.机械工业出版社，2009
- 注意：围绕知识点组织讲课内容，知识点来自于教材。

先修课程

- 从课程体系角度，认为先修课程《离散数学》、《数据结构》和《程序设计》提供了为本课程所期望的知识和能力基础，不再与本课程内容发生重叠。

关于编译器设计专题实验课的介绍classical

- 斯坦福大学CS143编译原理课程的实验环境
- COOL（Classroom Object Oriented Language）



教学安排

- 综合成绩评定
  - 平时成绩 30%，包括作业、课堂表现、到课情况
  - 期末考试 70%
- 授课方式
  - 线下：见课表安排
  - 思源学堂：师生对接处，对接课程目标、课件、课程资料等信息
  - 雨课堂（待定）：发布作业、公告等，同学提交作业
  - 微信群：临时联系处，通知、公告、反馈
  - 另外有任何联系需求都可发邮箱zhaoy@xjtu.edu.cn

1.5 形式语言概论

- 什么是形式语言？
- 有穷自动机
- 形式文法

什么是形式语言？

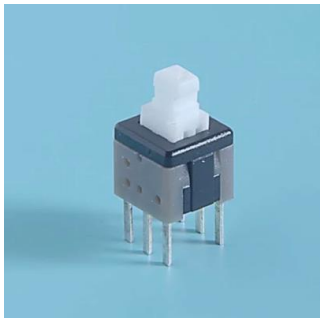
- 直观意义（定义）：形式语言是用来精确地描述语言（包括人工语言和自然语言）及其结构的手段。
- 形式语言学，也称为代数语言学。
- 形式语言以重写规则 $\alpha \rightarrow \beta$的形式表示，其中α与β均为字符串。 一个初步的字符串通过不同的顺序，不断应用不同的重写规则，可以得到不同的新字符串。

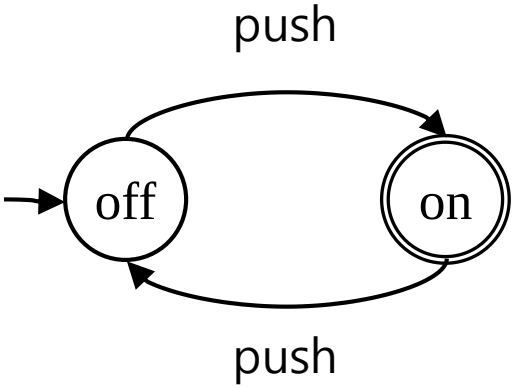
历史回顾

1930s	• 图灵研究图灵机 • 可判定性 • 停机问题
1940-1950s	• “有穷自动机”机器 • Noam Chomsky提出形式语言的“乔姆斯基体系”
1969	• Cook 提出了“NP难”问题
1970-	• 现代计算机科学: 编译器, 可计算性&复杂性理论 得到发展

