清华大学
Tsinghua University

清华大学
计算思维八周课程
《第二周》

顾学雍

清華大學 iCenter

目录

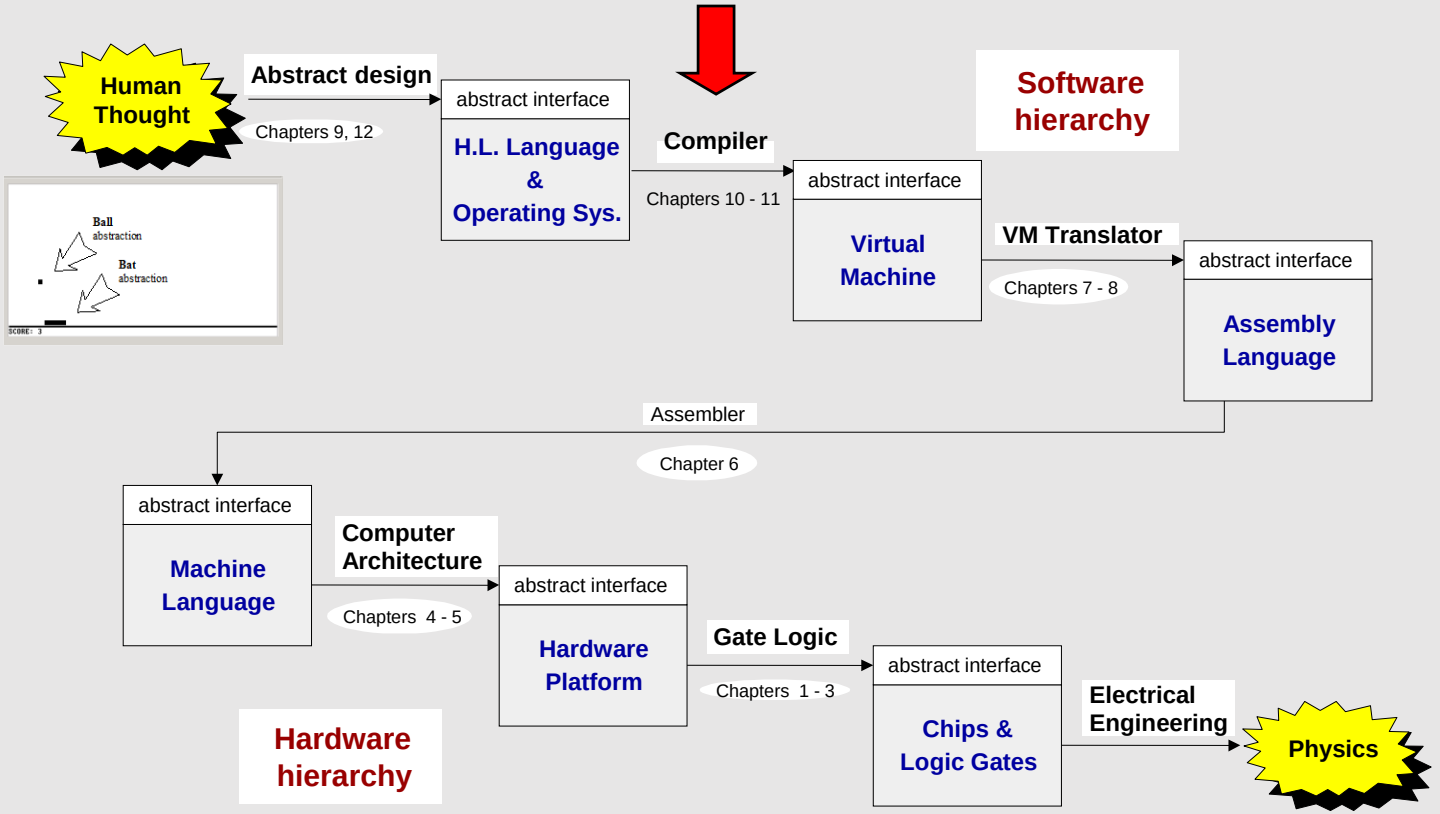
01. Nand2Tetris课程概括

Nand2Tetris课程概括

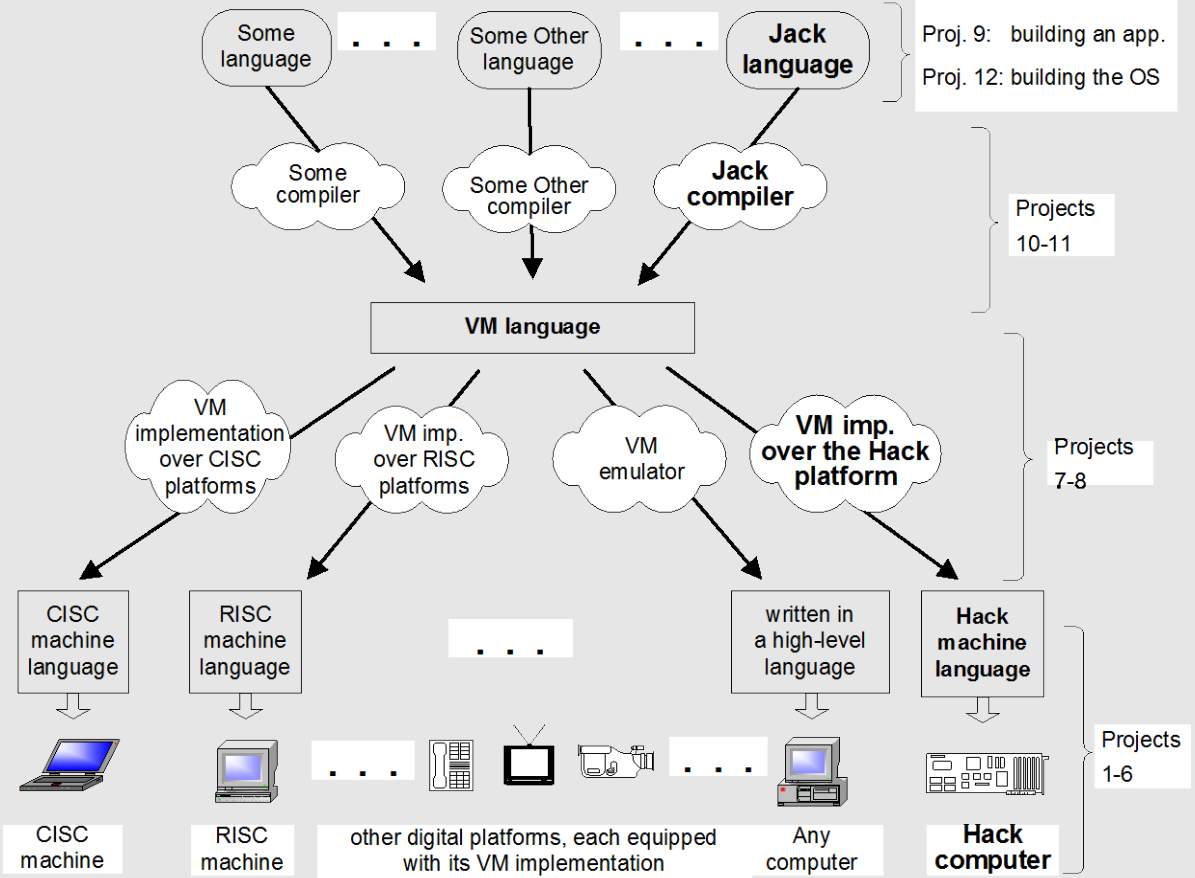
- 来源
  - 网络学校: Coursera.org
  - 讲师: Noam Nisan, Shimon Schocken
- 背景
  - 计算机科学专业的学生并不了解计算机的底层实现原理
  - 越来越少的学生选修编译相关的课程
  - 很多计算机结构课程的教学内容过于详细而且枯燥
  - 并没有激起从无到有的这种兴奋感

- 目标
  - 指导学生和自学者了解从电路起了解现今计算机的软硬件架构
  - 让学生透过完成12节课的作业充分了解计算机的抽象实现
- 课程特色
  - 模块化的课程内容和作业允许学生根据个人需要单独学习需要的部分
  - 作业中使用的仿真软件可以在各种操作系统上运行
  - 完成课程所需要的所有知识均包含在课程当中，适合本科或以上学历的计算机专业学生
  - 非专业学生建议先掌握基本的编程的知识
  - 仅从抽象层次讲解计算机软硬件架构，不考虑具体的硬件实现
  - 正如现今软硬件工程师在设计计算机时同样不考虑具体硬件实现

