计算机科学与技术方法论

董荣胜　古天龙　著

人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS

计算机科学与技术方法论

董荣胜　古天龙　著

人 民 邮 电 出 版 社

内 容 提 要

本书是作者多年来对计算学科方法论研究成果的总结。作者根据《计算作为一门学科》报告对整个计算学科综述性导引课程的严密性和挑战性的要求，借鉴了数学的公理化思想，对计算学科的主要内容进行了系统化、逻辑化的概括，并通过大量实例，深入浅出地阐明了计算学科中各主领域发展的基本规律，揭示了各主领域之间的内在联系，有助于人们对计算学科的深入了解。

本书的主要内容包括：计算机科学与技术方法论的构建，计算学科的历史、定义、根本问题，计算学科各主领域的基本问题，计算学科中的抽象、理论和设计 3 个学科形态，计算学科中的核心概念、数学方法、系统科学方法、形式化技术、社会和职业的问题等。为了使读者能更好地理解和掌握书中的内容，在各章末还附有一定数量的思考题。

本书是计算学科认知领域的一本学术专著，也可作为高等院校计算学科方法论、计算机导论等课程的教材或参考书，还可供相关专业的学生、教师和科技人员参考。

目　录

第1章　绪　论

今天，计算（Computing）技术作为现代技术的标志，已成为世界各国许多经济增长的主要动力，计算领域也已成为一个极其活跃的领域。计算学科正以令人惊异的速度发展，并大大延伸到传统的计算机科学的边界之外，成为一门范围极为宽广的学科。如何认知这个学科，引发了长期以来激烈的争论，并极大地影响着计算学科的发展和人才的培养。

要解决学科的认知问题，必须有一套科学的方法。就哲学方法论而言，学科方法论就是认知学科的方法和工具。它有助于人们对学科认识的逻辑化、程序化、理性化和具体化。一般来说，学科方法论的建立也是学科成熟的标志之一。

1984 年 7 月，美国计算机科学与工程博士单位评审部的领导们，在犹他州召开的会议上对计算认知问题进行了讨论。这一讨论以及其他类似讨论促使（美国）计算机协会（Association for Computing Machinery，简称 ACM）和（美国）电气和电子工程师学会计算机分会（Institute of Electrical and Electronics Engineers-Computer Society，简称 IEEE-CS）联手组成攻关组，开始用新的思维方式来理解计算学科。此后，ACM 和 IEEE-CS 以及不少学者在计算学科认知问题上做了大量富有成效的工作，发表了一系列报告和论文。现在，人们对计算学科的认知已相当成熟，从而使我们能够以一般科学技术方法论为指导，科学地建立起计算学科自己的方法论—计算机科学与技术方法论。

本书所建立的计算机科学与技术方法论，借鉴了数学的公理化思想来整理和总结计算学科，因此，它又可以作为计算学科中的一门严密而又能将学生引入学科各个富有挑战性领域的综述性导引课程，显然，这样的综述性导引课程现已自然地演变为计算学科方法论这样的课程了。

1.1　计算机科学与技术方法论产生的历史背景

计算学科源于欧美，诞生于 20 世纪 40 年代初。计算学科的理论基础可以说在第一台现代电子计算机出现以前就已经建立起来了，20 世纪 40 年代数字计算机产生后，促进了计算机设计、程序设计以及计算机理论等领域的发展。

早在 1943 年，英国的一台名叫“巨人”（Colossus）的计算机就投入了运行，用于译解德国密码，但由于英国政府 1970 年之前一直对它保密，人们无法了解，因此，一般认为，美国宾夕法尼亚大学于 1946 年 2 月 14 日研制成功的 ENIAC（Electronic Numerical Integrator and Calculator，电子数字积分器和计算器）是世界第一台多功能、全电子数字计算机。

1.1.1　早期关于“计算机科学”名称的争论

最早的计算机科学学位课程是由美国普渡大学于 1962 年开设的。随后，斯坦福大学也开设了同样的学位课程。但针对“计算机科学”这一名称，在当时引起了激烈的争

论。毕竟，当时的计算机主要用于数值计算，因此，大多数科学家认为使用计算机仅仅是编程问题，不需要作任何深刻的科学思考，没有必要设立学位。另外，很多人还认为，计算机从本质上说是一种职业而非学科。

1.1.2 《计算作为一门学科》报告的主要成果及其局限性

20 世纪 70～80 年代，计算技术得到了迅猛的发展，并开始渗透到大多数学科领域，但以往激烈的争论仍在继续。计算机科学能否作为一门学科？计算机科学是理科还是工科？或者只是一门技术、一个计算商品的研制者和销售者？

针对激烈的争论，1985 年春，ACM 和 IEEE-CS 联手组成攻关组，开始了对“计算作为一门学科”的存在性证明。经过近 4 年的工作，ACM 攻关组提交了《计算作为一门学科》（*Computing as a Discipline*）的报告，完成了这一任务。该报告的主要内容刊登在 1989 年 1 月的《ACM 通讯》（*Communications of the ACM*）杂志上。

1. 《计算作为一门学科》报告的主要成果

ACM 攻关组提交的报告得到了 ACM 教育委员会的认可并批准发行。该报告取得了以下 3 个重要成果：

（1）第一次给出了计算学科一个透彻的定义，回答了计算学科中长期以来一直争论的一些问题，完成了计算学科的“存在性”证明（Existence Proof）。

（2）在提出和解决计算教育中的第一个重大问题——计算作为一门学科的同时，还提出了未来计算教育必须解决的第二个重大问题——整个学科核心课程详细设计及第三个重大问题——整个学科综述性导引（导论）课程的构建。报告要求该综述性导引课程能以严密（Rigorous）的方式将学生引入整个学科各个富有挑战性的领域之中。

（3）给出了计算学科二维定义矩阵的定义及相关研究内容，为最终用“新的思想方法”解决计算教育中的第三个重大问题奠定了一定的基础。

2. 《计算作为一门学科》报告的局限性

《计算作为一门学科》报告所取得的第三个成果是它对学科的最重要贡献。而对计算学科二维定义矩阵及相关内容的研究，其实就是报告中所称的找到了一个思考我们学科的“知识框架”（Intellectual Framework），而非对策；一个指导方针，而非（具体）指示。进一步而言，“知识框架”及其研究的实质其实也就是报告中所指的“新的思想方法”。这个“新的思想方法”是对计算领域几十年来的概括和总结，其目标就是要构建起计算学科自己的方法论（尽管报告没有认识到这一点）。此后，学术界不少学者自觉或不自觉地在这种“新的思想方法”的基础上对计算学科方法论展开了研究。

根据报告关于“学科划分”的介绍，我们不难得出这样的结论：专家们在工作中并没有真正认识到“新的思想方法”的本质，只不过是在不自觉中遵循了一般科学技术方法论的思想。在学科的划分问题上，报告是这样说的：专家们颇费心力，最初他们偏向于选择“模型”（Model）与“实现”（Implementation）相对，以及“算法”（Algorithm）与“机器”（Machine）相对，这两种方案无疑都可以反映计算学科研究的基本内容，但是这两个方案不是太抽象就是彼此的界限太模糊，大多数人很难认同这种划分方法。后来，专家们认识到计算学科的基本原理已被纳入理论、抽象和设计 3 个过程中，学科的各分支领域正是通过这 3 个过程来实现它们的目标，因此，最后选择了抽象、理论和设计 3 个过程的内容作为“新的思想方法”的基本内容。

抽象、理论和设计是一般科学技术方法论最基本的研究内容，遗憾的是，该报告没有认识到这一点，从而未能科学地、完整地建立起计算学科的方法论，不仅如此，该报告在强调 3 个过程是错综复杂地缠绕在一起的时候，却忽视了另一个更为重要的内容，即 3 个过程的内在联系，从而在一定程度上削弱了人们对该报告的理解。

1.1.3 CC1991 的主要成果

1990 年，ACM 和 IEEE-CS 联合攻关组在《计算作为一门学科》报告的基础上提交了关于计算学科教学计划的 *Computing Curricula 1991*（以下简称 CC1991）报告，该报告的主要成果是：

（1）提取了计算学科中反复出现的（具有方法论性质的）12 个核心概念。

（2）“社会的、道德的和职业的问题”主领域的提出，使计算学科方法论的研究更加完备。

然而，由于 CC1991 没有解决计算教育中的第二和第三个重大问题，并且与“计算作为一门学科”报告相比，没有重大的突破，因此，雄心勃勃的 CC1991 教学计划的执行远没有达到它的预期目标。尽管如此，它仍然对计算学科教育产生了很大的影响，并使一些学者开始考虑如何构建计算学科方法论的问题。

1.1.4 CC2001 的主要成果

1998 年秋，IEEE-CS 和 ACM 联手组成任务组，开始了关于计算学科教学计划的 *Computing Curricula 2001*（以下简称 CC2001）的起草工作。经过 3 年多的工作，任务组于 2001 年 12 月提交了最终报告。该报告分析了自 CC1991 报告以来近 10 年的时间里，计算领域中来自技术和文化方面的变化。根据这些变化，任务组将 CC1991 报告划分的 11 个主领域扩展为 14 个主领域，提出了计算机科学知识体（Computer Science body of Knowledge）的新概念，并从领域、单元和主题三个不同的层次给出了知识体的内容，为整个学科核心课程的详细设计奠定了基础。

