



网络程序设计

郭学理 张 健 等 / 著



内 容 提 要

本书主要目标是使读者掌握网络环境下程序设计的原理和技术。内容主要包括计算机网络基本构成、网络主要协议及通信原理、TCP/IP 协议原理及编程技术、网络计算模型以及 Java, JSP, J2EE, .NET, PHP 等最新网络编程工具、环境、原理和技术。

本书可作为计算机科学与技术、信息安全、电子与通信等专业的本科高年级教材, 研究生参考教材以及工程技术人员的参考用书。

图书在版编目(CIP)数据

网络程序设计/郭学理, 张健等著. —武汉: 武汉大学出版社, 2004. 7

信息安全技术与教材系列丛书

ISBN 7-307-04171-5

I. 网… II. ①郭… ②张…[等] III. 计算机网络—程序设计 IV. TP393

中国版本图书馆 CIP 数据核字(2004)第 025495 号

责任编辑: 黄金文 责任校对: 黄添生 版式设计: 支 笛

出版发行: 武汉大学出版社 (430072 武昌 珞珈山)

(电子邮件: wdp4@whu.edu.cn 网址: www.wdp.whu.edu.cn)

印刷: 湖北省通山县印刷厂

开本: 787×980 1/16 印张: 33 字数: 675 千字

版次: 2004 年 7 月第 1 版 2004 年 7 月第 1 次印刷

ISBN 7-307-04171-5/TP·143 定价: 48.00 元

版权所有, 不得翻印; 凡购我社的图书, 如有缺页、倒页、脱页等质量问题, 请与当地图书销售部门联系调换。

信息安全技术与教材系列丛书

编 委 会

主 任: 沈昌祥(中国工程院院士, 武汉大学兼职教授)

副 主 任: 蔡吉人(中国工程院院士, 武汉大学兼职教授)

刘经南(中国工程院院士, 武汉大学校长)

肖国镇(中国密码学会副理事长, 武汉大学兼职教授)

执行主任: 张焕国(中国密码学会理事, 武汉大学教授)

委 员: 张孝成(江南计算所研究员)

屈延文(国家金卡工程办公室安全组组长, 武汉大学兼职教授)

卿斯汉(中国科学院信息安全技术工程中心主任, 武汉大学兼职教授)

冯登国(信息安全国家重点实验室主任, 武汉大学兼职教授)

吴世忠(中国信息安全产品测评认证中心主任, 武汉大学兼职教授)

朱德生(总参通信部研究员, 武汉大学兼职教授)

覃中平(华中科技大学教授, 武汉大学兼职教授)

谢晓尧(贵州工业大学副校长, 教授)

何炎祥(中国计算机学会常务理事, 武汉大学教授)

何克清(软件工程国家重点实验室副主任, 武汉大学教授)

黄传河(武汉大学教授)

江建勤(武汉大学出版社社长, 教授)

秘 书: 黄金文



序 言

21 世纪是信息的时代,信息成为一种重要的战略资源。信息科学成为最活跃的学科领域之一,信息技术改变着人们的生活和工作方式,信息产业成为新的经济增长点。信息的安全保障能力成为一个国家综合国力的重要组成部分。

当前,以 Internet 为代表的计算机网络的迅速发展和“电子政务”、“电子商务”等信息系统的广泛应用,正引起社会和经济的深刻变革,为网络安全和信息安全开拓了新的服务空间。

世界主要工业化国家中每年因利用计算机犯罪所造成的经济损失远远超过普通经济犯罪。内外不法分子互相勾结侵害计算机系统,已成为危害计算机信息安全的普遍性、多发性事件。计算机病毒已对计算机系统的安全构成极大的威胁。社会的信息化导致新的军事革命,信息战、网络战成为新的作战形式。

总之,随着计算机在军事、政治、金融、商业等部门的广泛应用,社会对计算机的依赖越来越大,如果计算机系统的安全受到破坏将导致社会的混乱并造成巨大损失。因此,确保计算机系统的安全已成为世人关注的社会问题和计算机科学的热点研究课题。