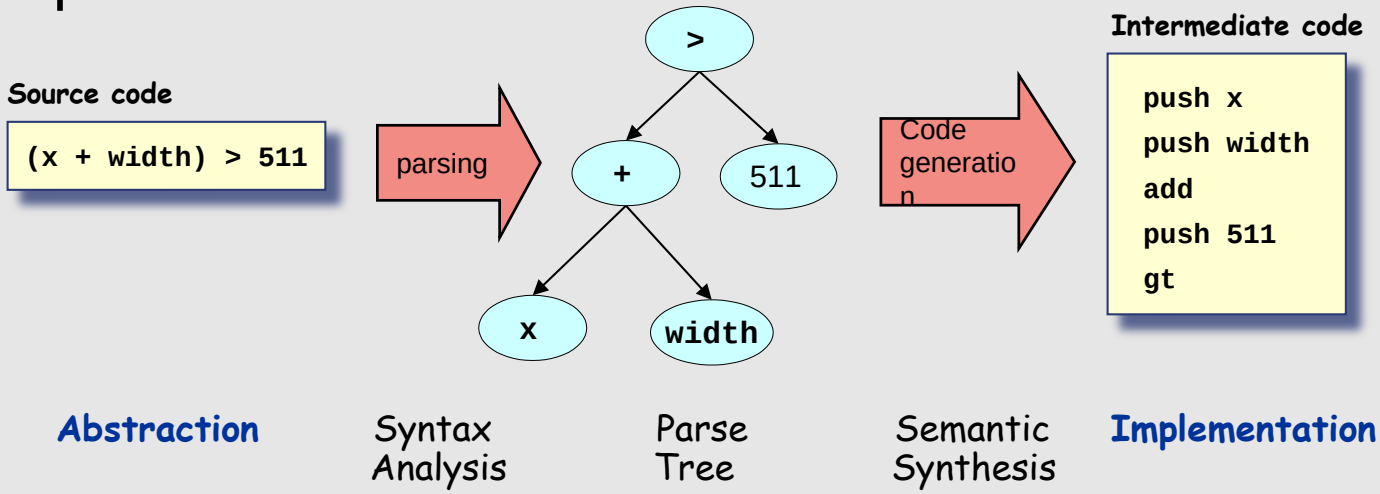
The big picture



A modern compilation model



Compilation 101



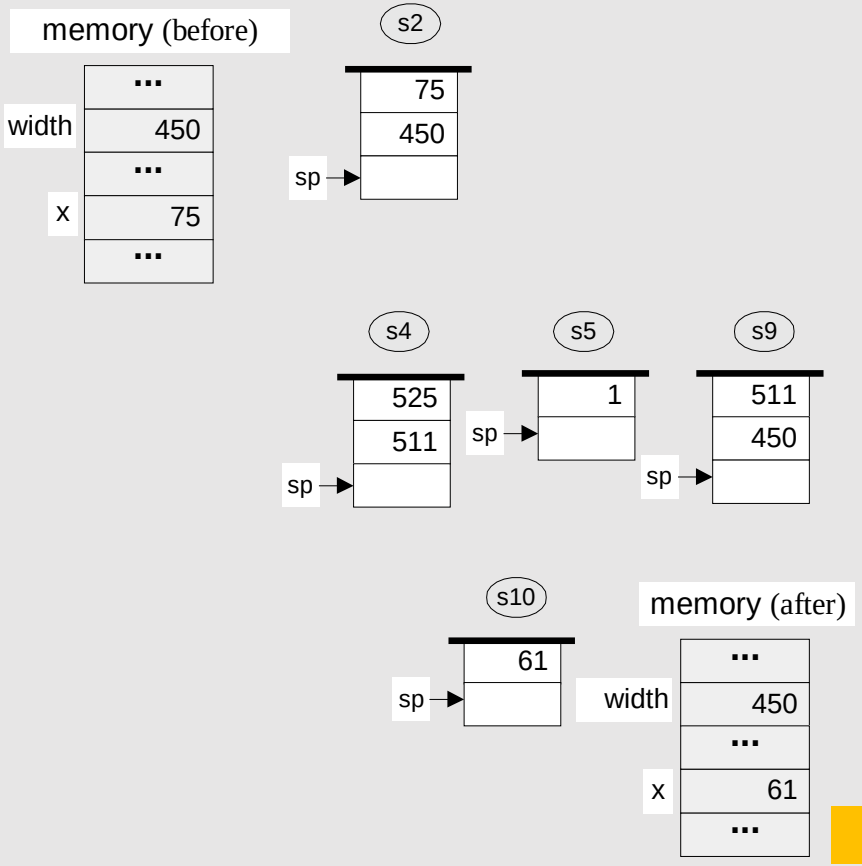
Observations:

- Modularity
- Abstraction / implementation interplay
- The implementation uses abstract services from the level below.

The Virtual Machine (our very own VM, modeled after Java's JVM)

```
if ((x+width)>511) {
   let x=511-width;
}
```

```
// VM implementation
   push x        // s1
   push width    // s2
   add           // s3
   push 511      // s4
   gt            // s5
   if-goto L1    // s6
   goto L2       // s7
L1:
   push 511      // s8
   push width    // s9
   sub           // s10
   pop x         // s11
L2:
...
```



课程大纲

- 第一部分: 布尔逻辑与电子电路
  - 对应课程
    - Lecture 1 - Boolean Logic 布尔逻辑: 讲解布尔逻辑与门电路
    - Lecture 2 - Boolean Arithmetic 布尔运算: 讲解基于布尔逻辑实现数字运算
    - Lecture 3 - Sequential Logic 时序逻辑: 介绍时钟和Flip-flop，以及基于其实现的存储部件
  - 对应作业
    - Project 1: 基于Nand门编写And, Or, Not等16個門電路的.hdl文件
    - Project 2: 基于And, Or, Not等元件编写全加法器，半加法器和ALU等元件的.hdl文件
    - Project 3: 基于DIFF編寫Flip-flop，RAM等存储部件的.hdl文件
- 第二部分: 机器语言
  - 对应章节
    - Lecture 4 - Machine Language 机器语言: 讲解机器语言与编码
  - 对应作业
    - Project 4: 利用汇编语言实现顯示屏顯示和乘法 (汇编语言经由CPUEmulator转换为二进制机器语言.hack)

课程大纲

- 第三部分: 汇编语言
  - 对应章节
    - Lecture 5 - Computer Architecture 计算机架构: 讲解计算机的抽象架构
    - Lecture 6 - Assembler 汇编器: 讲解汇编语言以及到机器语言的转换
  - 对应作业
    - Project 5: 编写CPU， Memory, computer等部件，并实现加法等程序的.hdl文件
    - Project 6: 运行由Jack语言编写Add, Max, 画矩陣和Pong遊戲的.asm代码
- 第四部分: 虚拟机
  - 对应章节
    - Lecture 7 - Virtual Machine I 虚拟机: 讲解VM中栈的push, pop操作
    - Lecture 8 - Virtual Machine II 虚拟机: 讲解基于VM中实现的函数调用 *　对应作业
    - Project 7: 利用VM Emulator运行讀取Memory和棧操作的程序，虛擬器的抽象存儲結構
    - Project 8: 利用VM Emulator运行函數調用和程序控制(if, loop)的程序，了解編程技巧的抽象實現

课程大纲

- 第五部分: 高级语言
  - 对应章节
    - Lecture 9: High Level Language 高级语言
    - Lecture 10: Compiler I 编译器
    - Lecture 11: Compiler II 编译器
  - 对应作业
    - Project 9: 利用Compiler运行HelloWord, List等Jack語言代碼
    - Project 10: 利用Compiler运行ArrayTest, Square等Jack語言代碼
    - Project 11: 利用Compiler运行Average, ComplexArray等Jack語言代碼
- 第六部分: 操作系统与软件应用
  - 对应章节
    - Lecture 12: OS
  - 对应作业
    - Project 12: 利用Jack实现Keyboard, Memory，Screen等功能
    - Project 13: 鼓励学生基于现有框架开发自己的应用
- 第七部分 分布式计算与基于计算服务的系统架构
  - Service Oriented Architecture, 微服务与DevOps (Continuous Integration,

课程大纲

- 第七部分 分布式计算与基于计算服务的系统架构
  - 对应章节
    - Lecture 12: Service oriented Architecture
    - Lecture 13: DevOps, Continuous Integration
    - Lecture 14: Kubernetes, Jenkins
  - 对应作业
    - Individual Project: 根据周杨波所提供的微服务，在个人计算机上安装一个可协同运转的围棋对弈网站。
    - Team Project: 利用现有的微服务的元组件，增加一个服务功能。
    - Class Project: 把小组开发的一个微服务功能，融合到现有围棋对弈计算微服务系统之中。

- Service Oriented Architecture, 微服务与DevOps (Continuous Integration,

START WITH (CHOICE OF MANY) STANDARD PARTS

Standardized

Open

Multi-vendor

Multi-platform

But these are just piece parts

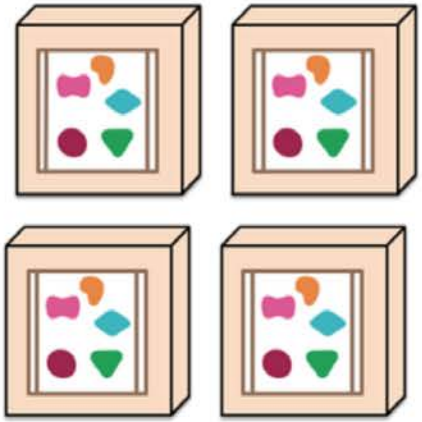
Gordon Haff
@ghaff
ghaff@redhat.com

5 November 2014

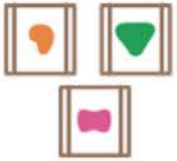
MICROSERVICES ENABLE SOFTWARE COMPONENT REUSE

A monolithic application puts all its functionality into a single process...