CC2001 最终报告肯定了《计算作为一门学科》报告为“计算作为一门学科”被更为广泛地承认所作的贡献，并称“这场为谋求合法性的战役取得了胜利”。但是 CC2001 也不得不承认与《计算作为一门学科》报告密切相关的 CC1991 教学计划的执行并没有达到预期的效果。为什么没有达到预期的效果？报告认为主要是缺乏更详细的课程指导。现在，新的 CC2001 报告不仅包含了更详细的课程设计内容，还给出了详细的课程描述。但是，许多重要的问题仍然没有解决，如由于整个学科大量内容的罗列，有可能使学生们只见树木，不见森林，因此，有必要寻找对整个学科进行科学的、系统的分析和总结的方法。可以预见，对新教学计划的争论仍将继续下去。

尽管 CC2001 最终报告还存在这样或那样的问题，但它的贡献是显著的，其主要的成果是完成了《计算作为一门学科》报告以来，必需解决而又未解决的关于整个学科核心课程详细设计这样一个重大问题。

1.1.5 计算教育面临的 3 个重大问题

计算教育 3 个重大问题的确定为未来计算教育的研究指明了方向，对计算学科的发展具有十分重要的意义。下面，我们简要地介绍和分析《计算作为一门学科》报告中提出的 3 个重大问题。

1. 3 个重大问题及其解决难点

（1）第一个重大问题——计算作为一门学科的存在性证明

解决难点：由于“证明一个学科的存在”是一个从来没有过的问题，因此，仅就“证明方式”来说，要得到学术界的广泛认可就是一件非常困难的事。

（2）第二个重大问题——整个学科核心课程的详细设计

解决难点：关于整个学科核心课程的详细设计，20 世纪 90 年代以来，各高校或多或少都存在着一些问题，例如随意性较强、没有一定的章法可循等等。在这种情况下，要让计算机的各种组织（如学会、专业委员会）以及企业界都认可其核心是计算机专业本科阶段必须学习的内容，就是一件非常困难的事。

（3）第三个重大问题——整个学科综述性导引（导论）课程的构建

解决难点：该问题的实质即是寻求一种统一的思想来认知计算机学科的本质，并对计算学科进行系统化和科学化的描述。ACM 攻关组针对这一问题提出了“严密性”和“挑战性”的要求。ACM 攻关组所指的“严密”，其英文单词是 Rigorous，这个词一般是指数学意义上的严密。也就是说，ACM 攻关组要求我们最好能用数学的方法来概括学科。现在的问题是，什么样的数学方法可以用来整理和总结学科？最好的答案自然就是公理化方法。而用公理化方法来概括学科，本身就是一件非常困难的工作。另外，在该课程中还应提出学科各主领域富有挑战性的科学问题，从而使问题难上加难。

2. 解决 3 个重大问题的意义

（1）第一个重大问题的解决，对学科的发展至关重要。如果在众多分支领域都取得重大成果并已得到广泛应用的“计算”，连作为一门学科的客观存在都不能被说明，那么它的发展将受到极大的限制。因此，这种为争取学科应有地位而进行的抗争，对学科的发展至关重要。

（2）第二个重大问题的解决，将为高校制定计算机教学计划奠定基础。确定一个公认的本科生必须掌握的核心内容，将避免教学计划设计中的随意性，从而为我们科学地制定教学计划奠定基础。

（3）第三个重大问题的解决，将使人们对整个计算学科的认知科学化、系统化和逻辑化。如果人们对计算学科的认知能建立在公理化的基础上，那么计算学科就可以被认为是严谨的科学，也可以被认为是一门成熟的学科，从而有助于计算学科的发展，并将为此而赢得人们特别的尊重。

1.1.6 计算机科学与技术方法论的提出

2001 年 7 月中旬，在上海召开的 CC2001 工作研讨会上，本书作者提交的《计算机科学与技术方法论》大会论文引起了 IEEE-CS 教育委员会副主席、CC2001 工作组负责人 Carl.Chang 教授（美籍华人）的注意，并在大会特邀报告中肯定了该论文所具有的批判性。2001 年 8 月 1 日，在网上公布的 CC2001 报告（草案），增加了作者充分肯定的原《计算作为一门学科》报告和 CC1991 报告中具有方法论性质的抽象、理论和设计三个过程，以及计算学科中反复出现的 12 个核心概念的内容，同时，还第一次增加了要求学生更多地了解方法论的内容。报告（草案）在新增的“关于学科导引性课程的基本宗旨”的论述中，再次将《计算作为一门学科》遗留下来的另一个尚未解决的重大问题——整个学科的综述性导引课程推向了前台。报告认为，符合所有有不同需要的学生的

综述性导引课程是不存在的，从一个侧面启发人们：不要再试图去建立那些引起类似于宗教战争那样激烈争论的、符合所有学生的综述性导引课程，而应当将注意力集中于学科中具有共同的、本质特征的内容上，从而解决这个问题。计算机科学与技术方法论承担的正是这项任务。因此，可以认为，《计算作为一门学科》提出的第三个重大问题已从整个学科综述性导引课程的构建问题演变为计算学科方法论课程的构建问题。

CC2001 教学计划是 IEEE-CS 和 ACM 一大批专家的成果，它和《计算作为一门学科》一样，代表的是一个集体的意见。由于它基本来源于计算教学实践，并经过几年的反复修改，因此，一般认为它基本遵循了计算教育的规律。然而，新的教学计划本身并不能说明其内容是否正确，它需要我们跳出教学计划本身这个问题，先解决学科的认知问题，再回头来看教学计划，这样做无疑将会更科学、更合理。

要解决学科的认知问题，必须有一套科学的方法。就哲学方法论而言，学科方法论是认知该学科的方法和工具，是学科认知领域的理论体系。

为此，我们有必要在前人工作的基础上，以一般科学技术方法论为指导，完整地、科学地建立起计算学科自己的方法论，为理解该学科提供方法，为科学地制定教学计划提供指导思想，为正确地理解和执行新的教学计划做准备。

1.2 计算机科学与技术方法论的建立

ACM 和 IEEE-CS 以及计算机界关于计算学科认知问题的大量研究成果，如《计算作为一门学科》报告、CC1991 报告、CC2001 报告，著名计算机科学家、图灵奖获得者戴克斯特拉（E. W.Dijkstra）教授“关于真正讲授计算科学的严酷性”（*On the Cruelty of Really Teaching Computing Science*）及其争论，以及学术界在有关计算教育哲学、计算机科学与技术方法论等方面的研究成果，促进了计算学科认知领域的发展，为计算机科学与技术方法论的建立奠定了基础。下面，我们以一般科学技术方法论为指导，科学地、完整地建立计算学科自己的方法论。

1.2.1 计算机科学与技术方法论的定义

计算机科学与技术方法论是对计算领域认识和实践过程中一般方法及其性质、特点、内在联系和变化发展进行系统研究的学问。计算机科学与技术方法论是认知计算学科的方法和工具，也是计算学科认知领域的理论体系。

在计算领域中，“认识”指的是抽象过程（感性认识）和理论过程（理性认识），“实践”指的是学科中的设计过程。抽象、理论和设计是具有方法论意义的 3 个过程，这 3 个过程是计算机科学与技术方法论中最重要的研究内容。

1.2.2 计算学科二维定义矩阵

1. 计算学科二维定义矩阵——知识框架

《计算作为一门学科》报告遵循了一般科学技术方法论的思想，它给出了计算学科二维定义矩阵（简称定义矩阵）的概念并细化了其内容。定义矩阵的一维是“3 个过程”（抽象、理论和设计），另一维是主领域。特别当主领域仅为计算学科时，定义矩阵便是《计算作为一门学科》报告中所指的“知识框架”。

“知识框架”反映了计算领域中人们的认识是从感性认识（抽象）到理性认识（理论），再由理性认识（理论）回到实践（设计）中来的科学思维方法。在这里，这个“知识框架”是对计算学科总的概括，它是稳定的；而“知识框架”的内容（值），即各主领域及其“3 个过程”的内容，则随计算技术的发展而变化。

2. 计算学科分支领域的划分

当前的计算学科已成为一门范围极为宽广的学科，将整个学科划分为若干分支领域有助于我们对计算学科的理解。分支领域的划分一般遵循以下 4 个原则：

（1）科目内容基础的协调一致；

（2）实质性的理论部分；

（3）有意义的抽象；

（4）重要的设计和实现。

并且，每一个分支必须被一个研究群体或几个相关的研究该分支的群体所确认。根据以上划分原则，《计算作为一门学科》报告将计算学科划分为 9 个主领域。CC1991 报告在此基础上，又增加了两个主领域：一个是程序设计语言引论；另一个是社会、道德和职业的问题。

自 CC1991 报告发表 10 年来，由于计算技术，特别是网络技术和图形学的迅猛发展，使它们不再适合作为操作系统和人—机通信两个主领域的研究主题。为此，CC2001 任务组将它们提取出来作为两个新的主领域：一个为网络计算，另一个为图形学和可视化计算。

另外，CC2001 报告还特意将离散数学从 CC1991 报告中的预备知识中抽取出来，列为学科的第一个主领域，命名为“离散结构”，以强调计算学科对它的依赖性。这样，CC2001 任务组最终将计算学科划分为 14 个主领域，如表 1.1 所示，这些领域基本覆盖了目前计算学科的内容。