信息安全事关国家安全,事关经济发展,必须采取措施确保信息安全。

发展信息安全技术与产业,人才是关键。培养信息安全领域的专业人才,成为当务之急。2001 年经教育部批准,武汉大学创建了全国第一个信息安全本科专业。到 2003 年,全国设立信息安全本科专业的高等院校增加到 20 多所。2003 年经国务院学位办批准武汉大学建立信息安全博士点。

为了增进信息安全领域的学术交流、为信息安全专业的大学生提供一套适用的教材,武汉大学组织编写了这套《信息安全技术与教材系列丛书》。这套丛书涵盖了信息安全的主要专业领域,既可用做本科生的教材,又可作为工程技术人员的技术参考书。

我觉得这套丛书的特点是内容全面、技术新颖、理论联系实际,努力反映信息安全领域的新成果和新技术。在我国信息安全专业人才培养刚刚起步的今天,这套丛



书的出版是非常及时的和十分有益的。

我代表编委会对丛书的作者和广大读者表示感谢。欢迎广大读者提出宝贵意见,以使丛书能够进一步修改完善。

中国工程院院士,武汉大学兼职教授

沈昌祥

2003年7月28日



前言

在 20 世纪 40 年代及 60 年代,电子计算机和计算机网络相继问世,这是人类发展历史长河中的两大创举。计算机技术和网络技术的发展,使人类在能源、材料、生命科学等领域之外,开发和利用“信息”这一崭新的战略资源,激发人类发展基于新科学、新技术的新手段,引发了人类工作方式和生活方式的巨变,加快了“网络经济”、“知识经济”新时代的到来。

计算机科学技术和网络技术包括其相应的“硬件”和“软件”,而程序设计和网络程序设计即是其“软件”中的主要组成部分和运行表现。程序设计经历了手工编程、汇编语言编程、半高级语言编程和高级语言编程多个阶段;程序设计方法从自顶向下、模块化,发展到面向对象,再到面向构件的阶段;程序设计环境由单机环境发展到多机系统和网络环境。本书主要阐述和研究网络环境下的程序设计方法和技术:首先讲述网络的基本构成、网络定义、TCP/IP 协议原理和实现技术,进而阐述网络计算模型和网络编程环境,最后详细介绍和分析网络程序设计的最新语言、工具、原理和技术,其中包括 Java, JSP, J2EE, .NET 和 PHP 等。

本书章节内容安排如下:

第一章至第二章为基础篇,对网络以及网络程序设计的基本原理进行了介绍。第一章网络程序设计概述,阐述了网络程序设计的概念,同时通过对程序设计方法及其发展的介绍,让读者在网络程序设计中可以采用正确的方法;第二章为网络基础,对计算机网络的定义、原理、协议、体系结构等进行了介绍,同时给出了本书中各示例程序的网络编程试验环境。

第三章至第七章为代码分析篇,其内容是对 Linux 下网络代码的分析,其主要的目的在于让读者了解 Internet 上的主要协议 TCP/IP 在 UNIX 下的具体实现,在学习网络编程中能够具有更为透彻和深入的认识。第三章 Linux 源代码概述对 Linux 下 TCP/IP 协议族的基本体系结构进行了介绍,并为第五章至第七章的协议分析内容进行了安排;第四章 Linux 网卡驱动程序分析介绍了 Linux 网络设备驱动程序的基本原理,并以目前广泛使用的 Realtek 8139 网卡驱动程序为例分析了 Linux 网卡的工作方式;第五章为 Linux IP 协议分析与编程,通过对 IP 层协议及其 Linux 下代码实现的分析,深入介绍了 UNIX 下的原始套接字编程;第六章 Linux TCP 协议分析与编程对 TCP 协议内容进行了较为详细的介绍,以 TCP 协议的客户机/服务器模型为基础分析了 Linux 下的 TCP 代码实现,最后介绍了 UNIX 下的 TCP 套接字编程方法;第七章