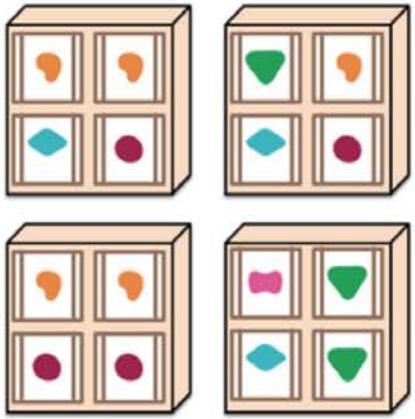
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



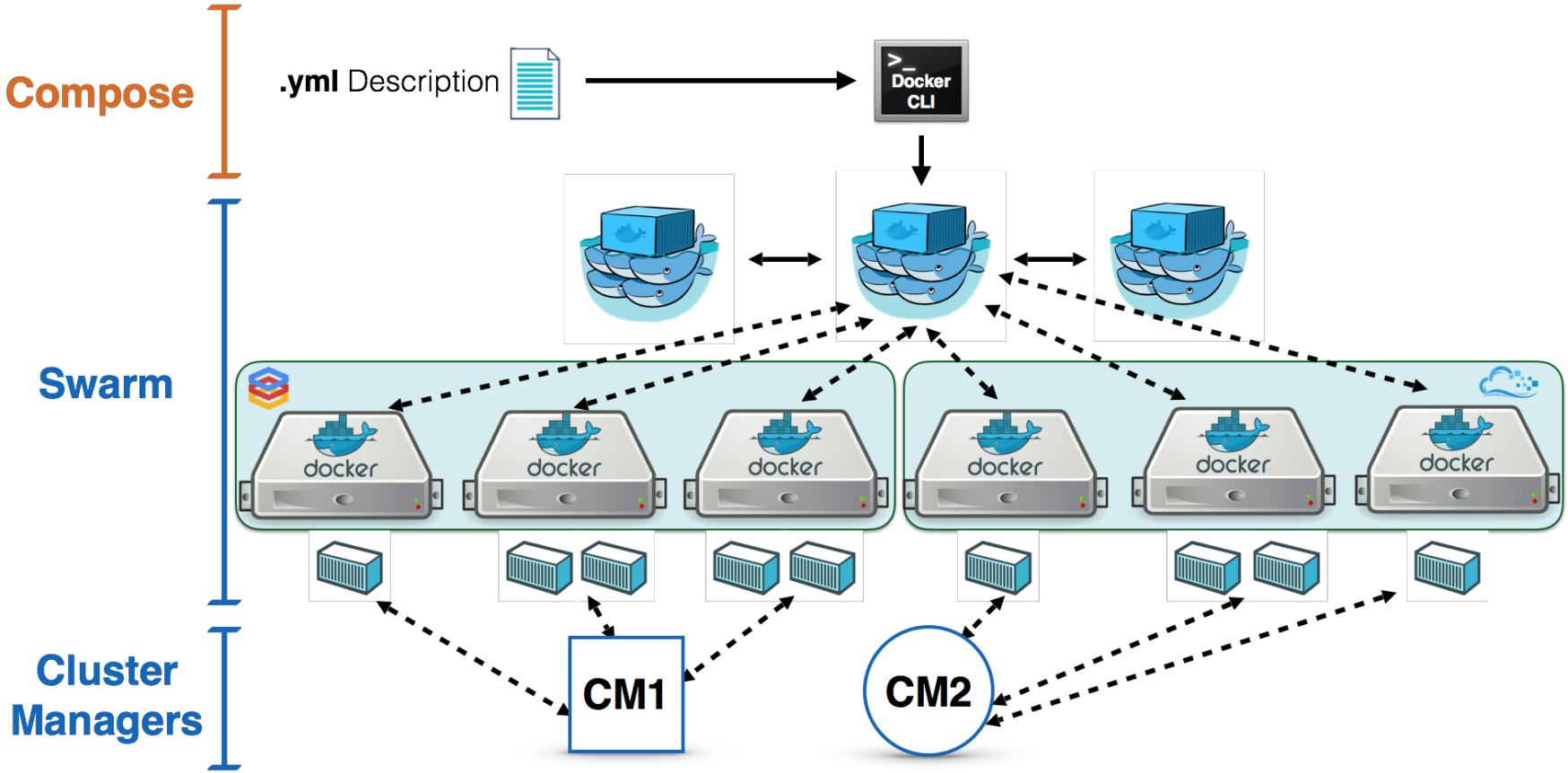
... and scales by distributing these services across servers, replicating as needed.



http://martinfowler.com/articles/microservices.html

Gordon Haff
@ghaff
ghaff@redhat.com

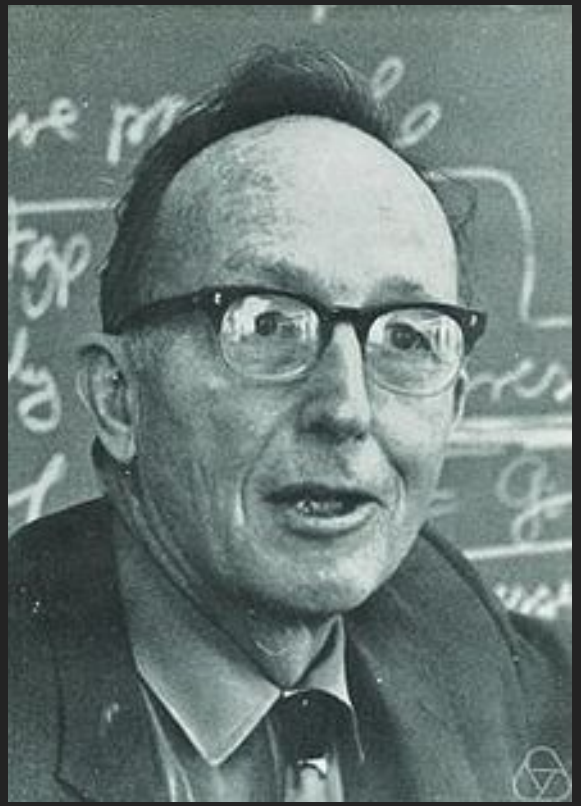
5 November 2014



https://blog.docker.com/media/2015/11/image00.png

02. 关键人物介绍

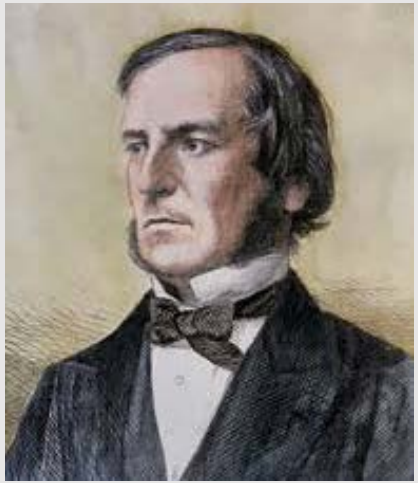
Saunders Mac Lane/Samuel Eilenburg 一切范畴皆由函数组合而成



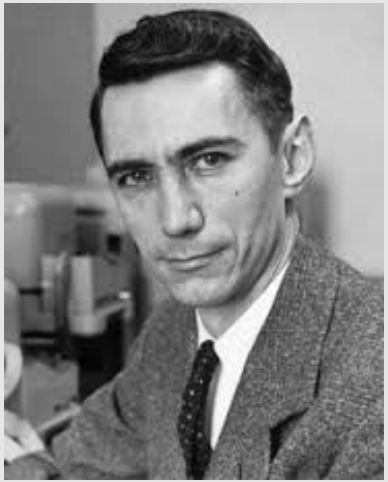
https://en.wikipedia.org/wiki/Saunders_Mac_Lane

https://en.wikipedia.org/wiki/Samuel_Eilenberg

Thinking is governed by the Computational Nature of Languages



George Boole
Boolean Algebra



Claude Shannon
A Symbolic Analysis of Relay and Switching Circuits

Bertrand Meyer
Design by Contract

乔治·布尔 George Boole

生平
1815.11.2～1864.12.8
19世纪最重要的数学家之一
思考“人的思想能不能用数学表达?”当时数学只用于计算，没有人意识到数学还能表达人的逻辑思维。
关键出版物

The Mathematical Analysis of Logic 《逻辑的数学分析》
出版于1847年
用通常的代数符号并以等式来表示逻辑关系
该分析方法可以说是对符号进行形式上的处理而抽出了符号所代表的具体涵义
该分析方法后来被称为“逻辑代数”和“布尔代数”。
这是他对符号逻辑诸多贡献中的第一次。

The Laws of Thought 《思维规律的研究》
出版于1854年
乔治·布尔最著名的著作
书中提出了一种与逻辑规律类似的概率算法
检验了大量概率研究中必然产生的独特问题
书中的内容使布尔的逻辑思想臻于完善

乔治·布尔 George Boole

理论说明

乔治·布尔提出的逻辑代数建基于集合

他认为逻辑思维的基础是一个个集合（Set）

每一个命题表达的都是集合之间的关系

e.g. 1所有人类组成一个集合H，所有会死的东西组成一个集合D 则 "所有人都是要死的" 可以描述为 H X D = H -- (1) 其中 X 表示集合的交集

e.g. 2考虑集合S只包含一个成员苏格拉底 则 "苏格拉底是人" 可以描述为 S X R = S -- (2)

e.g. 3结合(1)(2)推导 S X (H X D) = (S X H) X D = S X D = S 即可以得出苏格拉底会死

局限

布尔代数可以判断命题真伪，但是无法取代人类的理性思维

布尔代数必须依据一个或几个已经明确知道真伪的命题，才能做出判断

比如，只有知道"所有人都会死"这个命题是真的，才能得出结论"苏格拉底会死"

布尔代数只能保证推理过程正确，无法保证推理所依据的前提是否正确

如果前提是错的，正确的推理也会得到错误的结果

克劳德·艾尔伍德·香农

Claude Elwood Shannon

生平

1916年4月30日 — 2001年2月24日

美国数学家

信息论的创始人

关键出版物

A Symbolic Analysis of Relay and Switching Circuits 《继电器与开关电路的符号分析》

1938年香农在MIT获得电气工程硕士学位的硕士论文

把布尔代数的“真”与“假”和电路系统的“开”与“关”对应起来，并用1和0表示

提出利用布尔代数分析并优化开关电路，奠定了数字电路的理论基础

同时展示了利用电子电路执行布尔代数(由逻辑命题转换而成的代数方程)的方法

哈佛大学的Howard Gardner教授表示，“这可能是本世纪最重要、最著名的一篇硕士论文。”