表 1.1　　CC1991 与 CC2001 报告关于计算学科主领域的划分比较

CC1991（9＋2 个主领域）	CC2001（14 个主领域）
离散数学（预备知识）	离散结构（DS）
程序设计语言引论	程序设计基础（PF）
算法与数据结构	算法与复杂性（AL）
计算机体系结构	体系结构（AR）
操作系统	操作系统（OS）
	网络计算（NC）
程序设计语言	程序设计语言（PL）
人—机通信	人机交互（HC）
	图形学和可视化计算（GV）
人工智能与机器人学	智能系统（IS）
数据库与信息检索	信息管理（IM）
软件方法学与工程	软件工程（SE）
社会、道德和职业的问题	社会和职业的问题（SP）
数值与符号计算	科学计算（CN）

1.2.3 计算学科的本质问题归约为定义矩阵本质问题的说明

计算学科二维定义矩阵，如表 1.2 所示，是对计算学科的一个高度概括。因此，我们可以将把握计算学科的本质问题归约为把握计算学科二维定义矩阵的本质问题。要把握定义矩阵的本质，就是要分别把握定义矩阵的“横向”（抽象、理论和设计 3 个过程）以及“纵向”（各主领域）共有的、能反映各主领域内在联系的思想和方法的本质。

“横向”关系的内容，即抽象、理论和设计 3 个过程的内在联系与发展规律的内容，是计算机科学与技术方法论中最重要的内容。因为计算学科的基本原理不仅已被纳入抽象、理论和设计 3 个过程中，更重要的还在于，3 个过程之间的相互作用，推动了计算学科及其分支领域的发展。

“横向”关系中还蕴含着学科中的科学问题。由于人们对客观世界的认识过程就是一个不断提出问题和解决问题的过程，这种过程反映的正是抽象、理论和设计 3 个过程之间的相互作用，因此，科学问题与抽象、理论和设计 3 个过程在本质上是一致的，它与“3 个过程”共同构成了计算机科学与技术方法论中最重要的内容。

“纵向”关系的内容，即各分支领域中所具有的共同的能反映学科某一方面本质特征的内容，既有助于我们认知计算学科，又有助于我们更好地运用方法论中的思想从事计算领域的工作，它是方法论中仅次于科学问题与“3 个过程”的重要内容。

表 1.2　　计算学科二维定义矩阵

三个过程 / 学科主领域	抽象	理论	设计
1 离散结构（DS）			
2 程序设计基础（PF）			
3 算法与复杂性（AL）			
4 体系结构（AR）			
5 操作系统（OS）			
6 网络计算（NC）			
7 程序设计语言（PL）			
8 人机交互（HC）			
9 图形学和可视化计算（GV）			
10 智能系统（IS）			
11 信息管理（IM）			
12 软件工程（SE）			
13 社会和职业的问题（SP）			
14 科学计算（CN）			

“纵向”关系，即各分支领域之间，主要存在以下两个方面的联系：

（1）各分支领域中的某些研究内容是一致的，比如数据库中的并发控制、缓冲区管理的思想与操作系统中的并发控制和缓冲区管理的思想是一致的。

（2）计算学科中具有方法论性质的核心概念、数学方法、系统科学方法、形式化技术、社会和职业的问题贯穿于各分支领域之中，揭示了计算学科各分支领域的内在联系，使计算学科各分支领域结合成一个完整的体系，而不是一些互不相关的领域。

综上所述，计算学科二维定义矩阵中的科学问题，抽象、理论和设计 3 个过程（“横向”关系）与计算学科中的核心概念、数学方法、系统科学方法、形式化技术、社会和

职业的问题（“纵向”关系）构成了计算学科方法论的主要内容。自此，计算学科认知领域的理论体系——计算机科学与技术方法论已经建立。

1.3 计算机科学与技术方法论作为一个理论体系的阐述

由于计算机科学与技术方法论是从一般科学技术方法论中派生出来的，因此，我们先借鉴数学的公理化思想来阐述作为理论体系的一般科学与技术方法论的无矛盾性、独立性和完备性。

1.3.1 作为理论体系的科学技术方法论

1. 公理化方法

（1）公理化方法的基本思想

从尽可能少的概念（原始概念）和尽可能少的命题（原始命题，又称公理）出发，演绎出本领域（或学科）其他所有的概念和命题，从而构成这一领域（或学科）的全貌。

（2）公理系统需要满足的 3 个条件

① 无矛盾性（协调性或一致性）：在一个公理系统中不能推出自相矛盾的结论。

② 独立性：在一个公理系统中，任何一个公理都不能由其他公理推出。

③ 完备性：从一个公理系统出发能推出该领域（或学科）中的所有命题。

公理化方法是构建理论体系的一种有效方法，而判定一个理论体系（公理系统）是不是科学的，它的基础在逻辑上是否已经奠定，判定的标准就是该理论体系是否满足无矛盾性、独立性和完备性的条件。

如果一个领域（或学科）能用公理化的方法来构建，那么，该领域（或学科）就被认为是严谨的科学，也被认为是十分成熟的领域（或学科）。

需要指出的是：本书关于“认知理论”的阐述只是借鉴了数学的公理化思想，与纯数学上的证明不同，本书的目的不在于纯数学上的意义，而在于使用这种方法所获得的对计算学科进行总结的系统性、逻辑性和科学性。

2. 关于“认知”领域理论体系的建立

众所周知，在人类社会实践中，认识（Cognition）和实践（Practice）是两个最基本的概念，于是我们便将这两个概念作为人类社会实践中有关认知领域的原始概念。而认识又以实践为基础，因此，我们以“认识以实践为基础”这个命题作为原始命题（公理）。由于人类社会实践中所有的概念均蕴含于认识和实践两个原始概念之中，因此，由人类社会实践中任何概念组合构成的命题必然蕴含于“认识以实践为基础”这个原始命题之中，于是，我们便将“认识以实践为基础”作为人类社会实践中有关认知领域的惟一原始命题。自此，一个以认识和实践为原始概念，以“认识以实践为基础”为原始命题的关于“认知”领域的理论体系就构建起来了。

3. 关于“认识以实践为基础”的简要论述

（1）“认识以实践为基础”是对作为认识的一般过程和总体过程而言的，实践是认识的基础，人们的认识归根结底产生于人类的社会实践之中。

（2）“认识以实践为基础”没有也不可能排斥认识的局部和个别过程的复杂性和特殊性。

（3）人们现在的实践活动总是建立在人们以往实践活动所取得的认识基础上。

4. 一般科学技术方法论

为便于使用，并就技术学科而言，人们又增加了蕴含于“认识以实践为基础”命题中的，对认识和实践过程一般方法及其性质、特点、内在联系和变化发展进行研究的内容，并将这些内容构成的理论体系称为一般科学技术方法论。

5. 科学技术方法论的无矛盾性、独立性和完备性

在以认识和实践两个概念为原始概念，以“认识以实践为基础”作为惟一原始命题的理论体系——一般科学技术方法论中，由于原始命题仅有一个，即“认识以实践为基础”，因此，以上介绍的理论体系——一般科学技术方法论符合理论系统的无矛盾性和独立性的要求。

另外，由于人们在社会实践过程中所形成的所有概念都可以用“认识”和“实践”两个概念来概括，从而使任何概念组合构成的任何命题最终都蕴含于“认识以实践为基础”之中，因此，以上介绍的一般科学技术方法论又符合一个理论系统完备性的要求。

6. 一般科学技术方法论的形式化定义

对于计算专业和数学专业的读者而言，他们希望看到的是用形式化方法定义的理论体系，本书借助数学的集合论，给出一般科学技术方法论的形式化定义，以建立它坚实的理论基础。

一般科学技术方法论（用 M 表示）是一个四元组，即

$M=<Q，C，P，F>$

其中：

（1）Q 是一个包含子集 C 和子集 P 的集合；

（2）C 是所有属于“认识”概念的集合；

（3）P 是所有属于“实践”概念的集合；

（4）F 表示由 Q 到它自身的一个函数。

函数 F 有以下性质：

① 函数 F 具有自反性，即对于任何概念 $q\in Q$，有 $F(q)=q$。

② 对于任何两个 Q 中不在同一子集（C，P）的概念 e_i 和 e_j，判定 $F(e_i)=e_j$ 是否成立的条件，就是看它是否符合命题“认识以实践为基础”的要求，符合则判定等式成立，否则就不成立。

③ 对于任何一个概念 $e_i\in P$，必有一个或多个概念 $e_j\in C$，满足 $F(e_i)=e_j$，反之，则不一定成立。该性质的含义在于：人们当前从事的一切有目的的实践活动，都是以一定的认识为指导的，而有些认识（在此指理性认识），到目前为止，可能尚未应用于实践之中。

1.3.2 作为理论体系的计算机科学与技术方法论

1. 计算机科学与技术方法论的无矛盾性、独立性和完备性阐述

计算机科学与技术方法论是一个具体的科学技术方法论，它具有一般科学技术方法论的共性，又具有自己的特性，因此，必须将其具体化，以形成计算学科的一个理论体系。

在计算领域中，“认识”指的是学科中的抽象（Abstraction）过程和理论（Theory）过程，“实践”指的是学科中的设计（Design）过程。因此，在计算机科学与技术方法论中的原始概念由学科中更具体的“抽象”、“理论”和“设计”构成。