有穷自动机理论

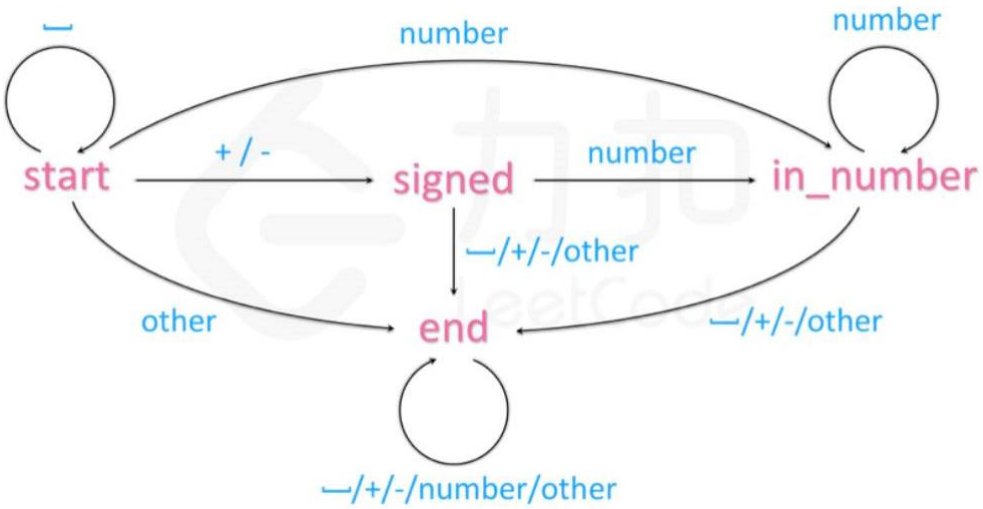
- 在理论计算机科学中，自动机理论是对抽象机器和它们能解决的问题的研究。
- 自动机是有限状态机器，通过状态之间的转移来消耗掉输入串，以判定是否为自动机所接受。
- 设想有穷自动机运行过程中从左到右逐一符号地读输入串，根据输入符号进行状态转移，一旦输入串被读完，自动机就“停止”了。
- 根据自动机停止时的状态是不是“接受”状态，就得到自动机是否接受这个符号串的结论。
- 自动机接受的符号串的全集被称为“这个自动机接受的语言”。





有穷自动机的应用

- 数字电路设计与性能检查
- 单词的模型：词法分析器



热衷开源的宝藏Boy，
blog.csdn.net

区汝就：有限自动机在数字逻辑电路设计中的应用，2020

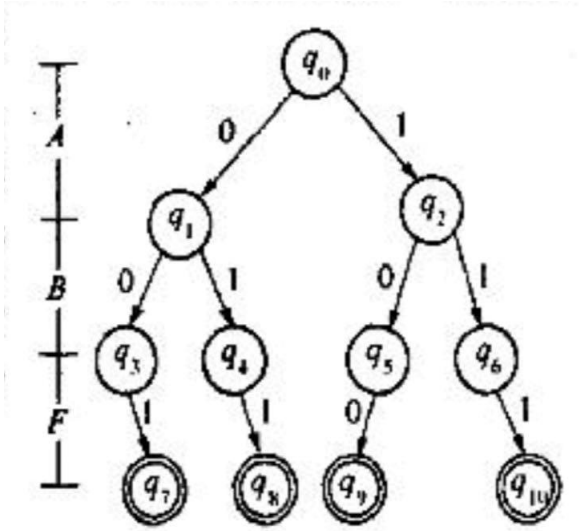


图1 逻辑函数为 $F=\bar{A}+B$ 的组合逻辑电路对应的 FA

有穷自动机的应用

- 单词、短语或其它模式：文本搜索、模式匹配

匹配Email地址：\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

匹配网址URL：[a-zA-z]+://[^\s]*

匹配帐号是否合法(字母开头，允许5-16字节，允许字母数字下划线)：^[a-zA-Z][a-zA-Z0-9_]{4,15}$
评注：表单验证时很实用

匹配国内电话号码：\d{3}-\d{8}|\d{4}-\d{7}
评注：匹配形式如 0511-4405222 或 021-87888822

匹配腾讯QQ号：[1-9][0-9]{4,}
评注：腾讯QQ号从10000开始

匹配中国邮政编码：[1-9]\d{5}(?!\d)
评注：中国邮政编码为6位数字

匹配身份证：\d{15}|\d{18}
评注：中国的身份证为15位或18位

匹配ip地址：\d+\.\d+\.\d+\.\d+
评注：提取ip地址时有用

还有DFA用于实现敏感词过滤等等。。。

有穷自动机的应用

- 对于系统中的各种约定建立模型：协议验证

  基于DFA的协议解析方法
  基于DFA的高速协议分类引擎研究

- 一些物理系统、管理系统的模型：模拟仿真

  口香糖球售货机模型
  简单购物模型
  ……

一些类型的“有穷自动机”