与英国科学家艾伦·图灵1937年发表著名的《论应用于解决问题的可计算数字》存在一个共同点

证明了采用简单二进制指令操作的机器不仅可以用于解决数学问题，同时也适用于所有的逻辑问题。

[注] 两人后来在图灵所在的贝尔实验室相识

A mathematical Theory of communication 《通信的数学理论》

1948年发表于 Bell System Technical Journal

香农在数学与工程研究上的顶峰

把通信理论的解释公式化

对最有效地传输信息的问题进行了研究

[注] 论文由香农和威沃共同署名。前辈威沃（Warren Weaver，1894-1978）当时是洛克菲勒基金会自然科学部的主任，他为文章写了序言

参考资料

其他摘要

- 如何区分机器和人
  - 对于这个领域的大多数问题，笛卡儿都能够运用现代的语言进行表述。“我思故我在”这个著名的哲学观点正是出自他在1637 年出版的《谈谈方法》(Discourse on the Method)一书中。笛卡儿在这本书中写道：
  - 如果存在一些跟我们的身体类似的机器，它们能够在各个方面尽可能接近地模仿我们的动作，我们还是可以利用两条非常可靠的标准来判明它们并不是真正的人类。第一条是……这种机器绝不能对自己接收到的任何内容都做出条理清晰的回应，而这是最愚蠢的人都能办到的。第二条是，虽然某些机器在完成某些工作的时候可以做得跟我们一样好，甚至可以做得更好，但是它们肯定做不好其他的事情，这点表明它们的行为并非建立在理解的基础上。
  - 摘录自 机器也许有一天能像人类一样思考 - 电子工程世界网

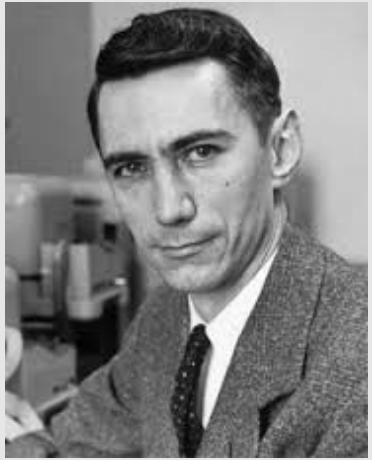
《编码的奥妙》第十章：http://vdisk.weibo.com/s/BZE2czijFfNzb?sudaref=www.baidu.com

布尔代数入门 - 阮一峰的网络日志：http://www.ruanyifeng.com/blog/2016/08/boolean-algebra.htm

克劳德·艾尔伍德·香农 - 百度百科： https://baike.baidu.com/item/克劳德·艾尔伍德·香农/10588593?fr=aladdin&fromid=1146248&fromtitle=香农

机器也许有一天能像人类一样思考 - 电子工程世界网 http://www.eeworld.com.cn/wltx/article_2017071415927.html

The Most Important Master Thesis, ever!



A Symbolic Analysis of Relay and Switching Circuits*

*Claude E. Shannon***

I. Introduction

In the control and protective circuits of complex electrical systems it is frequently necessary to make intricate interconnections of relay contacts and switches. Examples of these circuits occur in automatic telephone exchanges, industrial motor-control equipment, and in almost any circuits designed to perform complex operations automatically. In this paper a mathematical analysis of certain of the properties of such networks will be made. Particular attention will be given to the problem of network synthesis. Given certain characteristics, it is required to find a circuit incorporating these characteristics. The solution of this type of problem is not unique and methods of finding those particular circuits requiring the least number of relay contacts and switch blades will be studied. Methods will also be described for finding any number of circuits equivalent to a given circuit in all operating characteristics. It will be shown that several of the well-known theorems on impedance networks have roughly analogous theorems in relay circuits. Notable among these are the delta-wye and star-mesh transformations, and the duality theorem.

Formal Languages can be formulas or diagrams

Frequently a function may be written in several ways, each requiring the same minimum number of elements. In such a case the choice of circuit may be made arbitrarily from among these, or from other considerations.

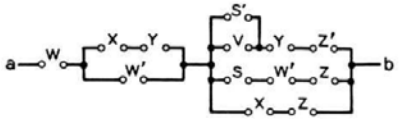


Figure 5. Circuit to be simplified

As an example of the simplification of expressions consider the circuit shown in Figure 5. The hindrance function X_{ab} for this circuit will be:

$$\begin{aligned} X_{ab} &= W + W'(X + Y) + (X + Z)(S + W' + Z)(Z' + Y + S'V) \\ &= W + X + Y + (X + Z)(S + 1 + Z)(Z' + Y + S'V) \\ &= W + X + Y + Z(Z' + S'V) \ . \end{aligned}$$

A transformation will now be described for reducing the number of elements required to realize a set of simultaneous equations. This transformation keeps $X_{0k} (k = 1,2 \ldots n)$ invariant, but $X_{jk} (j,k = 1,2 \ldots n)$ may be changed, so that the new network may not be equivalent in the strict sense defined to the old one. The operation of all the relays will be the same, however, This simplification is only applicable if the X_{0k} functions are written as sums and certain terms are common to two or more equations. For example, suppose the set of equations is as follows:

$$\begin{aligned} W &= A + B + CW \ , \\ X &= A + B + WX \ , \\ Y &= A + CY \ , \\ Z &= EZ + F \ . \end{aligned}$$

This may be realized with the circuit of Figure 14, using only one A element for the three places where A occurs and only one B element for its two appearances. The justification is quite obvious. This may be indicated symbolically by drawing a vertical line after the terms common to the various equations, as shown below.

$$\begin{array}{lll} W = & & B + \ \vert \ CW \\ X = & A + & \ \vert \ WX \\ Y = & & CY \ \vert \\ Z = & F + EZ & \end{array}$$

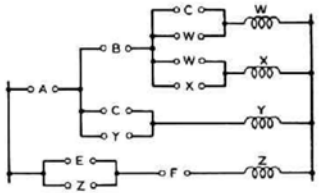


Figure 14. Example of reduction of simultaneous equations

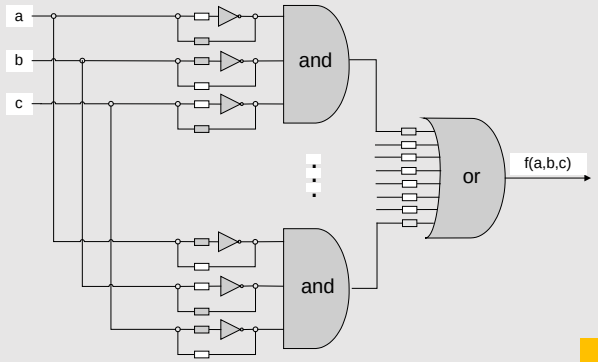
The “Symbol Table” between graphical and formulaic languages

Table I. Analogue Between the Calculus of Propositions and the Symbolic Relay Analysis

Symbol	Interpretation in Relay Circuits	Interpretation in the Calculus of Propositions
X	The circuit X	The proposition X
0	The circuit is closed	The proposition is false
1	The circuit is open	The proposition is true
$X + Y$	The series connection of circuits X and Y	The proposition which is true if either X or Y is true
$X\ Y$	The parallel connection of circuits X and Y	The proposition which is true if both X and Y are true
X'	The circuit which is open when X is closed and closed when X is open	The contradictory of proposition X
=	The circuits open and close simultaneously	Each proposition implies the other

Perspective

- Each Boolean function has a canonical representation
- The canonical representation is expressed in terms of And, Not, Or
- And, Not, Or can be expressed in terms of Nand alone
- Ergo, every Boolean function can be realized by a standard PLD consisting of Nand gates only
- Mass production
- Universal building blocks, unique topology
- Gates, neurons, atoms, …



End notes: Canonical representation

Whodunit story: Each suspect may or may not have an alibi (a), a motivation to commit the crime (m), and a relationship to the weapon found in the scene of the crime (w). The police decides to focus attention only on suspects for whom the proposition Not(a) And (m Or w) is true.

Truth table of the "suspect" function $s(a,m,w) = \overline{a} \cdot (m + w)$

a	m	w	*minterm*	suspect(a,m,w)= not(a) and (m or w)
0	0	0	$m_0 = \overline{a}\,\overline{m}\,\overline{w}$	0
0	0	1	$m_1 = \overline{a}\,\overline{m}\,w$	1
0	1	0	$m_2 = \overline{a}\,m\,\overline{w}$	1
0	1	1	$m_3 = \overline{a}\,m\,w$	1
1	0	0	$m_4 = a\,\overline{m}\,\overline{w}$	0
1	0	1	$m_5 = a\,\overline{m}\,w$	0
1	1	0	$m_6 = a\,m\,\overline{w}$	0
1	1	1	$m_7 = a\,m\,w$	0

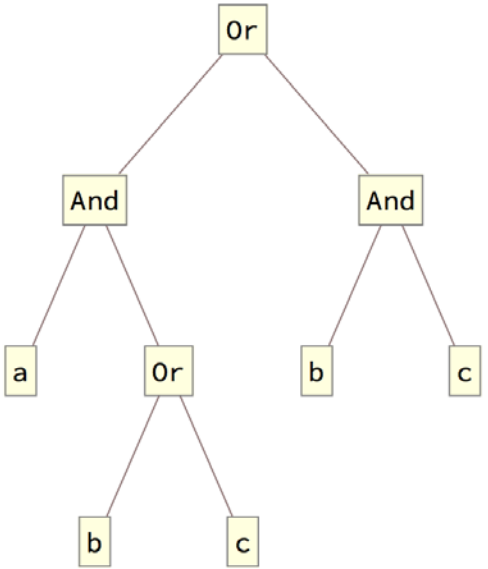
Canonical form: $s(a,m,w) = \overline{a}\,\overline{m}\,w + \overline{a}\,m\,\overline{w} + \overline{a}\,m\,w$