在计算领域中，将“认识以实践为基础”命题进一步具体化为“认识以实践为基础，并从感性认识（抽象）到理性认识（理论），再由理性认识（理论）回到实践（设计）”。

在以抽象、理论和设计作为原始概念，以“认识以实践为基础，并从感性认识到理性认识，再由理性认识回到实践”作为惟一原始命题的理论体系——计算机科学与技术方法论中，由于原始命题仅有一个，因此，计算机科学与技术方法论基本满足了理论系统的无矛盾性和独立性的要求。

另外，由于计算学科中所有的概念都蕴含于 3 个原始概念之中，从而使计算学科中任何概念组合构成的命题最终蕴含于惟一的原始命题之中，因此，计算机科学与技术方法论又符合一个理论系统完备性的要求。

2. 计算机科学与技术方法论的形式化定义

计算机科学与技术方法论（用 C 表示）是一个具体的科学技术方法论，它将一般科学技术方法论中的 C 和 P 两个元组改为更具体的 A、T、D，因此，它是一个五元组，即

$C=<Q，A，T，D，F>$

其中：

（1）Q 是一个包含子集 A、T、D 的集合；

（2）A 是计算学科中所有属于“抽象”概念的集合；

（3）T 是计算学科中所有属于“理论”概念的集合；

（4）D 是计算学科中所有属于“设计”概念的集合；

（5）F 表示由 Q 到它自身的一个函数。

函数 F 有以下性质：

① 函数 F 具有自反性，即对于任何概念 $q\in Q$，有 $F(q)=q$。

② 对于任何两个 Q 中不在同一子集（A，T，D）的概念 e_i 和 e_j，判定 $F(e_i)=e_j$ 是否成立的条件，就是看它是否符合命题“认识以实践为基础，并从感性认识（抽象）到理性认识（理论），再由理性认识（理论）回到实践（设计）”的要求，符合则判定等式成立，否则就不成立。

③ 对于任何一个概念 $d_i\in D$，必有一个或多个概念 $e_i\in A$ 或 $e_i\in T$，满足 $F(d_i)=e_i$，反之，则不一定成立。该性质的含义在于：人们当前从事的一切有目的的工程设计，都是以一定的认识为指导的，而有些认识（在此指理性认识），到目前为止，可能尚未应用于工程设计之中。

④ 对于任何一个概念 $t_i\in T$，必有一个或多个概念 $e_i\in A$，满足 $F(t_i)=e_i$，反之，则不一定成立。该性质的含义在于：任何理性认识（理论）都是建立在感性认识（抽象）的基础上，而感性认识不一定都能上升到理性认识。

3．计算机科学与技术方法论的主要内容

计算机科学与技术方法论中惟一的原始命题，其实也就是计算学科二维定义矩阵中的“横向”关系，即抽象、理论和设计 3 个过程内在联系所要研究的内容。

由于“科学研究从问题开始”与“认识以实践为基础”是从不同角度得出的不同命题，而其本质是一致的，因此，我们将科学问题从抽象、理论和设计 3 个过程中提取出来，构成与 3 个过程具有相同地位的重要内容。

为了更易于把握计算学科的本质，又可以将蕴含于计算学科中的抽象、理论和设计 3 个过程中各分支领域所具有共同特征的核心概念、数学方法、系统科学方法、形式化技术以及社会和职业的问题提取出来，与计算学科中的科学问题和抽象、理论和设计 3 个过程一起共同构成计算机科学与技术方法论的主要内容。

显然，计算机科学与技术方法论中的主要内容反映的正是计算学科二维定义矩阵中“横向”和“纵向”关系所研究的内容，因此，可以说，前面我们已经建立起了计算机科学与技术方法论这个学科认知领域的理论体系。

4. 计算机科学与技术方法论内容的进一步补充

值得注意的是，由于上述计算机科学与技术方法论是采用公理化方法来构建的，而公理化方法主要用于回顾性的总结，对计算学科中有争论的问题以及未来探索性的展望作用有限。因此，我们在以上构建的基础上，又增加了两个对计算学科认知有重要影响的内容，这些内容以任务的形式加以确认：一个是对计算教育史上和当前计算教育实际中有争议的问题作出科学的分析和评论；另一个是根据计算技术发展的趋势和要求，对计算教育中提出的新课题作出回答，对未来的计算教育作

出科学的预测。

原有的计算机科学与技术方法论的内容再加上现在新增的内容，又形成了一门新的基础理论—计算教育哲学。由于计算教育哲学是一门具有方法论性质的学问，而又有助于学科的进一步认知，因此，作者暂且将它合并在本书给出的计算机科学与技术方法论之中，待条件成熟时，再将它独立出来。

1.4 计算机科学与技术方法论研究的意义

计算机科学与技术方法论遵循一般科学技术方法论的普遍原理，但是，它又不同于一般科学技术方法论。一般科学技术方法论在学科认识中具有一般性的指导意义；而计算机科学与技术方法论直接面对和服务于计算学科的认识过程，使人们对计算学科的认识逻辑化、程序化、理性化和具体化，它是我们认知计算学科的工具。就某种意义而言，计算机科学与技术方法论的建立正是计算学科成熟的标志之一。

计算机科学与技术方法论的研究不仅具有理论意义，也具有现实意义，它能促进计算学科的发展，有助于计算学科的建设与人才培养。下面，从 5 个方面来概括它的意义。

1. 有助于我们正确理解计算学科中所蕴含的科学思维方法

抽象、理论和设计是计算机科学与技术方法论中最基本的 3 个概念。3 个学科形态的划分，反映了人们的认识从感性认识（抽象）到理性认识（理论），再回到实践（设

计）中来的科学思维方法，并有助于我们从方法论的高度来认知计算学科。

2. 有助于总结和提升计算学科所积累的各种方法与经验

计算学科自 20 世纪 40 年代诞生以来，科学家们在短短 50 多年的历史中取得了大量的里程碑式的科学业绩，使该学科得到了迅猛的发展。同时，计算机科学家们也创造了大量的获取这些成果的工具——研究方法，这是一份更为宝贵的财富。然而就大多数研究者而言，研究成果是他们自觉追求的目标，而方法则是为取得成果而采取的手段。一旦研究工作获得成功，大多数研究者对于自己的科研成果的价值甚为清楚，而对获得成果所使用的方法却重视不够，一般很少自觉地将自己的研究方法加以系统地总结和提升，使它们成为具有普遍意义的思想工具。

计算机科学与技术方法论的任务就是要系统地总结和提升这些方法，使它们成为具有普遍意义的思想工具，从而被计算领域的工作者所掌握，以便使我们更好地从事该领域的工作。

3. 有助于促进计算学科的发展

从计算学科的发展来看，计算学科的每一个重要进展几乎都与研究方法的改进密切地联系着，计算学科每个领域的发展总是与那一时期的研究方法的发展水平相适应。

例如，在 20 世纪 50 年代，程序设计受当时技术的限制，如果说程序设计采用了一定的方法的话，那就是技巧；20 世纪 60 年代末期由于软件危机以及硬件的发展，出现了规范化的程序设计；20 世纪 70 年代末期，形成了以结构化方法为主的系统开发方法；20 世纪 80 年代人们发现，由于行业间语言的障碍，要弄清用户的需求有困难，于是提出了原型化的思想方法；20 世纪 90 年代开始，随着人们认识的进一步深化，为直面要解决的现实世界，提出了目前占主导地位的面向对象的思想方法。

新的思想方法是不会凭空产生的，它是对已有方法进行改进的结果，从以上研究方法的发展来看，它符合实践、认识、再实践、再认识的一般规律。也就是说，它需要我们对现有方法进行深入研究，在肯定它们具有认识功能的同时，又注意到它们的局限性，要根据研究对象的特点与需要，对研究方法进行改进，实现方法上的突破，从而促进学科的发展。

4. 有助于确立正确的思想原则，把握正确的研究方向

恩格斯在其著作《自然辩证法》中说过：不管自然科学家采取什么样的态度，他们还是得受哲学的支配。问题只在于他们是愿意受某种坏的时髦的哲学来支配他们，还是愿意受一种建立在通晓历史及其成就的基础上的理论思维的支配。一个研究者如果不具备正确的理论思维形式，不以正确的思想方法作指导，在具体的研究实践中可能会常常碰壁，甚至对理应到手的研究成果也可能会视而不见，置若罔闻。

计算机科学与技术方法论是在一般科学技术方法论的指导下建立的，它吸收了其他学科已有的方法论成果，遵守共同的方法论原则。同时，它又是解决计算学科特殊矛盾所需的、有着计算学科特点的、相对独立的方法论，它直接面对和服务于计算学科的认识过程，使人们对计算学科的认识逻辑化、程序化、理性化和具体化，它有助于我们在计算学科的研究中确立正确的思想原则，把握正确的研究方向。

5. 有助于计算学科的建设和人才培养

如何进行计算学科的建设，是一个长期以来争论的问题。计算机科学与技术方法论从计算学科的科学问题，学科的抽象、理论和设计 3 个过程，学科的核心概念，数学方法、系统科学方法、形式化技术，以及社会和职业的问题等方面出发，深刻地揭示了计算学科的本质，有助于我们对计算学科的认识，从而有助于计算学科的建设。