Linux UDP 协议分析与编程通过对 UDP 协议及其代码实现的分析,介绍了 UNIX 下的 UDP 编程方法与技巧。

第八章至第十三章为编程技术篇,主要介绍目前网络程序设计中主流的编程技术;第八章网络计算模型介绍了网络计算模型的概念及其演变历程,讨论了各种相关的中间件技术;第九章跨平台的编程语言(JAVA)对 Java 语言及其编程方法进行了介绍,强调了利用 Java 语言进行网络程序设计的方法;第十章为网站平台的主流开发技术(JSP),介绍了 JSP 基本语法和 JSP 的内部对象、JSP 与 XML 的交互技术等内容,让读者能够掌握使用 JSP 搭建简单的电子商务平台;第十一章为多层应用模式的主流技术(J2EE),通过对 J2EE 平台的分析,介绍了如何使用 J2EE 进行网络程序设计;第十二章 Windows 平台的主流开发平台(.NET)主要讨论了 Microsoft 新提出的基于 Windows 操作系统的网络应用程序主流开发平台.NET;第十三章为动态主页编程利器(PHP),介绍了 PHP 的基本语法,并详细探讨了利用 PHP 实现动态主页的编程技巧。

本书由郭学理任主编,负责全书的总体策划、审稿和最终统稿;张健编写第一章至第七章;刘茂福编写第八章、第九章、第十二章;刘靖编写第十章至第十一章;张文涛编写第十三章。

在教材的编写过程中,大量参考与借鉴了已出版的各类教材以及 Internet 上发布的各种资料,在这里向这些参考文献的作者表示感谢,尤其是《程序设计方法学》的作者陈世鸿老师、计算机网络电子教材的作者王祖荣老师,感谢他们在基础篇中提供的相关内容。

由于本书的编写工作较为仓促,同时编者水平有限,难免存在疏漏和错误,敬请广大读者批评指正。

作 者

2003 年 12 月于珞珈山

目 录

第一章 网络程序设计概述	1
1.1 程序设计发展	1
1.2 程序设计方法	4
1.3 程序设计方法沿革	18
1.4 网络程序设计发展	26
1.5 章节介绍	29
第二章 网络基础	31
2.1 网络概念	31
2.2 网络拓扑结构与传输介质	34
2.3 网络体系结构与协议	43
2.4 通信原语	55
2.5 编程环境	58
2.6 小结	59
第三章 Linux 网络源代码概述	60
3.1 Linux 下 TCP/IP 协议实现结构	60
3.2 Linux 下 TCP/IP 启动	65
3.3 相关数据结构	67
3.4 Linux 网络源代码分析安排	79
第四章 Linux 网卡驱动程序分析	80
4.1 网卡驱动程序概述	80
4.2 网卡初始化	82
4.3 网卡的打开与关闭	89
4.4 数据包的发送与接收	91
4.5 相关函数与结构	103
4.6 小结	105

第五章 Linux IP 协议分析与编程	106
5.1 IP 协议概述	106
5.2 Linux 中 IP 协议分析	114
5.3 IP 协议编程	132
5.4 小结	148
第六章 Linux TCP 协议分析与编程	149
6.1 TCP 协议概述	149
6.2 Linux 中 TCP 协议分析	160
6.3 TCP 协议编程	177
6.4 小结	195
第七章 Linux UDP 协议分析与编程	196
7.1 UDP 协议概述	196
7.2 Linux 中 UDP 协议分析	198
7.3 UDP 协议编程	213
7.4 小结	220
第八章 网络计算模型	221
8.1 网络计算模型简介	221
8.2 通用数据库应用程序接口 ODBC	228
8.3 中间件技术	241
8.4 数据库中间件 JDBC	251
8.5 小结	264
第九章 跨平台的程序设计语言——Java	265
9.1 Java 语言概述	265
9.2 Java 中的面向对象技术	271
9.3 Java 应用程序	280
9.4 Java 网络应用程序设计	288
9.5 Java 网络通信程序设计	295
9.6 小结	308
第十章 网站平台的主流开发技术——JSP	309
10.1 JSP 概述	309

10.2 JSP 平台的搭建	312
10.3 一个简单的 JSP 实例	313
10.4 JSP 基本语法	314
10.5 JSP 内置对象详解	332
10.6 JSP 和 XML	346
10.7 JSP 数据库编程	351
10.8 JSP/Servlet 网络程序设计	354
10.9 小结	360
第十一章 多层应用模式的主流技术——J2EE	362
11.1 J2EE 有关概念和技术	362
11.2 客户层	367
11.3 Web 层	379
11.4 EJB 层	382
11.5 小结	400
第十二章 Windows 主流开发平台——.NET	401
12.1 .NET 概述	401
12.2 .NET 基本框架	412
12.3 .NET 类库框架	425
12.4 小结	435
第十三章 动态主页编程利器——PHP	437
13.1 PHP 概述	437
13.2 PHP 基本语法	449
13.3 PHP 网络程序设计	495
13.4 小结	512
参考文献	513