自动生成第一章作业的答案

```
TableForm[BooleanTable[{a, b, c, (a && (b || c)) || (b && c)}, {a, b, c}] // Boole,
 TableHeadings → {None, {a, b, c, (a && (b || c)) || (b && c)}}]
```

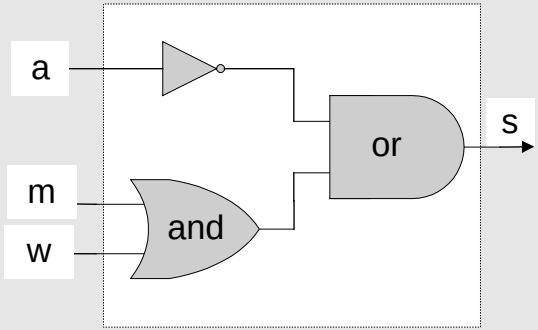
a	b	c	(a && (b || c)) || (b && c)
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

```
Or[And[a, b], And[b, c], And[a, c], And[a, b, c]] // FullSimplify // TreeForm
```

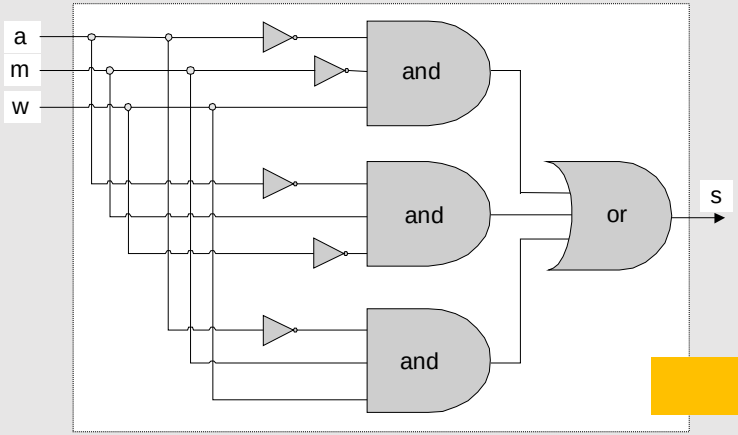


End notes: Canonical representation (cont.)

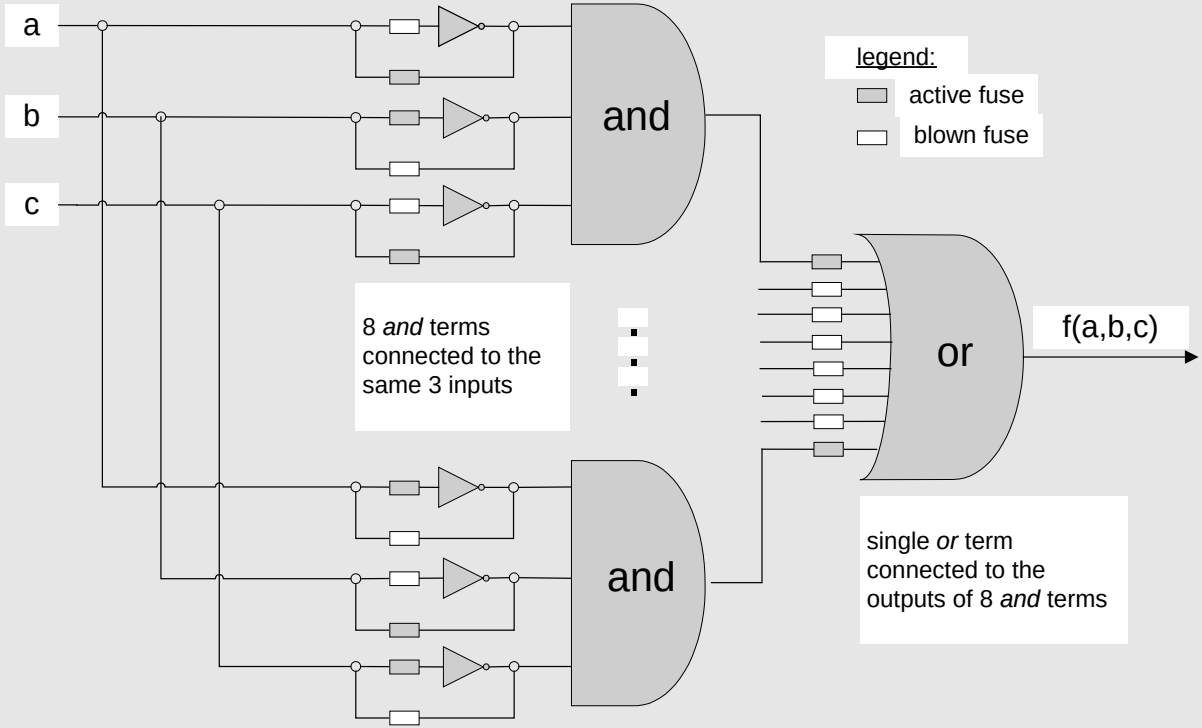
$$s(a,m,w) = \overline{a} \cdot (m + w)$$



$$s(a,m,w) = \overline{a}\,\overline{m}\,w + \overline{a}\,m\,\overline{w} + \overline{a}\,m\,w$$



End notes: Programmable Logic Device for 3-way functions



PLD implementation of $f(a,b,c) = a\,\bar{b}\,c + \bar{a}\,b\,\bar{c}$

(the on/off states of the fuses determine which gates participate in the computation)

Shannon's System Design in Computational Thinking

Design of an Electric Combination Lock

An electric lock is to be constructed with the following characteristics. There are to be five pushbutton switches available on the front of the lock. These will be labeled a, b, c, d, e. To operate the lock the buttons must be pressed in the following order: c, b, a and c simultaneously, d. When operated in this sequence the lock is to unlock, but if any button is pressed incorrectly an alarm U is to operate. To relock the system a switch g must be operated. To release the alarm once it has started a switch h must be operated. This being a sequential system either a stepping switch or additional sequential relays are required. Using sequential relays let them be denoted by w, x, y and z corresponding respectively to the correct sequence of operating the push buttons. An additional time-delay relay is also required due to the third step in the operation. Obviously, even in correct operation a and c cannot be pressed at exactly the same time, but if only one is pressed and held down the alarm should operate. Therefore assume an auxiliary time delay relay v which will operate if either a or c alone is pressed at the end of step 2 and held down longer than time s, the delay of the relay.

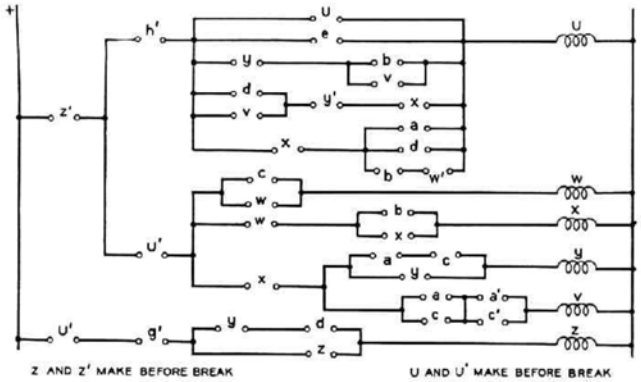


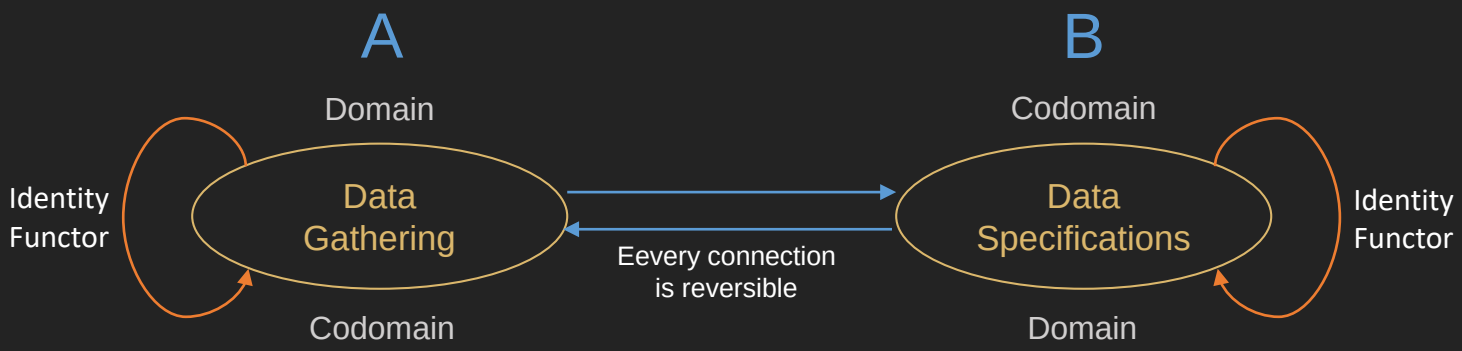
Figure 34. Combination-lock circuit

Combination Lock is a Cryto-security system

XLP 極限學習過程

How Category Theory Simplifies System Development

Can be simplified as :



Infinite possibilities!!! → Find the viable paths to succeed!