计算专业的学生如能在大学的学习中系统地接受学科方法论的指导，以了解、懂得研究工作的一般程序、操作技术与正确的思维方法，无疑有助于自己的成长。

思 考 题

1．简述计算机科学与技术方法论产生的历史背景。

2．《计算作为一门学科》报告取得的主要成果是什么？存在的局限性是什么？

3．CC1991 报告的主要成果是什么？

4．CC2001 报告的主要成果是什么？

5．计算教育面临的 3 个重大问题及其难点、意义。

*6．为什么说《计算作为一门学科》报告提出的第三个重大问题已从“整个学科综述性导引课程的构建问题”演变为“计算机科学与技术方法论的构建问题”？

7．为什么计算学科的本质问题可以归约为计算学科二维定义矩阵的本质问题？

8．如何正确把握计算学科二维定义矩阵的本质？

9．什么是计算机科学与技术方法论？请给出其形式化定义。

10．简述公理化方法的基本思想。

11．公理系统需要满足哪 3 个条件？

12．本书关于“认知理论”的阐述借鉴了数学的公理化思想，其的目的是什么？

*13．用公理化的思想方法简要论述计算机科学与技术方法论作为理论体系是怎样建立起来的。

14．计算机科学与技术方法论的主要内容是什么？

15．计算机科学与技术方法论的研究有何理论意义和现实意义？

第 2 章　计算学科中的科学问题

2.1　概　述

2.1.1　科学问题的定义

科学问题是指一定时代的科学认识主体，在已完成的科学知识和科学实践的基础上，提出的需要解决且有可能解决的问题。它包含一定的求解目标和应答域，但尚无确定的答案。

科学问题是认识的一种形式，它既包含先前的实践和认知的基础，又预示着进一步的实践和认识的方向，它是“认识以实践为基础”这一命题的具体化形式，它产生于人们的社会生产实践与科学实践过程中。从科学史来看，人类科技进步的历史就是一个不断提出科学问题又不断解决科学问题的历史。因此，科学问题在方法论中占有极其重要的地位。能否在所从事的工作中提出（或从众多的问题中抽取）关键和重要的科学问题，对我们每个人来说都是一个挑战。而方法论的学习有助于我们自觉地、主动地去迎接这种挑战。

2.1.2　科学问题的主要特征和方法论作用

1. 科学问题的主要特征

（1）时代性：从历史的观点来看，任何一个科学问题都具有它的时代特性。每一个时代都有它自己的科学问题，而这些问题的解决对科学的发展具有深远的意义。

（2）混沌性：科学问题显示了人们对已有知识的不满，并渴望对新知识的追求，但这种追求开始的时候是模糊不清的。

（3）可解决性：科学问题是由于决心解决而又有可能解决才提出的，提出科学问题后便要力图解决它。

（4）可变异性：相对科学问题的可解决性而言，如果一个问题未能解决，似乎就不是科学问题，其实不然，如果它还能引出另外具有可解决性的科学问题，则原问题仍属于科学问题。

（5）可待解性：由于尚不具备解决问题的全部条件，因此许多科学问题在当前一段时间里还很难解决或无法解决，但绝非永远不可解决。

2. 科学问题的方法论作用

（1）科学问题的裂变式作用

对于一门学科而言，原先科学问题的提出与解决，会诱发出新的科学问题，而新的科学问题的解决又会诱发更新的科学问题，这种父子型、子孙型科学问题的连续出现和相继解决，可以导致该门学科的重大理论突破。例如对“数学基础问题”的研究，导致了“形式系统相容性问题”的研究，最后出现“能行性问题”的研究，并最终于 20 世纪 30 年代由图灵、哥德尔、丘奇和波斯特等人共同奠定了计算学科的理论基础，实现

了人类对计算认识问题的重大突破。

（2）科学问题的聚变式作用

对不同科学问题的研究最终导致同一科学问题的发现，这种殊途同归的结果，就是科学问题聚变式作用的结果。

（3）科学问题的激励作用

新的重大科学问题的确定总是在以往时代科学问题结束之际到来的，它犹如一面旗帜，象征着人类科学认识进入到一个崭新的阶段，它召唤和激励着人们为解决这些富有挑战性的问题而勇往直前。

在科学哲学中，从不同角度出发，科学问题有不同的分类方法，本书不对这些分类方法进行讨论，仅对反映计算学科本质的根本问题、学科各领域的基本问题、在学科中起重要作用的典型问题，以及人工智能中的若干哲学问题进行分析。

2.2 计算的本质、计算学科的定义及其根本问题

计算学科根本问题的认识过程与人们对计算过程的认识是紧密联系在一起的，因此，要分析计算学科的根本问题，首先要分析人们对计算本质的认识过程。

2.2.1 计算本质的认识历史

在很早以前，人们就碰到了必须计算的问题。已经考证的，远在旧石器时代，刻在骨制和石头上的花纹就是对某种计算的记录。然而，在 20 世纪 30 年代以前，人们并没有真正认识计算的本质。尽管如此，在人类漫长的岁月中，人们一直没有停止过对计算本质的探索。很早以前，我国的学者就认为，对于一个数学问题，只有当确定了其可用算盘解算它的规则时，这个问题才算可解。这就是古代中国的“算法化”思想，它蕴含着中国古代学者对计算的根本问题，即“能行性”问题的理解，这种理解对现代计算学科的研究仍具有重要的意义。中国科学院院士吴文俊教授正是在这一基础上围绕几何定理的机器证明展开研究，并开拓了一个在国际上被称为“吴方法”的新领域——数学的机械化领域。吴文俊教授为此于 2000 年获得首届国家最高科学技术奖。

算盘作为主要的计算工具流行了相当长的一段时间，直到中世纪，哲学家们提出了这样一个大胆的问题：能否用机械来实现人脑活动的个别功能？最初的试验目的并不是制造计算机，而是试图从某个前提出发机械地得出正确的结论，即思维机器的制造。早在 1275 年，西班牙神学家雷蒙德·露利（R.Lullus）就发明了一种思维机器（“旋转玩具”），从而开创了计算机器制造的先河。

“旋转玩具”引起了许多著名学者的研究兴趣，并最终导致了能进行简单数学运算的计算机器的产生。1641 年，法国人帕斯卡（B.Pascal）利用齿轮技术制成了第一台加法机；1673 年德国人莱布尼茨（G.W.V.Leibniz）在帕斯卡的基础上又制造了能进行简单加、减、乘、除的计算机器；19 世纪 30 年代，英国人巴贝奇（C.Babbage）设计了用于计算对数、三角函数以及其他算术函数的“分析机”；20 世纪 20 年代，美国人布什（V.Bush）研制了能解一般微分方程组的电子模拟计算机等。计算的这一历史包含了人们对计算过程的本质和它的根本问题进行的探索，同时，还为现代计算机的研制积累了经验。

其实，对计算本质的真正认识取决于形式化研究的进程，而“旋转玩具”就是一种

形式化的产物，不仅如此，它还标志着形式化思想革命的开始。

2.2.2 康托尔的集合论和罗素悖论

形式化方法和理论的研究起源于对数学的基础研究。数学的基础研究是指对数学的对象、性质及其发生、发展的一般规律进行的科学研究。

德国数学家康托尔（G.Cantor，1845～1918）从 1874 年开始，对数学基础作了新的探讨，发表了一系列集合论方面的著作，从而创立了集合论。康托尔创立的集合论对数学概念作了重要的扩充，对数学基础的研究产生了重大影响，并逐步发展成为数学的重要基础。

然而不久，数学家们却在集合论中发现了逻辑矛盾，其中最为著名的是 1901 年罗素（B. Russell）在集合论概括原则的基础上发现的“罗素悖论”，从而导致了数学发展史上的第三次危机。

罗素悖论可以这样形式化地定义：$S=\{x \mid x \notin S\}$。为了使人们更好地理解集合论悖论，罗素将“罗素悖论”改写成“理发师悖论”。其大意是，一个村庄的理发师宣布了这样一条规定：“给且只给村里那些不自己刮胡子的人刮胡子”。现在要问：理发师给不给自己刮胡子呢？如果理发师给自己刮胡子，他就属于那类“自己刮胡子的人”，按规定，该理发师就不能给自己刮胡子；如果理发师不给自己刮胡子，那么，他就属于那类“不自己刮胡子的人”，按规定，他就应该给自己刮胡子。由此可以推出两个相互矛盾的等价命题：理发师自己给自己刮胡子⇔理发师自己不给自己刮胡子。

2.2.3 希尔伯特纲领

为了消除悖论，奠定更加牢固的数学基础，20 世纪初，逐步形成了关于数学基础研究的逻辑主义、直觉主义和形式主义三大流派。其中，形式主义流派的代表人物是大数学家希尔伯特（D.Hilbert）。他在数学基础的研究中提出了一个设想，其大意是：将每一门数学的分支形式化，构成形式系统或形式理论，并在以此为对象的元理论即元数学中，证明每一个形式系统的相容性，从而导出全部数学的相容性。希尔伯特的这一设想，就是所谓的“希尔伯特纲领”。