第一章 网络程序设计概述



网络程序设计是在网络环境下进行程序设计的课程,该课程的学习不仅局限于某一类特殊程序设计语言的编程实现,而且还要在各种网络程序语言的应用中体现出网络的特性。在该课程中,网络的特性、协议、应用占据了主导地位,而各种网络编程语言只是网络功能的实现工具。在本书的编写过程中,贯彻了“网络程序设计 = 网络 + 程序设计”这一指导思想,在对网络各层的协议进行详细分析的基础上,介绍了各种编程语言对协议或应用的具体实现。

1.1 程序设计发展

程序设计是一门艺术,是一门建立在严格的规范基础上而又充满了灵活性的艺术。1946年2月14日,第一台计算机 ENIAC 在美国宾夕法尼亚大学诞生,继而“存储程序式”计算机 EDSAC, MARKI, WHIRLWIND 等被研制成功,计算机就开始在各行各业发挥出巨大的作用,但程序设计在这一阶段则是枯燥无味的工作,其重要性远远低于硬件设计;1957年,高级编程语言 FORTRAN 的问世将程序设计推向更高的层次,同时结构化编程思路的应用也导致了程序设计黄金时代的到来;20世纪90年代至今,在程序设计领域,不断涌现出新的程序设计语言、程序设计方法与思想,展现在我们眼前的是一片欣欣向荣的景象,而程序设计的发展也开始出现纵深化、专业化,在不同的行业中发挥着特殊的作用。

在程序设计的发展史中,存在着两条主线,一条主线是程序设计语言从机器语言、汇编语言发展到高级语言;而另一条主线是程序设计方法由传统的面向问题处理过程的结构化程序设计方法,发展到面向对象的可视化程序设计方法,进而产生基于分布式软件模型的面向组件的程序设计方法,同时面向服务的程序设计方法和语言正处于研发之中。

1.1.1 基本概念

大多数人都认为,软件就是用以指挥计算机运行活动所使用的程序,其实软件的精确定义是“软件 = 程序 + 文档”,尤其是文档,在软件开发的定义、设计、编码和维护等阶段发挥着重要的作用。软件分为系统软件和应用软件,应用软件用来完成面向用户的某些应用,而系统软件更为通用,通常是独立于应用的,用于支持基本的计



计算机功能以及所有的应用领域。

所谓程序,简单地说是—系列指令的有序组合,计算机通过运行该组指令,达到预期的目的。而从一个问题的提出,程序的编制修改,一直到程序的正确运行的整个过程就是程序设计。由于计算机的操作主要是对各种类型的数据进行处理,因此,程序中还必须对操作的对象——数据进行—定的安排,数据在计算机中的组织称为数据结构,而操作步骤的描述又称为算法,所以我们可以进一步理解为:程序 = 数据结构 + 算法。

程序设计就是按照—定的目的组织数据和安排对数据操作、处理的步骤,以及创建合理有效的用户界面的过程。—个完整的程序设计过程大体分为—下几个步骤:① 清晰描述程序的基本功能;程序编写之前,先要明确程序的基本功能目的,同时通过程序流程图等方法对功能实现进行设计。② 用程序语言书写预定的算法方案;在选择了程序开发语言后,按照设计的程序流程图等进行编程实现。③ 证明程序的正确性,编写完毕的程序需要与设计进行比较,检查编程实现是否忠于设计,在必要的时候应该证明程序流程单入单出、不存在死循环、可在有限步骤内运行结束。④ 测试和修改程序;测试是程序设计的重要阶段,测试方法分为黑盒测试和白盒测试,通过测试可以发现程序中出现的错误并及时修改。