有穷自动机	有少量存储的装置 用于建模非常简单的事情
下推自动机	有无限量存储并以受限方式访问的装置 可用于语法分析
图灵机	有无限量存储的装置 这是实际的计算机
时间界限图灵机	存储器无限大，但运行时间有界 这是以合理速度运行的计算机

形式文法

- 形式文法描述形式语言的基本想法是，从一个特殊的开始符号出发，不断地应用一些产生式规则，从而生成出一个符号串的集合。
- 产生式规则指定了某些符号组合如何被另外一些符号组合替换。

- 1. S→aSb
- 2. S→ba

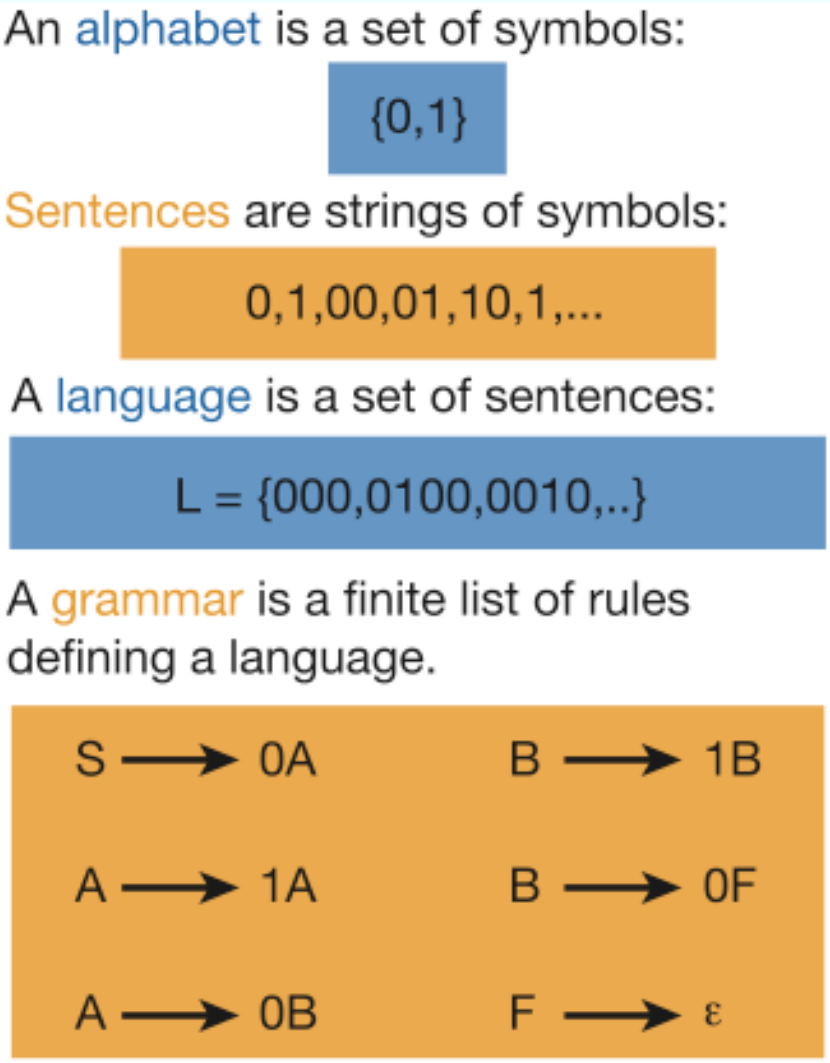
- S⇒aSb⇒aaSbb⇒aababb
- ba, abab, aababb, aaababbb, …

形式文法

- 在计算机科学中，形式语言是字母表上，一些有限长符号串的集合。
- 形式文法是描述这种语言的一种途径。

- Languages: “语言是句子的集合，句子是有穷长度的且由某个有穷字母表中的符号构成”
- Grammars: “文法可以被看做是一个装置，它恰好枚举一个语言的句子