03. 合约式设计

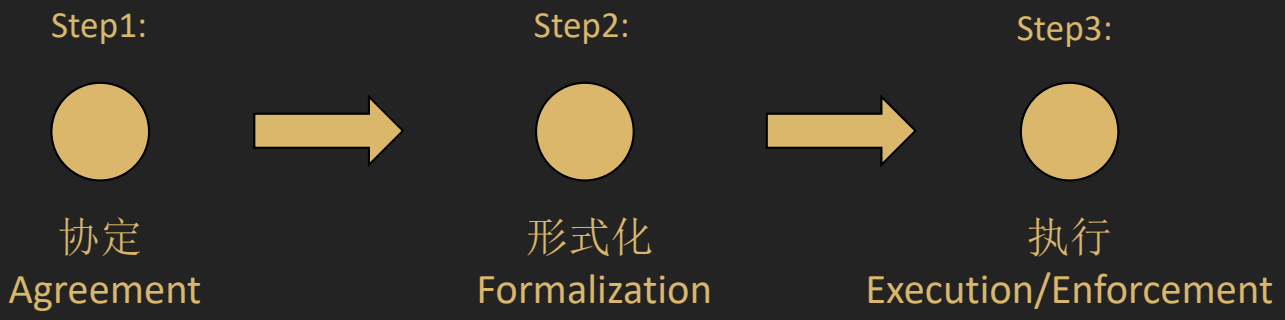
溯源及关键字说明

- 溯源
  - 1995年由多产的跨领域法律学者尼克·萨博（Nick Szabo）提出
  - 定义
  - "一个智能合约是一套以数字形式定义的承诺，包括合约参与方可以在上面执行这些承诺的协议"
  - "合约是一套达成共识的协定, 是形成关系的传统方式"
  - 合约的抽象概念是在个人、机构和他们拥有的东西（财产）之间形成关系的一种公认的工具
- 关键字说明
  - 承諾
    - 合约参与方同意的（经常是相互的）权利和义务
    - 例: 以一个销售合约为典型例子。卖家承诺发送货物，买家承诺支付合理的货款
    - 承诺定义了合约的本质和目的
  - 数字形式
    - 合约必须写入计算机可读的代码中
    - 當参与方达成协定，智能合约建立的权利和义务由一台计算机或者计算机网络执行
  - 协议
    - 合约承诺的技术实现（technical implementation）
    - 选择哪个协议取决于许多因素
    - 最重要的因素是在合约履行期间被交易资产的本质
      - 例: 再以一个销售合约为典型例子，若参与方同意货款以比特币支付，那麼选择的协议就是比特币协议

实践与智能财产

- 嵌入式合约（embedded contracts）
  - "智能合约的基本理念是，许多合约条款能够嵌入到硬件和软件中。"
  - 在物联网時代為嵌入式合約提供了技術基礎
    - 物理实体能够从互联网中检索信息和向互联网发送信息，它们也能够通过软件控制它们自身的使用
  - 实例
    - 物質消費，例如自动贩卖机, 销售点终端等
    - 数字内容消费，例如音乐、电影和电子书领域的数字版权管理机制
  - 从这个意义上理解，**智能合约是赛博空间（虚拟空间）和物理空间（实体空间）之间的桥梁**。

普遍的合约模型

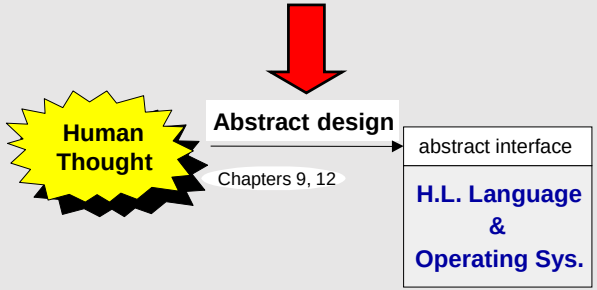


- What are smart contracts in search of a consensus； https://medium.com/@heckerhut/whats-a-smart-contract-in-search-of-a-consensus-c268c830a8ad
- 什么是智能合约？ - 巴比特： http://www.8btc.com/what-are-smart-contracts-in-search-of-a-consensus

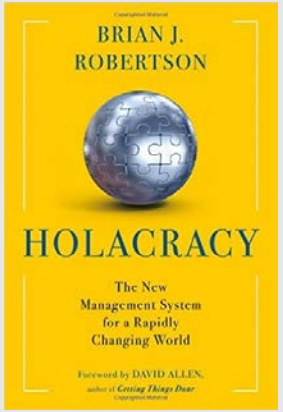
实践与智能财产

- 嵌入式合约（embedded contracts）
  - "智能合约的基本理念是，许多合约条款能够嵌入到硬件和软件中。"
  - 在物联网時代為嵌入式合約提供了技術基礎
    - 物理实体能够从互联网中检索信息和向互联网发送信息，它们也能够通过软件控制它们自身的使用
  - 实例
    - 物質消費，例如自动贩卖机, 销售点终端等
    - 数字内容消费，例如音乐、电影和电子书领域的数字版权管理机制
  - 从这个意义上理解，**智能合约是赛博空间（虚拟空间）和物理空间（实体空间）之间的桥梁**。

The even bigger picture 计算思维与系统设计 （16周）



撰写设计合约



Flexible Organizational Structure

With clear roles and accountabilities

New Meeting Format

Geared toward action and eliminating over-analysis

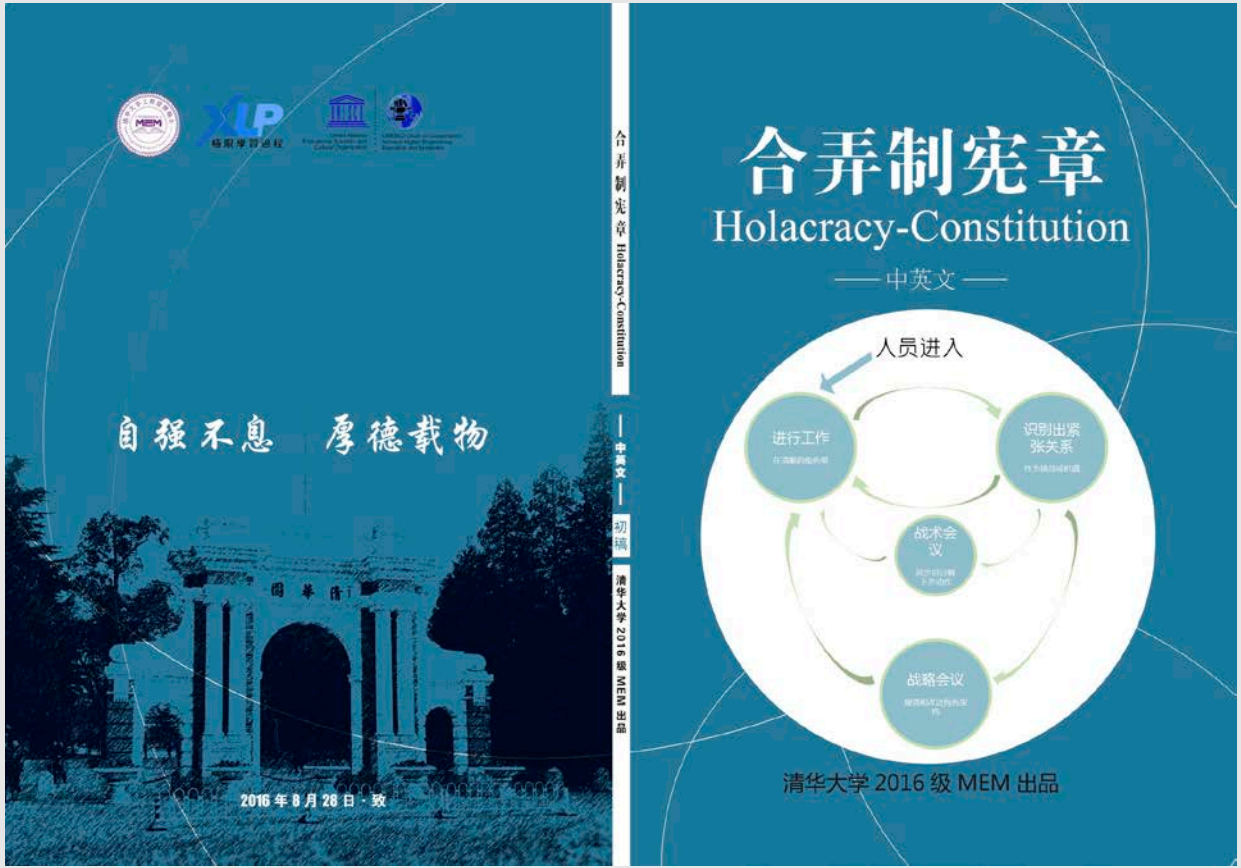
More Autonomy to Teams and Individuals

For individuals to solve issues themselves and cut through bureaucracy

Unique Decision-making Process

To continuously evolve the organization's structure.