1.1.2 程序设计语言

程序设计语言泛指—切用于书写计算机程序的语言,包括机器语言、汇编语言,以及—般称为高级语言的完全符号形式的独立于具体计算机的语言。

程序设计语言具有明显的层次性或级别特性,可分为低级语言和高级语言,低级语言又分为机器语言与汇编语言。“级”的概念是指程序员与计算机对话的复杂程度,程序设计语言分类的级别与关系如图 1-1 所示。通常情况下,程序员在使用高级语言进行程序设计时只需涉及计算机“做什么”,而不必涉及计算机“怎样做”,而软件工具将高级语言的程序代码转换为低级语言代码。低级语言则需要告诉计算机“怎样做”,要求程序员详细说明由计算机执行的每一次操作。在使用高级程序设计语言时,程序员不必记住数据在主存中的存储单元;而在使用低级语言时,程序员必须在程序中提供相应的逻辑,以便监视数据在主存中的物理位置。如图 1-1 所示,利用高级程序语言编写的程序可以在编译工具的帮助下,直接生成目标机器语言;也可以由代码生成器首先生成中间代码,该代码可采用汇编语言,再生成机器的目标代码。

机器语言是由机器指令和运算数据所构成的二进制串语言,只有机器语言可以直接在计算机上运行,所有其他语言最终都要被编译成为机器语言执行。

汇编语言是指借助于符号而不是二进制串语言来表示指令的语言,所以又称为符号语言,其最大的特点就是存在—个与机器指令—一对应的指令系统。汇编语言增加了程序的可记忆性和可理解性,—般来说,—条汇编语句都直接被编译为对应

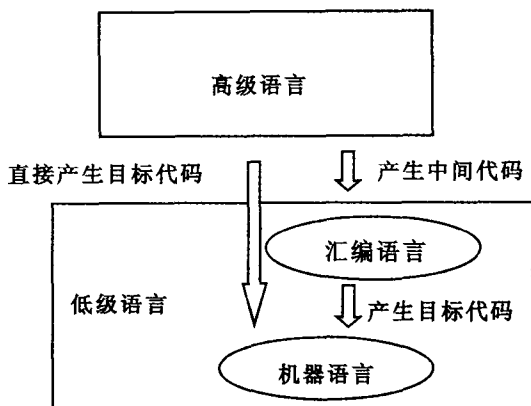


图 1-1 程序设计语言层次与关系

的机器指令或简单的指令段。

在 20 世纪 70 年代之前,应用程序开发多采用机器和汇编语言,特别是系统软件的开发,当时的程序设计人员认为低级语言编写的程序运行效率高,但是利用低级语言的开发工作较为繁琐,同时无法进行代码优化,容易产生重复代码和无用代码。但是,随着高级语言和结构化编程思想的产生,编程已经成为一项具有严格规范和 workflows 的工作,同时高级语言在程序编写和功能叙述上和自然语言较为相似,程序便于阅读和理解,编写效率高;最重要的是函数、过程的大量应用,代码编译优化技术的不断提高使得高级语言程序在编译成为机器语言后更为简洁、代码重用率更高;现代的程序设计中,除非在系统底层实现和部分基于特殊硬件的编程中仍然使用低级语言,大多数程序设计工作都是使用高级语言实现的。

高级语言是根据所求解问题的算法,写出处理过程即可获得运行结果的语言,采用类似自然语言的组织方法和符号方式实现程序设计,所以又称为算法语言,高级语言多具有明显的面向过程特性,功能强大,便于表达特定的要求和控制,并且开始向面向对象、可视化和构件化方向发展。高级程序语言的分类以及相互之间的关系,如图 1-2 所示。

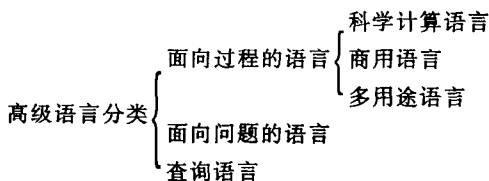


图 1-2 高级语言分类



面向过程的语言——面向过程的语言(POL)较为灵活,可以用来刻画几乎所有科学的和商业的过程。其典型的特点就是程序顺序地编写指令,系统根据用户的规定进行处理。除非程序逻辑控制流程的跳转,否则程序中的每一条指令都是按顺序执行的。