- *N. Chomsky, Information and Control, Vol 2, 1959*



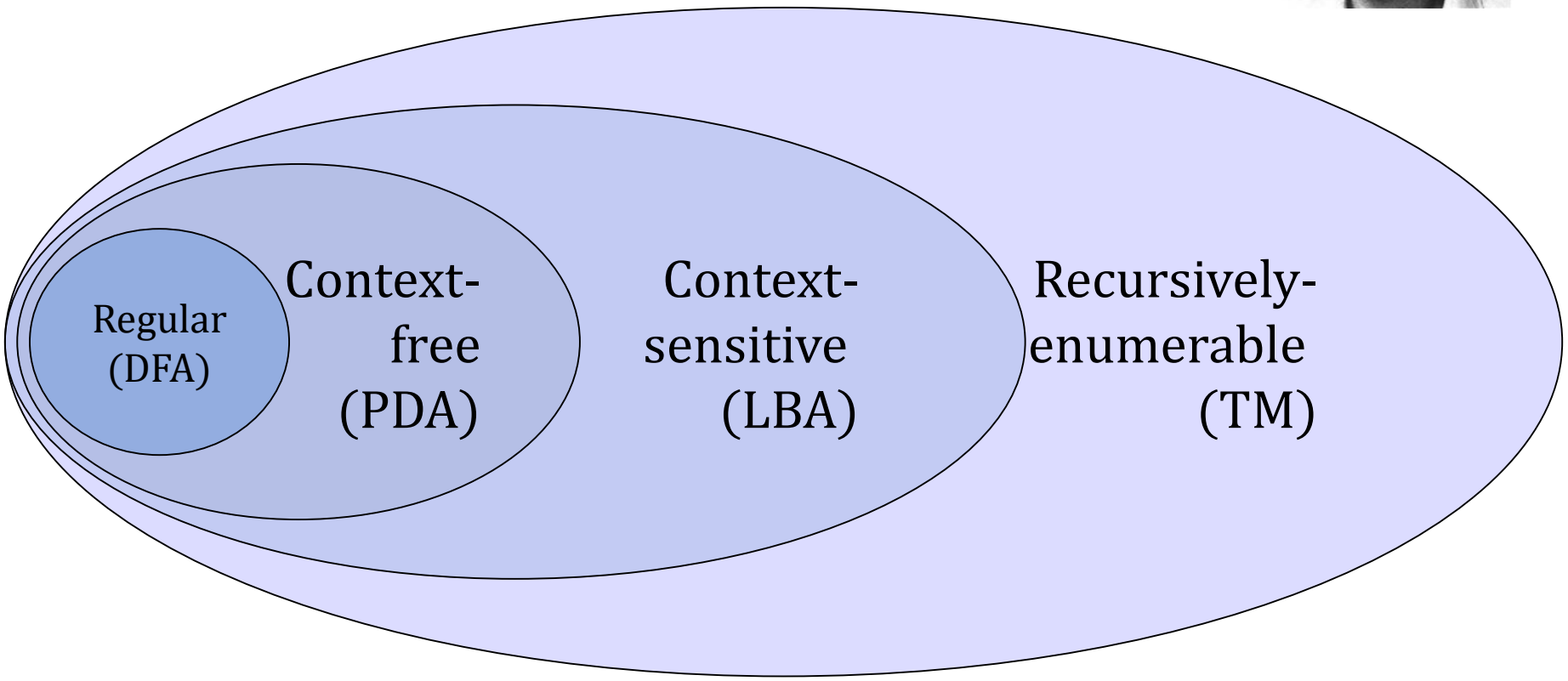
自动机、形式文法、形式语言的等价性

自动机	形式文法	形式语言
DFA	正则文法	正则语言
PDA	CFG	CFL
TM	0型文法	递归可枚举语言

- 自动机便于执行
- 形式文法便于推导、推理、生成
- 形式语言便于枚举

Chomsky 体系

计算机科学中刻画形式文法表达能力的一个分类谱系，是由诺姆·乔姆斯基于1956年提出的。



3-型文法	2-型文法	1-型文法	0-型文法
$A\to\varepsilon\|a\|cB$	$A\to\gamma$	$\alpha A\beta\to\alpha\gamma\beta$	无限制文法