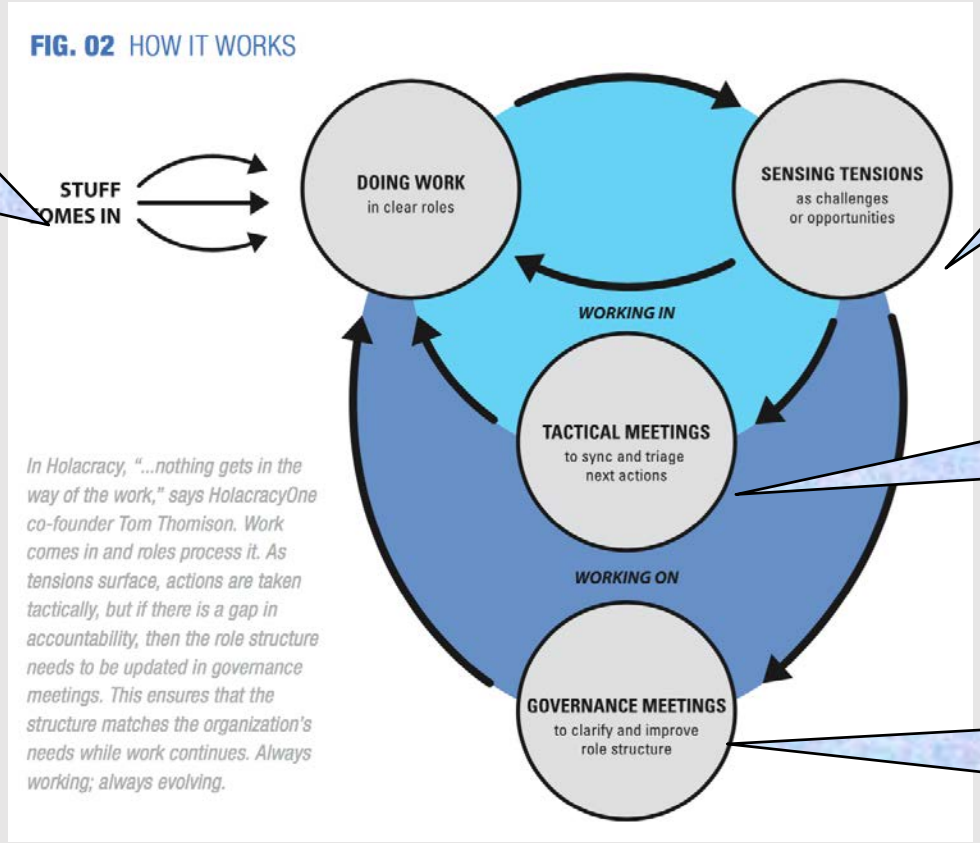
合弄制宪章中文版



http://toyhouse.cc/wiki/index.php/合弄制宪章4.1

团队协同的工作流程 （Workflow of Operating Systems）

任务排程

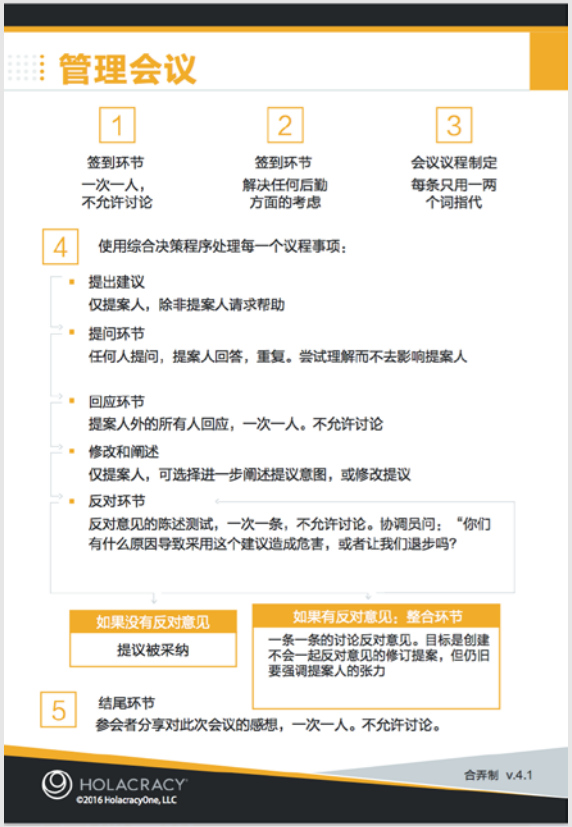


发掘问题

战术会议

治理会议

会议进程规范


管理会议
1
签到环节
一次一人，
不允许讨论
2
签到环节
解决任何后勤
方面的考虑
3
会议议程制定
每条只用一两
个词指代
4
使用综合决策程序处理每一个议程事项：
提出建议
仅提案人，除非提案人请求帮助
提问环节
任何人提问，提案人回答，重复。尝试理解而不去影响提案人
回应环节
提案人外的所有人回应，一次一人。不允许讨论
修改和阐述
仅提案人，可选择进一步阐述提议意图，或修改提议
反对环节
反对意见的陈述测试，一次一条，不允许讨论。协调员问：“你们有什么原因导致采用这个建议造成危害，或者让我们退步吗?
如果没有反对意见
提议被采纳
如果有反对意见：整合环节
一条一条的讨论反对意见。目标是创建不会一起反对意见的修订提案，但仍旧要强调提案人的张力
5
结尾环节
参会者分享对此次会议的感想，一次一人。不允许讨论。
HOLACRACY
©2016 HolacracyOne, LLC
合弄制 v.4.1


战术会议
1
签到环节
一次一人，发现并说出分散你注意力的事情，不允许讨论。
2
清单核查
协调员按照角色宣读反复出现的行动清单，参会者针对前期（例，前一周）反复出现的行动回答“是”或“否”。
3
指标核查
给每个角色发一份简短的指标记录报告，强调最新数据。
4
进站更新
协调员按条宣读每个圈子项目班上的项目，并询问“有没有更新？”项目负责人回答：“无更新”或介绍上次会议后发生的新情况。允许提问，不允许讨论。
5
制定议程
制定解决张力的议程。每项用一两个词指代。
6
诊断张力
解决每一个议题的事项：
二
协调员问：“你需要什么？”
三
议题事项提出者按需求请他人加入
三
记录被要求及接受的下一步行动或项目
四
协调员问：“你的需求得到满足了吗?
7
结束环节
每个人分享对会的感受。不允许讨论
HOLACRACY
©2016 HolacracyOne, LLC
合弄制 v.4.1