面向过程的语言分为科学计算语言、商用语言与多用途语言。科学计算语言是代数或公式化的语言,用于进行典型的科学计算处理要求(包括矩阵操作、精度计算等),第一个最为流行的科学计算语言是 FORTRAN;商用语言面向商业应用,具有循环周期处理(如单据的反复打印)和大量数据处理的能力,COBOL 是其中的佼佼者;多用途语言是具有商业语言和科学计算语言两者特色的语言,可以在多种应用环境中使用,最早出现的是 PL/1(程序计算语言/1),随后涌现出 PASCAL,ADA,C 等多种多用途语言。值得一提的是,在语言的发展中,获“计算机的诺贝尔奖”——即图灵奖的程序设计语言有 ALGOL60,APL,C,FORTRAN,LISP,ML,PASCAL,SIMULA 等。

面向问题的语言——面向问题的语言是为了实现某种特定应用或解决特定问题,具有便于做统计、字处理和灵活性的特点。然而,一种面向问题的语言总是局限于某种应用。目前已经存在几十种应用的面向问题的语言,包括离散和连续模拟(例如,GPSS,SIMSCRIPT,GASP-IV),程控机器刀具(例如 APT)等。

查询语言——查询语言利用高级的、类似自然语言的命令来检索和编排满足管理查询和制表要求的数据。用查询语言可以交互式地完成一次询问(直接与计算机通信),执行程序产生的结果直接显示在终端上。其特色包括类似于自然语言的命令,对数据有限的数学运算操作,对报表的自动编排、排序以及按关键字挑选记录等。典型的查询语言如关系数据库管理系统中使用的标准查询语言——SQL(Standard Query Language)。

1.2 程序设计方法

高质量的软件设计必须依赖于严格的程序设计的基本理论,在程序设计方法学这一小节中,通过对程序设计方法基本理论的了解,使读者能够在网络程序设计中运用正确有效的设计方法设计出更为优秀的软件。

对程序设计方法学的介绍分为控制、基本数据结构与程序三个主要部分。控制部分介绍了几种基本的控制模型及其相互之间的关系;在基本数据结构部分,着重讨论了数据的数学模型、数据类型的定义和表示方法等;程序部分则介绍了算法设计的相关问题。

由于程序设计方法学不属于本课程的主要内容,本节的文字借鉴了国内外较为成熟的教材,详细内容见参考文献[1]。



1.2.1 控制部分

控制部分在程序运行过程中控制信息经历的路径,是较为重要的部分。控制信息沿着选定路径,引导计算过程从一个路径节点到达另一个节点,也使计算状态到达另一个状态。

控制部分主要讨论有限自动机、下推自动机和图灵机三种数学模型。

1. 有限自动机模型

有限自动机(FA),又称为有限状态机(FSM)或时序机,可以理解为一台机器或一种控制机构,其目的是为了自动仿效一个事先确定的操作系列或响应一条已编码的指令。对有限自动机的理解如图 1-3 所示,一个简单的有限自动机主要由输入、处理和输出三个部分构成,对其进行细化又可以分为五个元素:

- (1) 一个带输出的有限状态集, $Q = \{q_0, q_1, \dots, q_n\}$, q_0 称为初始状态;
- (2) 一个形成输入符号串的字符表, $\Sigma = \{a, b, \dots\}$;
- (3) 一个可能的输出符号表, $\Gamma = \{x, y, \dots\}$;
- (4) 一个转换表,指定对于一个状态,当输入一个字符时,应该转移到的下一个状态;
- (5) 一个输出集,指定对于一个状态,当输入一个字符时,应该输出 Γ 中的某个字符。