关于可计算性、复杂性理论

- **图灵机的运行时间：**
  对于某输入，图灵机运行的步数。
  - 时间有穷：从开始运行到停机所经过的步数。
  - 时间无穷：不停机。

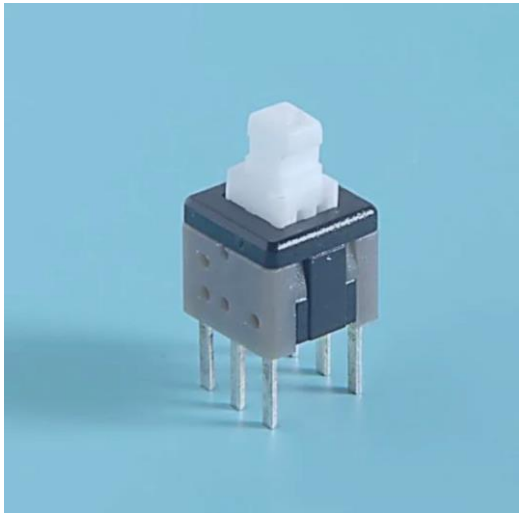
- 可计算性：即能否为图灵机所判定（可判定性）
- 计算复杂性：对问题按难度分类，寻找对难问题的转化求解方法。

本课程讨论范围

自动机	文法	语言
DFA	正则文法	正则语言
PDA	CFG	CFL

两相开关的DFA建模

建模对象：两相开关，是一个物理系统。

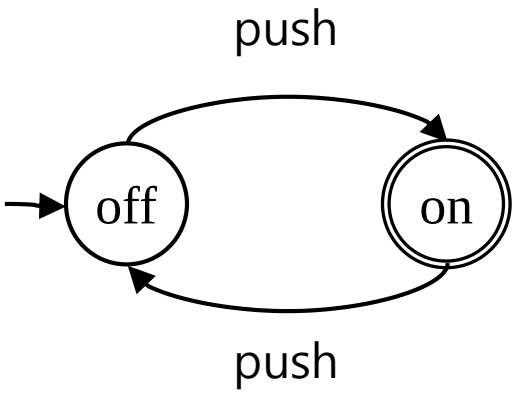


- 有“开”和“关”两个状态；
- 外界按压（push）表示在系统边界上发生的交互行为；
- 开始时开关的状态为“开”；
- 一个push导致状态“开”、“关”切换一次；
- 一到多个push之后开关的状态是“开”，则达到接通之目的。

建模结果：一个DFA

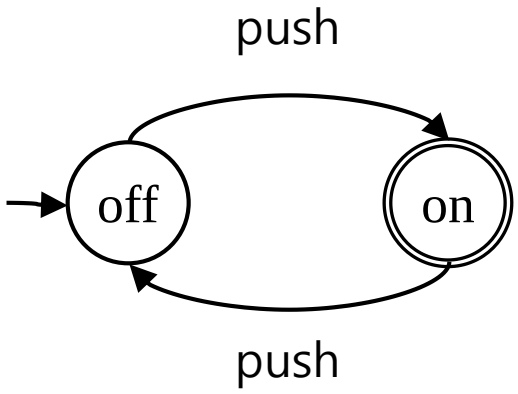
两相开关DFA

- 建模结果



- 两个状态on、off
- 初始状态off
- 接受状态on
- 状态转移弧
  (off, push) -> on
  (on, push) -> off
- 转移弧上的标记push
- 输入符号push

两相开关DFA运行过程



- ✓ 给定输入串，由单个符号push组成的符号串；
- ✓ 从初始状态开始，并将输入串第一个符号作为当前输入符号；
- ✓ 对于当前输入（push表示）发生状态转移；
- ✓ 当输入串消耗完自动机到达on则接受；
- ✓ 否则不接受；
- ✓ 本模型中接受表示物理系统接通，否则是断开的。

1.6 本课程中心概念

- 中心概念：符号，字母表，符号串，语言，问题
- 主文法与主符号系统

符号、字母表

- 符号是什么？指代事物的手段

> a letter, figure, or other character or mark or a combination of letters or the like used to designate something.