“希尔伯特纲领”的目标，其实质就是要寻找通用的形式逻辑系统，该系统应当是完备的，即在该系统中，可以机械地判定任何给定命题的真伪。

“希尔伯特纲领”的研究基础是逻辑和代数，主要源于 19 世纪英国数学家乔治·布尔（G.Boole）所创立的逻辑代数体系（即布尔代数）。1854 年，布尔在他的著作中成功地将“真”、“假”两种逻辑值和“与”、“或”、“非”3 种逻辑运算归结为一种代数。这样，形式逻辑系统中的任何命题都可用数学符号表示出来，并能按照一定的规则推导出结论。尽管布尔没有将“布尔代数”与计算机联系起来，但他的工作却为现代计算机的诞生作了重要的理论准备。希尔伯特的工作建立在布尔工作的基础上，并使其进一步具体化。

希尔伯特对实现自己的纲领充满信心。然而，1931 年，奥地利 25 岁的数理逻辑学家哥德尔（K.Gödel）提出的关于形式系统的“不完备性定理”中指出，这种形式系统是不存在的，从而宣告了著名的“希尔伯特纲领”的失败。希尔伯特纲领的失败同时也暴露了形式系统的局限性，它表明形式系统不能穷尽全部数学命题，任何形式系统中都存在着该系统所不能判定其真伪的命题。

“希尔伯特纲领”虽然失败了，但它仍然不失为人类抽象思维的一个伟大成果，它的历史意义是多方面的。

首先，“希尔伯特纲领”是在保存全部古典数学的前提下去排除集合论悖论的，它给数学基础问题的研究带来了全新的转机。其次，希尔伯特纲领的提出使元数学得到了确立和发展。最后，对计算学科而言，最具意义的是，希尔伯特纲领的失败启发人们应避免花费大量的精力去证明那些不能判定的问题，而应把精力集中于解决具有能行性的问题。

2.2.4 图灵对计算本质的揭示

在哥德尔研究成果的影响下，20 世纪 30 年代后期，图灵（A.M.Turing）从计算一个数的一般过程入手对计算的本质进行了研究，从而实现了对计算本质的真正认识。

根据图灵的研究，直观地说，所谓计算就是计算者（人或机器）对一条两端可无限延长的纸带上的一串 0 和 1 执行指令，一步一步地改变纸带上的 0 或 1，经过有限步骤，最后得到一个满足预先规定的符号串的变换过程。图灵用形式化方法成功地表述了计算这一过程的本质。图灵的研究成果是哥德尔研究成果的进一步深化，该成果不仅再次表明了某些数学问题是不能用任何机械过程来解决的思想，而且还深刻地揭示了计算所具有的“能行过程”的本质特征。

图灵的描述是关于数值计算的，不过，我们知道英文字母表的字母以及汉字均可以用数来表示，因此，图灵机同样可以处理非数值计算。不仅如此，更为重要的是，由数值和非数值（英文字母、汉字等）组成的字符串，既可以解释成数据，又可以解释成程序，从而计算的每一过程都可以用字符串的形式进行编码，并存放在存储器中，以后使用时译码，并由处理器执行，机器码（结果）可以从高级符号形式（即程序设计语言）机械地推导出来。

图灵的研究成果是：可计算性=图灵可计算性。在关于可计算性问题的讨论时，不可避免地要提到一个与计算具有同等地位和意义的基本概念，那就是算法。算法也称为能行方法或能行过程，是对解题（计算）过程的精确描述，它由一组定义明确且能机械执行的规则（语句、指令等）组成。根据图灵的论点，可以得到这样的结论：任一过程是能行的（能够具体表现在一个算法中），当且仅当它能够被一台图灵机实现。图灵机与当时哥德尔、丘奇（A.Church）、波斯特（E.L.Post）等人提出的用于解决可计算问题的递归函数、λ 演算和 POST 规范系统等计算模型在计算能力上是等价的。在这一事实的基础上，形成了现在著名的丘奇—图灵论题。

图灵机等计算模型均是用来解决“能行计算”问题的，理论上的能行性隐含着计算模型的正确性，而实际实现中的能行性还包含时间与空间的有效性。

2.2.5 现代计算机的产生以及计算学科的定义

伴随着电子学理论和技术的发展，在图灵机这个思想模型提出不到 10 年的时间里，世界上第一台电子计算机诞生了。

其实，图灵机反映的是一种具有能行性的用数学方法精确定义的计算模型，而现代计算机正是这种模型的具体实现。正如前面提到的那样，计算学科各分支领域均可以用模型与实现来描述。而模型反映的是计算学科的抽象和理论两个过程，实现反映的是计算学科的设计过程，模型与实现已蕴含于计算学科的抽象、理论和设计 3 个过程之中。

计算学科各分支领域中的抽象和理论两个过程关心的是解决具有能行性和有效性的模型问题，设计过程关心的是模型的具体实现问题。正因为如此，计算学科中的 3 个形态是不可分割、密切相关的。

计算运用了科学和工程两者的方法学，理论工作已大大地促进了这门艺术的发展。同时，计算并没有把新的科学知识的发现与利用这些知识解决实际的问题分割开来。理论和实践的紧密联系给该学科带来了力量和生机。

正是由于计算学科理论与实践的紧密联系，并伴随着计算技术的飞速发展，计算学科现已成为一个极为宽广的学科。

在进行大量分析的基础上，《计算作为一门学科》报告给计算学科作了以下定义：

计算学科是对描述和变换信息的算法过程，包括对其理论、分析、设计、效率、实现和应用等进行的系统研究。它来源于对算法理论、数理逻辑、计算模型、自动计算机器的研究，并与存储式电子计算机的发明一起形成于 20 世纪 40 年代初期。

计算学科包括对计算过程的分析以及计算机的设计和使用。该学科的广泛性在下面一段来自美国计算科学鉴定委员会（Computing Sciences Accreditation Board）发布的报告摘录中得到强调：

计算学科的研究包括从算法与可计算性的研究到根据可计算硬件和软件的实际实现问题的研究。这样，计算学科不但包括从总体上对算法和信息处理过程进行研究的内容，也包括满足给定规格要求的有效而可靠的软硬件设计——它包括所有科目的理论研究、实验方法和工程设计。

2.2.6 计算学科的根本问题

同样，也是在大量分析的基础上，《计算作为一门学科》报告对学科中的根本问题作了以下概括。

计算学科的根本问题是：什么能被（有效地）自动进行。

计算学科的根本问题讨论的是“能行性”的有关内容。而凡是与“能行性”有关的讨论，都是处理离散对象的。因为非离散对象，即所谓的连续对象，是很难进行能行处理的。因此，“能行性”这个计算学科的根本问题决定了计算机本身的结构和它处理的对象都是离散型的，甚至许多连续型的问题也必须在转化为离散型问题以后才能被计算机处理。例如，计算定积分就是把它变成离散量，再用分段求和的方法来处理的。

正是源于计算学科的根本问题，以离散型变量为研究对象的离散数学对计算技术的发展起着十分重要的作用。同时，又因为计算技术的迅猛发展，使得离散数学越来越受到重视。为此，CC2001 报告特意将它从 CC1991 报告的预备知识中抽取出来，列为计算学科的第一个主领域，命名为“离散结构”，以强调计算学科对它的依赖性。

尽管计算学科已成为一个极为宽广的学科，但其根本问题仍然是：什么能被（有效地）自动进行。甚至还可以更为直率地说，计算学科所有分支领域的根本任务就是进行计算，其实质就是字符串的变换。

在弄清计算学科的根本问题的实质后，还可以对计算学科根本问题作进一步的阐述：

在论述人的计算能力方面，著名的数理逻辑学家美籍华人王浩教授认为，如果我们同意只关心作为机械过程的计算，那么许多论述将更加清楚，并能避免像精神与人体的

关系这样的心理学和哲学的问题。因此，就机械过程的计算而言，王浩认为：人要做机器永远不能做的某些计算是不容易的。

2.2.7 从计算的角度认知思维、视觉和生命过程

以 1975 年图灵奖共同获得者西蒙（H.A.Simon）和纽厄尔（A.Newell）为代表的符号主义者认为：认知是一种符号处理过程。同时，他们还提出了人类思维过程也可用某种符号来描述的思想，即思维就是计算（认知就是计算）的思想。除了思维、认知之外，有关视觉认知理论的学者也把视觉看作是一种计算。其中，以 Marr（D.Marr）的计算视觉理论最为著名。计算视觉理论目前已得到广泛的应用。

1994 年 11 月美国科学家阿德勒曼（L.M.Adleman）在《科学》（*Science*）杂志上发表了《解决组合问题的分子计算》（*Molecular Computation of Solutions to Combinatorial Problems*）论文，并引起了广泛的关注，该论文论证了 DNA（脱氧核糖核酸）计算技术的可行性，并用 DNA 计算机解决了一个简单的有向哈密尔顿路径问题。2002 年，阿德勒曼教授的研究工作又有了一些新的进展，应用 DNA 计算机可以解决具有 100 万种可能结果的有向哈密尔顿路径问题，阿德勒曼的工作从一个侧面探讨了生命过程就是一种计算的思想。在生命科学，特别是基因技术取得重要突破的今天，如何充分运用生命科学所取得的成果来研究计算学科中的问题，或者反过来用计算这个具有科学方法论意义的思想工具来研究生命科学中的问题，都是值得学术界重视的问题。