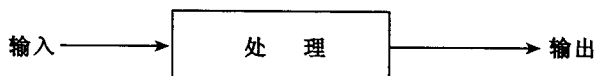


图 1-3 有限自动机组成

图 1-4 是一个典型 FA 的状态转换图(TG),在状态转换图中包含了转换表和输

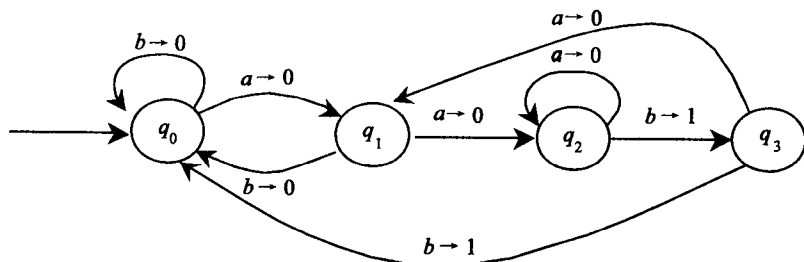


图 1-4 有限自动机状态图



出集两部分内容,输入符号表 $\Sigma = \{a, b\}$, 输出符号表 $\Gamma = \{0, 1\}$; 该 FA 的作用在于统计任何一个字符串 $\in \Sigma^*$ 中出现的子串 aab 的个数, 每当字符串中出现一个 aab 时输出 1, 否则输出 0。

有限自动机数学模型是一个四元组, 表示为 $FA = (Q, q_0, NEXT, OUT)$, 其中:

(1) $Q = \{q_0, q_1, \dots, q_n\}$ 是 $n+1$ 个状态的集合;

(2) $q_0 \in Q$ 是初始状态;

(3) $NEXT$ 是状态转换函数, 定义为 $NEXT: Q \times \Sigma \rightarrow Q$, $Q \times \Sigma$ 是集合 Q 和输入字母表 Σ 的笛卡尔积;

(4) OUT 是输出函数, 定义为 $OUT: Q \times \Sigma \rightarrow \Gamma$, Γ 是输出字母表。

任意一个符号串的集合称为一个语言, 在一个有限字符集 Σ 上, 能够定义无数个语言的语言集 $\mathcal{R} = \{L | L \subseteq \Sigma^*\}$, 其中只能有一部分能被 FA 所识别, 记为 $\mathcal{R}_{FA}$, 是 $\mathcal{R}$ 的一个真子集。正则表达式 RE 是一个公式, 是 FA 的另一种表示方法, 它规定了满足该公式的所有符号的集合, 因此一个 RE 也表示一个语言, 语言的句子可以用 RE 表示, RE 是语言中句子的文法, 或称语法。

例如, 设字母表 $\Sigma = \{a, b\}$, 语言 L 是 Σ 上长度为 3 的字符串的集合, 表示为

$$L = \{aaa, aab, aba, baa, abb, bab, bba, bbb\},$$

L 的正则表达式为

$$RE = (a+b)(a+b)(a+b),$$

也可以表示为

$$RE = (a+b)^3.$$

有限自动机 (FA)、状态转换图 (TG) 和正则表达式 (RE) 都可以用于表示或接收一个语言, 这三者在语言的识别能力上是等价的。但是 FA 能够识别的语言是有限的, 例如, FA 就无法识别 NMN 语言, NMN 语言的定义为 $NMN = (1^n 0^m 1^n | n, m > 0)$ 。

2. 下推自动机模型

FA 是最简单的控制模型, 所能识别的语言是有限的, 主要原因在于 FA 极其有限的记忆能力, 在 FA 中记忆是靠状态自身实现的, 而下推自动机具有较强的语言识别能力。

所谓下推自动机 (PDA), 简单地说就是由一个 FA 和一个栈组成。一个下推自动机 (PDA) 是一个连通有向图, 有向图的节点和有向边的定义如下:

(1) $Q = \{q_0, q_1, \dots, q_k\}$, 每个状态就是一个节点, 其中:

① $q_0 = \text{START}$, 仅有输出边, 以空转换 “ $\varepsilon \rightarrow$ ” 标记, 没有输入边。

② 存在一个终止状态集 $F = F_1 \cup F_2$ 。若 $q_i \in F_1$, 则 $q_i = \text{ACCEPT}$, 表示输入字被接受; 若 $q_i \in F_2$, 则 $q_i = \text{REJECT}$, 表示输入字被拒绝。

③ 在 Q 中存在多个若干 PUSH 状态, 该状态将输入字母压入栈顶, 其输出边是空转换 “ $\varepsilon \rightarrow$ ” 到达下一个状态; 同时, 存在若干 POP 状态, 其输出边标记 “ $x \rightarrow$ ”, $x \in$