- 符号的来源与它的指代物
- 符号作为名字
- 计算系统与它的那些有穷个的符号
- 形式化原则

符号、字母表

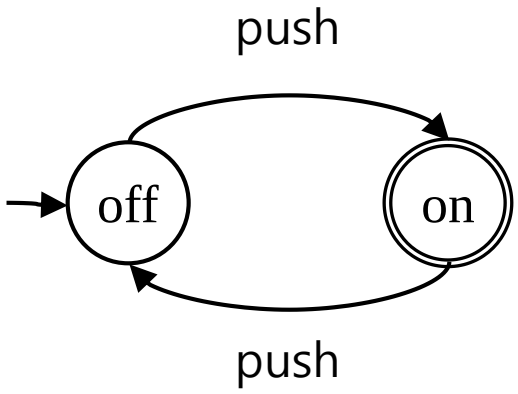
- 字母表是感兴趣的符号的非空集合有穷集合
- 感兴趣？认知域、论域、计算系统

- 二进制数字集合，$\Sigma_B=\{0,1\}$。
- 十进制数字集合，$\Sigma_D=\{0,1,2,3,4,5,6,7,8,9\}$。
- 所有小写拉丁字母集合，$\Sigma_a=\{a,b,\ldots,z\}$。
- 所有ASCII字符的集合。
- 空白字符的集合，S={\t,\n,\r,\0,' '}。
- 字母加特殊符号$构成的集合，$\Sigma_{a\$}=\Sigma_a\cup\{\$\}$。

符号串

- 符号串是构建在字母表上的最为基础的结构。
- 一个符号串是由字母表中符号组成的有穷长度的一个序列。
  - abfbz 是Σ_a上的一个串， $\Sigma_a = \{a, b, c, d, ..., z\}$
  - 90221是Σ_D上的一个串， $\Sigma_D = \{0, 1, ..., 9\}$
  - ac$bc是$\Sigma_{a\$}$上的一个串， $\Sigma_{a\$} = \{a, b, ..., z, \$\}$
- 符号串的元素、元素的前驱、元素的后继、元素的索引
- w为一个符号串，|w|为w的长度，长度为0的符号串ε
- 函数π(w, i)为w的i-前缀，π(w)为所有前缀组成的集合
- 函数τ(w, i)为w的i-后缀，τ(w)为所有后缀组成的集合
- Σ上的所有符号串组成一个集合记为Σ^*

例子（注意到与形式化原则的冲突）



- 状态：on, off
- 初始状态：on
- 接受状态：off
- 转移弧：
  off状态射出的push弧射入on状态；
  on状态射出的push弧射入off状态；
- 转移弧上的标记：push
- 输入符号：push
- 字母表：{push}
- 输入：字母表上符号串

语言

- 长度为0的符号串是空串，习惯上采用 ε 表示空串。
- 字母表Σ上的符号串全集是Σ^*

- Σ上的语言是Σ 上符号串的集合
- 显然， Σ 上的语言是Σ^*的一个子集

L_1 是Σ_a上的含有子串“to”的所有串

stop, to, toe are in L_1

e, oyster are not in L_1

$L_1 = \{x \in \Sigma_a^* \mid x$ 含有子串“to”$\}$

语言的例子

$L_2 = \{x \in \Sigma_D^* \mid x \text{被7整除}\}$

$= \{7, 14, 21, \ldots\}$

$L_3 = \{s\$s \mid s \in \Sigma_{a\$}^*\}$

ab\$ab in L_3

ab\$ba not in L_3

a\$\$a\$ not in L_3

什么是问题？

- 要求解的问题都是判定性的问题

给定词s, 该词包含“to”子串吗?
给定整数n, 它能被7整除吗?
给定符号串s 和 t, 它们相同吗?