2.3 计算学科各主领域的基本问题

计算学科各主领域的基本问题来源于各主领域所研究的内容，因此，在给出各主领域的基本问题前，先给出各主领域所包括的主要内容。

1. 离散结构

主要内容：包括集合论、数理逻辑、近世代数、图论以及组合数学等。该领域与计算学科各主领域有着紧密的联系，CC2001 为了强调它的重要性，特意将它列为计算学科的第一个主领域。该主领域以抽象和理论两个学科形态出现在计算学科中，它为计算学科各分支领域解决其基本问题提供了强有力的数学工具。

2. 程序设计基础

（1）主要内容：包括程序设计结构、算法、问题求解和数据结构等。

（2）基本问题主要包括：

① 对给定的问题，如何进行有效的描述并给出算法？

② 如何正确选择数据结构？

③ 如何进行设计、编码、测试和调试程序？

3. 算法与复杂性

（1）主要内容：包括算法的复杂度分析、典型的算法策略、分布式算法、并行算法、可计算理论、P 类和 NP 类问题、自动机理论、密码算法以及几何算法等。

（2）基本问题主要包括：

① 对于给定的问题类，最好的算法是什么？要求的存储空间和计算时间有多少？空间和时间如何折衷？

② 访问数据的最好方法是什么？

③ 算法最好和最坏的情况是什么？

④ 算法的平均性能如何？

⑤ 算法的通用性如何？

4. 体系结构

（1）主要内容：包括数字逻辑、数据的机器表示、汇编级机器组织、存储技术、接口和通信、多道处理和预备体系结构、性能优化、网络和分布式系统的体系结构等。

（2）基本问题主要包括：

① 实现处理器、内存和机内通信的方法是什么？

② 如何设计和控制大型计算系统，而且使其令人相信，尽管存在错误和失败，但它仍然是按照我们的意图工作的？

③ 哪种类型的体系结构能够有效地包含许多在一个计算中能够并行工作的处理元素？

④ 如何度量性能？

5. 操作系统

（1）主要内容：包括操作系统的逻辑结构、并发处理、资源分配与调度、存储管理、设备管理、文件系统等。

（2）基本问题主要包括：

① 在计算机系统操作的每一个级别上，可见的对象和允许进行的操作各是什么？

② 对于每一类资源，能够对其进行有效利用的最小操作集是什么？

③ 如何组织接口才能使得用户只需与抽象的资源而非硬件的物理细节打交道？

④ 作业调度、内存管理、通信、软件资源访问、并发任务间的通信以及可靠性与安全的控制策略是什么？

⑤ 通过少数构造规则的重复使用进行系统功能扩展的原则是什么？

6. 网络计算

（1）主要内容：包括计算机网络的体系结构、网络安全、网络管理、无线和移动计算以及多媒体数据技术等。

（2）基本问题主要包括：

① 网络中的数据如何进行交换？

② 网络协议如何验证？

③ 如何保证网络的安全？

④ 分布式计算的性能如何评价？

⑤ 分布式计算如何组织才能够使通过通信网连接在一起的自主计算机参加到一项计算中，而网络协议、主机地址、带宽和资源则具有透明性？

7. 程序设计语言

（1）主要内容：包括程序设计模式、虚拟机、类型系统、执行控制模型、语言翻译系统、程序设计语言的语义学、基于语言的并行构件等。

（2）基本问题主要包括：

① 语言（数据类型、操作、控制结构、引进新类型和操作的机制）表示的虚拟机的可能组织结构是什么？

② 语言如何定义机器？机器如何定义语言？

③ 什么样的表示法（语义）可以有效地用于描述计算机应该做什么？

8. 人－机交互

（1）主要内容：包括以人为中心的软件开发和评价、图形用户接口设计、多媒体系统的人机接口等。

（2）基本问题主要包括：

① 表示物体和自动产生供阅览的照片的有效方法是什么？

② 接受输入和给出输出的有效方法是什么？

③ 怎样才能减小产生误解和由此产生的人为错误的风险？

④ 图表和其他工具怎样才能通过存储在数据集中的信息去理解物理现象？

9. 图形学和可视化计算

（1）主要内容：包括计算机图形学、可视化、虚拟现实、计算机视觉等 4 个学科子领域的研究内容。

（2）基本问题主要包括：

① 支撑图像产生以及信息浏览的更好模型。

② 如何提取科学的（计算和医学）和更抽象的相关数据？

③ 图像形成过程的解释和分析方法。

10. 智能系统

（1）主要内容：包括约束可满足性问题、知识表示和推理、Agent、自然语言处理、机器学习和神经网络、人工智能规划系统和机器人学等。

（2）基本问题主要有：

① 基本的行为模型是什么？如何建造模拟它们的机器？

② 规则评估、推理、演绎和模式计算在多大程度上描述了智能？

③ 通过这些方法模拟行为的机器的最终性能如何？

④ 传感数据如何编码才使得相似的模式有相似的代码？

⑤ 电机编码如何与传感编码相关联？

⑥ 学习系统的体系结构怎样？

⑦ 这些系统是如何表示它们对这个世界的理解的？

11. 信息管理

（1）主要内容：包括信息模型与信息系统、数据库系统、数据建模、关系数据库、数据库查询语言、关系数据库设计、事务处理、分布式数据库、数据挖掘、信息存储与检索、超文本和超媒体、多媒体信息与多媒体系统、数字图书馆等。

（2）基本问题主要包括：

① 使用什么样的建模概念来表示数据元素及其相互关系？

② 怎样把基本操作（如存储、定位、匹配和恢复）组合成有效的事务？

③ 这些事务怎样才能与用户有效地进行交互？

④ 高级查询如何翻译成高质量的程序？
⑤ 哪种机器体系结构能够进行有效的恢复和更新？
⑥ 怎样保护数据，以避免非授权访问、泄露和破坏？
⑦ 如何保护大型的数据库，以避免由于同时更新引起的不一致性？
⑧ 当数据分布在许多机器上时如何保护数据、保证性能？
⑨ 文本如何索引和分类才能够进行有效的恢复？

12. 软件工程

（1）主要内容：包括软件过程、软件需求与规格说明、软件设计、软件验证、软件演化、软件项目管理、软件开发工具与环境、基于构件的计算、形式化方法、软件可靠性、专用系统开发等。

（2）基本问题主要包括：
① 程序和程序设计系统发展背后的原理是什么？
② 如何证明一个程序或系统满足其规格说明？
③ 如何编写不忽略重要情况且能用于安全分析的规格说明？
④ 软件系统是如何历经不同的各代进行演化的？
⑤ 如何从可理解性和易修改性着手设计软件？

13. 社会和职业的问题

（1）主要内容：包括计算的历史、计算的社会背景、分析方法和工具、专业和道德责任、基于计算机系统的风险与责任、知识产权、隐私与公民的自由、计算机犯罪、与计算有关的经济问题、哲学框架等。

（2）基本问题主要包括：
① 计算学科本身的文化、社会、法律和道德的问题；
② 有关计算的社会影响问题，以及如何评价可能的一些答案的问题；
③ 哲学问题；
④ 技术问题以及美学问题。

14. 科学计算

（1）主要内容：包括数值分析、运筹学、模拟和仿真、高性能计算。

（2）基本问题主要包括：
① 如何精确地以有限的离散过程近似表示连续和无限的离散过程？
② 如何处理这种近似产生的错误？
③ 给定某一类方程在某精确度水平上能以多快的速度求解？
④ 如何实现方程的符号操作，如积分、微分以及到最小项的归约？
⑤ 如何把这些问题的答案包含到一个有效的、可靠的、高质量的数学软件包中？

2.4 计算学科中的典型问题及其相关内容

在人类社会的发展过程中，人们提出过许多具有深远意义的科学问题，其中也对计算学科一些分支领域的形成和发展起了重要的作用。另外，在计算学科的发展过程中，

为了便于对计算学科中有关问题和概念的本质的理解，人们还给出了不少反映该学科某一方面本质特征的典型实例，本书将它们一并归于计算学科的典型问题。计算学科典型问题的提出及研究，不仅有助于我们深刻地理解计算学科，而且还对该学科的发展有着十分重要的推动作用。下面分别对图论中有代表性的哥德斯堡七桥问题，算法与算法复杂性领域中有代表性的梵天（Hanoi，又译为汉诺）塔问题，算法复杂性中的难解性问题、*P* 类问题和 *NP* 类问题，证比求易算法，*P*=？*NP* 问题，旅行商问题与组合爆炸问题，进程同步中的生产者—消费者问题、哲学家共餐问题，程序设计中的 GOTO 语句等问题及其相关内容进行分析。计算学科中其他的典型问题，请读者参考有关资料。

2.4.1 哥尼斯堡七桥问题

17 世纪的东普鲁士有一座哥尼斯堡（Konigsberg）城（现为俄国的加里宁格勒（Kaliningrad）城），城中有一座奈佛夫（Kneiphof）岛，普雷格尔（Pregol）河的两条支流环绕其旁，并将整个城市分成北区、东区、南区和岛区 4 个区域，全城共有 7 座桥将 4 个城区相连起来，如图 2.1 所示。人们常通过这 7 座桥到各城区游玩，于是产生了一个有趣的数学难题：寻找走遍这 7 座桥，且只许走过每座桥一次，最后又回到原出发点的路径。该问题就是著名的“哥尼斯堡七桥问题”。