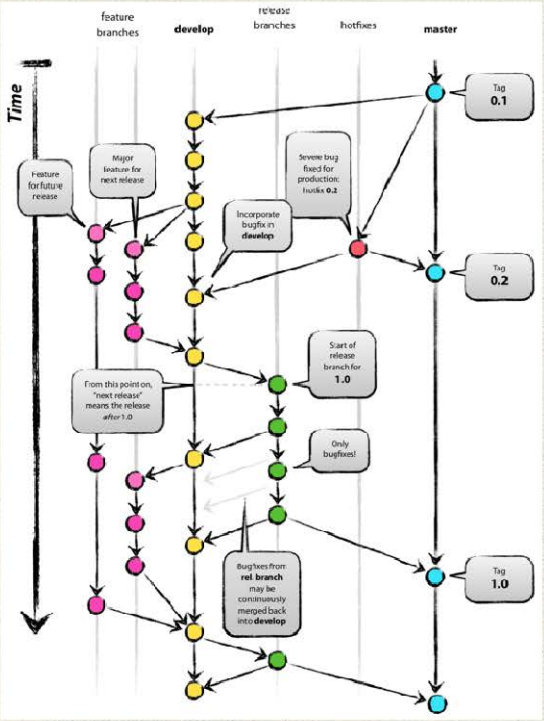


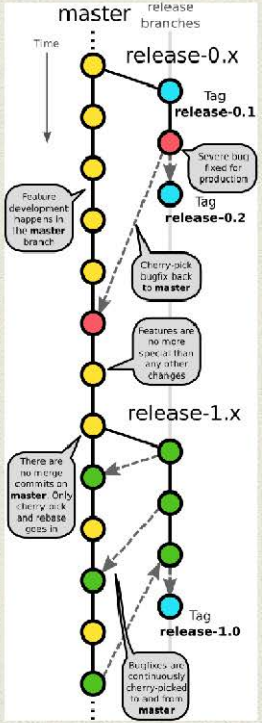
Git与GitHub的工作策略

课道开发培训

叶盛誉

简化的GitFlow流程

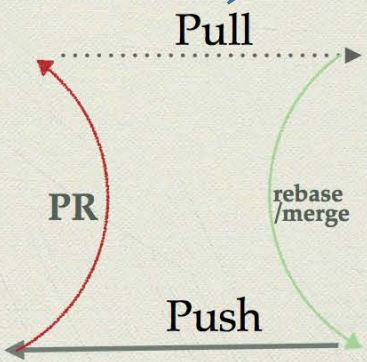




简化的GitFlow流程

这张图像什么？

- 远程仓库
  - 主分支master
  - 发行分支release
  - 个人工作分支
  - 其他分支



- 本地仓库
  - 主分支master
  - 发行分支release
  - 个人工作分支
  - 个人GitFlow分支

Arrows are “mappings” between objects, categories and functors

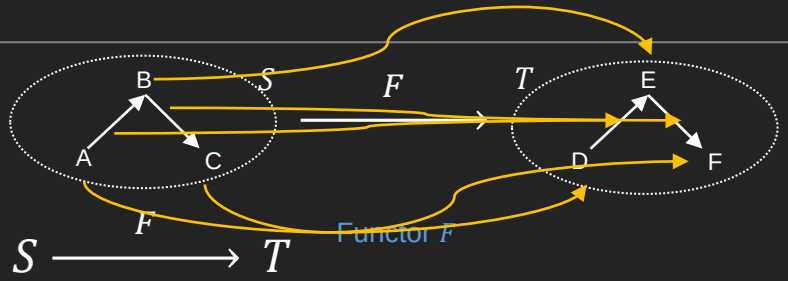
Category theory : *Arrow chasing*

1. Function

   A function *compares two objects*

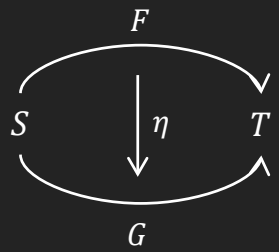
2. Functor

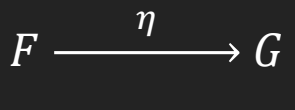
   A Functor *compares two* categories



3. Natural Transformation

   A Natural Transformation compares two Fuctors

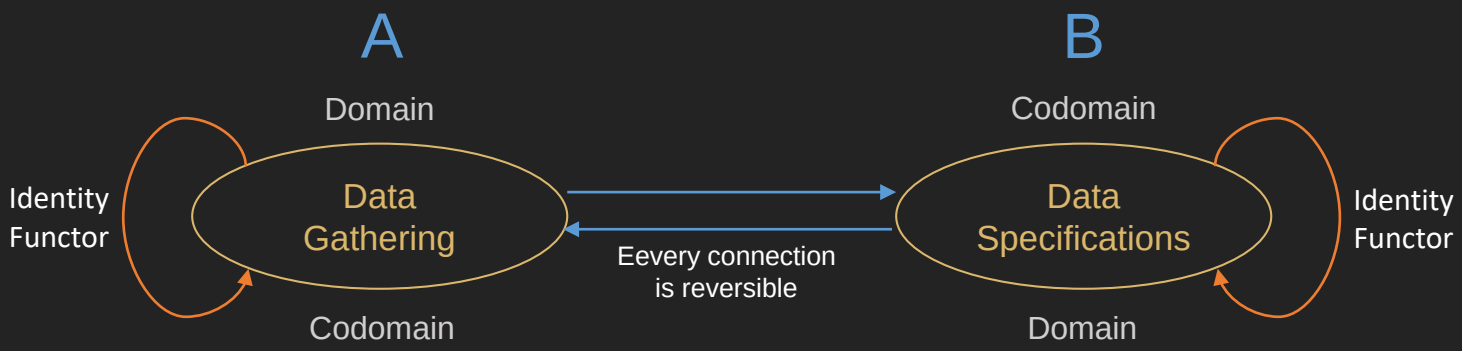




Source : Schalk and Simmons, 2005

How Category Theory Simplifies System Development

Can be simplified as :



Infinite possibilities!!! → Find the viable paths to succeed!

04. 电子设计自动化

电子设计自动化概述

- 背景
  - Electronics Design Automation, EDA
  - 在20世纪60年代中期从计算机辅助设计（CAD）、计算机辅助制造（CAM）、计算机辅助测试（CAT）和计算机辅助工程（CAE）的概念发展而来的。
- 步骤
  - 设计者在EDA软件平台上，用硬件描述语言Verilog HDL完成设计文件
  - 由计算机自动地完成逻辑编译、化简、分割、综合、优化、布局、布线和仿真
  - 由计算机自动地完成对于特定目标芯片的适配编译、逻辑映射和编程下载等工作
- 效果
  - EDA技术的出现，极大地提高了电路设计的效率和可操作性，减轻了设计者的劳动强度。
  - 芯片设计的复杂程度可以得到显著提升
  - 芯片布线布局对人工设计的要求降低，而且软件错误率不断降低
  - 随着计算机仿真技术的发展，设计项目可以在构建实际硬件电路之前进行仿真

- EDA - 百度百科： https://baike.baidu.com/item/eda/182009

发展历史

- 19世纪70年代以前
  - 集成电路的设计、布线等工作只能靠手工完成，当然当时集成电路的复杂程度远不及现在
- 19年代70年代 (第一階段)
  - 开发人员尝试将集成电路的整个设计过程自动化
  - 第一个自动完成电路布线和布局的工具面世
  - 设计自动化会议（Design Automation Conference）成立，旨在促进电子设计自动化的发展

EDA技术发展历史与现状论述： https://wenku.baidu.com/view/d0ee714551e79b8969022670.html

发展历史

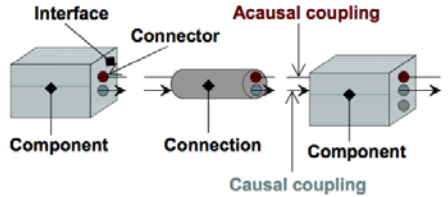
- 19年纪80年代 (第二階段)
  - 卡弗尔·米德和琳·康维于1980年发表的论文《超大规模集成电路系统导论》（Introduction to VLSI Systems）提出通过编程语言来进行芯片设计
    - 集成电路逻辑仿真、功能验证的工具的性能得到相当的改善
    - 芯片设计的复杂程度可以得到显著提升
    - 芯片布线布局对人工设计的要求降低，而且软件错误率不断降低
    - 随着计算机仿真技术的发展，设计项目可以在构建实际硬件电路之前进行仿真
    - 直至今日，通过编程语言来设计、验证电路预期行为，利用工具软件综合得到低抽象级物理设计的这种途径，仍然是数字集成电路设计的基础
  - 1981年起, 电子设计自动化逐渐开始商业化
  - 1984年, 设计自动化会议（Design Automation Conference）上举办了第一个以电子设计自动化为主题的销售展览
  - 1986年, Gateway(1985年成立于美国爱荷华州的IT领域的公司）推出了一种硬件描述语言Verilog (也是现在最流行的高级抽象设计语言)
  - 1987年，在美国国防部的资助下，另一种硬件描述语言VHDL被创造出来。

发展历史

- 19年纪90年代 (第三階段)
  - 技术的发展更侧重于逻辑综合
  - 目前的数字集成电路的设计都比较模块化
    - 设计人员在进行逻辑设计时无需考虑信息单元的具体硬件工艺
  - 硬件语言的标准化进一步确立，电子技术在各个领域的广泛应用
    - 全新的EDA技术的应用和发展得以大力推動
  - 可编程逻辑器件面世
    - 微电子厂家可以为用户提供各种各样的可编程逻辑器件，使得设计者通过设计芯片实现电子系统的功能
    - 设计师逐步从使用硬件转向设计硬件
    - 从单个电子产品开发转向系统级电子产品开发
      - EDA工具以系统级为核心，包括系统行为级描述与系统性能综合、系统仿真与系统测试、系统划分与指标分配等一整套电子系统设计自动化的工具
- 20世纪及以后
  - 仿真验证和设计两个方面都支持标准硬件描述语言的功能强大的EDA软件不断推出
    - 目前最新的发展趋势是将描述语言、验证语言集成为一体
      - 典型的例子有SystemVerilog
  - 更大规模的可编程逻辑器件不断推出
  - 系统级、行为验证级硬件语言趋于更加高效和简单

可计算图：Computable Diagrams

Software Component Model



A component class should be defined *independently of the environment,* very essential for *reusability*

A component may internally consist of other components, i.e. *hierarchical* modeling

Complex systems usually consist of large numbers of *connected* components

Physical Connector

- Classes Based on Energy Flow

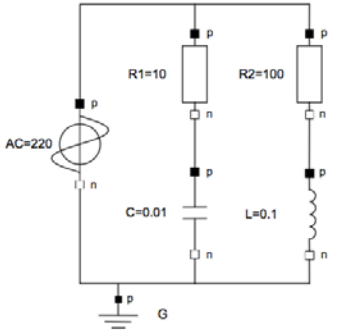
Domain Type	Potential	Flow	Carrier	Modelica Library
Electrical	Voltage	Current	Charge	`Electrical.Analog`
Translational	Position	Force	Linear momentum	`Mechanical.Translational`
Rotational	Angle	Torque	Angular momentum	`Mechanical.Rotational`
Magnetic	Magnetic potential	Magnetic flux rate	Magnetic flux	
Hydraulic	Pressure	Volume flow	Volume	`HyLibLight`
Heat	Temperature	Heat flow	Heat	`HeatFlow1D`
Chemical	Chemical potential	Particle flow	Particles	Under construction
Pneumatic	Pressure	Mass flow	Air	`PneuLibLight`

可计算图形语意：Computable Semantics

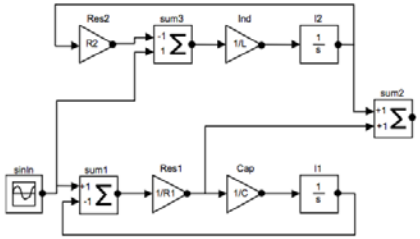
Modelica vs Simulink Block Oriented Modeling Simple Electrical Model

Modelica:
Physical model – easy to understand

Keeps the physical structure

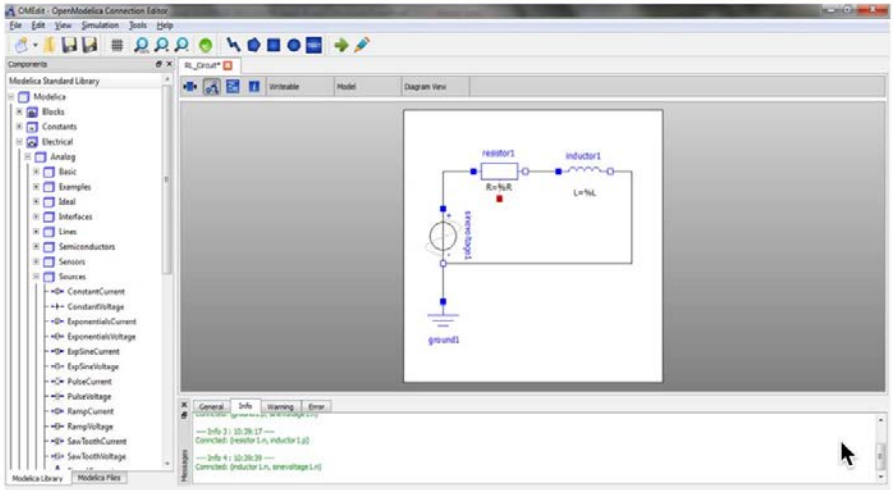


Simulink:
Signal-flow model – hard to understand



MODELICA

Graphical Modeling - Using Drag and Drop Composition



MODELICA