假定字母表是Σ_1，问题即 $s \in L_1$?

假定字母表是Σ_2，问题即 $n \in L_2$?

问题即 s与t的逆连接起来是否为回文?

问题即 $s\$t \in \{x\$x \mid x \in \Sigma^*, \$ \notin \Sigma\}$?

判定性问题的答案为是或否。

任何其它问题如查找、统计等都可以等价转化为判定性问题。

主文法与主符号系统

- 主文法贯穿始终解决了碎片化问题和可扩展问题。
- 命名惯例：一致性、简洁性
- 是代数系统

关于形式化证明

- 形式化证明很有用，在后续课程中用起来
  - 演绎证明
  - 归纳证明：数学归纳法
  - 反证法
- 鸽巢原理
- 直接运用先修课的知识

数学归纳法

Example

Prove that $1^2 + 2^2 + 3^2 + \ldots + n^2$ is $\frac{n(n+1)(2n+1)}{6}$.

- **Basis (n=1):** $\frac{1(1+1)(2\cdot 1+1)}{6} = \frac{2\cdot 3}{6} = 1 = 1^2$.
- **Induction:** Let $\mathcal{P}(n) := 1^2 + \ldots + n^2$ is $\frac{n(n+1)(2n+1)}{6}$.

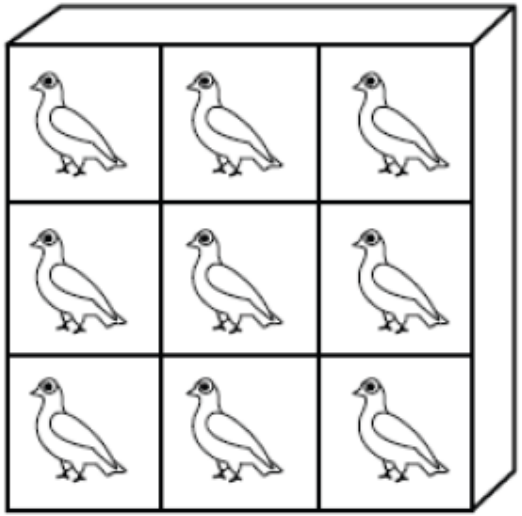
  $\mathcal{P}(n+1) = 1^2 + \ldots + n^2 + (n+1)^2 = \frac{n(n+1)(2n+1)}{6} + (n+1)^2$

  $= (n+1)\left\{\frac{n(2n+1)+6n+6}{6}\right\} = (n+1)\left\{\frac{2n^2+7n+6}{6}\right\}$

  $= (n+1)\left\{\frac{2n^2+4n+3n+6}{6}\right\} = \frac{(n+1)(n+2)(2n+3)}{6}$
- Since $\mathcal{P}(1)$ is true and $\mathcal{P}(n) \Rightarrow \mathcal{P}(n+1)$, we conclude that $\mathcal{P}(n)$ is true for all $n \geq 1$.

鸽巢原理

如果有n+1个鸽子和n个鸽巢，那么某个鸽巢必须至少住进两只鸽子。



如果有kn+1个鸽子和n个鸽巢，那么某个鸽巢必须至少住进k+1只鸽子，k>=1.

Pigeonhole Principle

If there are $n+1$ pigeons and n pigeonholes, then some pigeonhole must contain at least two pigeons.

其它预备知识

- 图论
  - 有向图、无向图
  - 结点的度、入度、出度
  - 路径、路径长度、环
- 集合论
  - 集合运算及性质
  - 关系、映射、函数
  - 划分、覆盖
- 数据结构
  - 线性表、树、杂凑表、链表
  - 操作、算法及其复杂度

小结与作业

- 知识点：编译相关术语，编译过程，图灵机模型，有穷自动机构成及原理，形式文法概念，符号、字母表、符号串、语言、问题等概念。

- 作业：p18-19习题1.1和1.2以及1.3或1.4二选一
- 阅读：教材第一章