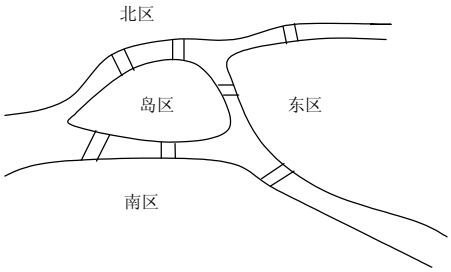


图 2.1　哥尼斯堡地图

1736 年，大数学家列昂纳德 • 欧拉（L.Euler）发表了关于“哥尼斯堡七桥问题”的论文——《与位置几何有关的一个问题的解》（*Solutio Problematis ad Geomertriam Situs Pertinentis*），他在文中指出，从一点出发不重复地走遍七桥，最后又回到原出发点是不可能的。

为了解决哥德斯堡七桥问题，欧拉用 4 个字母 A、B、C、D 代表 4 个城区，并用 7 条线表示 7 座桥，如图 2.2 所示。在图 2.2 中，只有 4 个点和 7 条线，这样做是基于该问题本质考虑的，它抽象出问题最本质的东西，忽视问题非本质的东西（如桥的长度等），从而将哥尼斯堡七桥问题抽象为一个数学问题，即经过图中每边一次且仅一次的回路问题了。欧拉在论文中论证了这样的回路是不存在的，后来，人们把有这样回路的图称为欧拉图。

欧拉在论文中将问题进行了一般化处理，即对给定的任意一个河道图与任意多座桥，判定可能不可能每座桥恰好走过一次，并用数学方法给出了 3 条判定的规则：

（1）如果通奇数座桥的地方不止两个，满足要求的路线是找不到的。

（2）如果只有两个地方通奇数座桥，可以从这两个地方之一出发，找到所要求的路线。

（3）如果没有一个地方是通奇数座桥的，则无论从哪里出发，所要求的路线都能实

现。

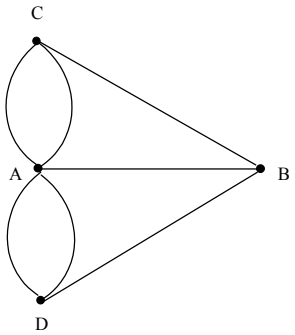


图 2.2　图

欧拉的论文为图论的形成奠定了基础。今天，图论已广泛地应用于计算学科、运筹学、信息论、控制论等学科之中，并已成为我们对现实问题进行抽象的一个强有力的数学工具。随着计算学科的发展，图论在计算学科中的作用越来越大，同时，图论本身也得到了充分的发展。

在图论中还有一个很著名的“哈密尔顿回路问题”。该问题是爱尔兰著名学者威廉·哈密尔顿爵士（W.R.Hamilton）1859 年提出的一个数学问题。其大意是：在某个图 G 中，能不能找到这样的路径，即从一点出发不重复地走过所有的结点，最后又回到原出发点。“哈密尔顿回路问题”与“欧拉回路问题”看上去十分相似，然而又是完全不同的两个问题。“哈密尔顿回路问题”是访问每个结点一次，而“欧拉回路问题”是访问每条边一次。对图 G 是否存在“欧拉回路”前面已给出充分必要条件，而对图 G 是否存在“哈密尔顿回路”至今仍未找到满足该问题的充分必要条件。

2.4.2　梵天塔问题

相传印度教的天神梵天在创造地球这一世界时，建了一座神庙，神庙里竖有三根宝石柱子，柱子由一个铜座支撑。梵天将 64 个直径大小不一的金盘子，按照从大到小的顺序依次套放在第一根柱子上，形成一座金塔（如图 2.3 所示），即所谓的梵天塔（又称汉诺塔）。天神让庙里的僧侣们将第一根柱子上的 64 个盘子借助第二根柱子全部移到第三根柱子上，既将整个塔迁移，同时定下 3 条规则：

（1）每次只能移动一个盘子；

（2）盘子只能在三根柱子上来回移动，不能放在他处；

（3）在移动过程中，三根柱子上的盘子必须始终保持大盘在下，小盘在上。

天神说：“当这 64 个盘子全部移到第三根柱子上后，世界末日就要到了”。这就是著名的梵天塔问题。

用计算机求解一个实际问题，首先要从这个实际问题中抽象出一个数学模型，然后设计一个解此数学模型的算法，最后根据算法编写程序，经过调试、编译、连接和运行，从而完成该问题的求解。从实际问题中抽象出一个数学模型的实质，其实就是要用数学的方法抽取其主要的、本质的内容，最终实现对该问题的正确认识。

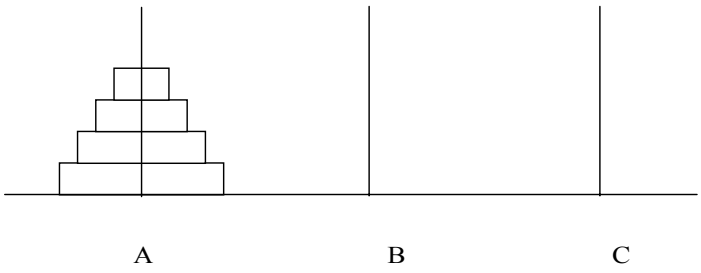


图 2.3　梵天塔

梵天塔问题是一个典型的只有用递归方法（而不能用其他方法）来解决的问题。递归是计算学科中的一个重要概念。所谓递归，就是将一个较大的问题归约为一个或多个子问题的求解方法。当然，要求这些子问题比原问题简单一些，且在结构上与原问题相同。

根据递归方法，我们可以将 64 个盘子的梵天塔问题转化为求解 63 个盘子的梵天塔问题，如果 63 个盘子的梵天塔问题能够解决，则可以先将 63 个盘子先移动到第二个柱子上，再将最后一个盘子直接移动到第三个柱子上，最后又一次将 63 个盘子从第二个柱子移动到第三个柱子上，这样则可以解决 64 个盘子的梵天塔问题。依此类推，63 个盘子的梵天塔求解问题可以转化为 62 个盘子的梵天塔求解问题，62 个盘子的梵天塔求解问题又可以转化为 61 个盘子的梵天塔求解问题，直到 1 个盘子的梵天塔求解问题。再由 1 个盘子的梵天塔的求解求出 2 个盘子的梵天塔，直到解出 64 个盘子的梵天塔问题。

下面，用 C 语言对该问题的求解算法进行描述。

```
hanoi(int n,char left,char middle,char right)
{
if(n==1) move(1,one,_,three);
else
{
hanoi(n-1,left,right,middle);
move(1,left,_,right);
hanoi(n-1,middle,left,right);
}
}
```

注：*n* 表示 *n* 个盘子的梵天塔问题，left 表示第一个柱子，middle 表示第二个柱子，right 表示第三个柱子。函数 hanoi(n-1,left,right,middle)表示 *n*-1 阶梵天塔从第一个柱子借助第三个柱子先移到第二个柱子上，函数 move(1,left,_,right)表示将第一个柱子上最后一个盘子直接放到第三个柱子上，函数 hanoi(n-1,middle,left,rignt)表示 *n*-1 个盘子从第二个柱子借助第一个柱子移到第三个柱子上。

在以上 C 语言描述的算法基础上，作适当扩充就可以形成一个完整的程序，经过编译和连接后，计算机就可以执行这个程序，并严格地按照递归的方法将问题求解出来。

现在的问题是当 *n*=64 时，即有 64 个盘子时，需要移动多少次盘子，要用多少时间。

按照上面的算法，*n* 个盘子的梵天塔问题需要移动的盘子数是 *n*-1 个盘子的梵天塔

问题需要移动的盘子数的 2 倍加 1。于是

$$\begin{aligned} h(n)&=2h(n\text{-}1)+1\\ &=2(2h(n\text{-}2)+1)+1=2^2h(n\text{-}2)+2+1\\ &=2^3h(n\text{-}3)+2^2+2+1\\ &\cdots\cdots\\ &=2^nh(0)+2^{n-1}+\ldots+2^2+2+1\\ &=2^{n-1}+\ldots+2^2+2+1=2^n\text{-}1 \end{aligned}$$

因此，要完成梵天塔的搬迁，需要移动盘子的次数为：

$$2^{64}\text{-}1=18446744073709551615$$

如果每秒移动一次，一年有 31536000 秒，则僧侣们一刻不停地来回搬动，也需要花费大约 5849 亿年的时间。假定计算机以每秒 1000 万个盘子的速度进行搬迁，则需要花费大约 58490 年的时间。

就这个例子，读者可以了解到理论上可以计算的问题，实际上并不一定能行，这属于算法复杂性方面的研究内容。

2.4.3 算法复杂性中的难解性问题、*P* 类问题和 *NP* 类问题

算法复杂性包括算法的空间以及时间两方面的复杂性问题，梵天塔问题主要讲的是算法的时间复杂性。

关于梵天塔问题算法的时间复杂度，可以用一个指数函数 $O(2^n)$来表示，显然，当 n 很大（如 10000）时，计算机是无法处理的。相反，当算法的时间复杂度的表示函数是一个多项式，如 $O(n^2)$时，则可以处理。因此，一个问题求解算法的时间复杂度大于多项式（如指数函数）时，算法的执行时间将随 n 的增加而急剧增长，以致即使是中等规模的问题也不能求解出来，于是在计算复杂性中，将这一类问题称为难解性问题。人工智能领域中的状态图搜索问题（解空间的表示或状态空间搜索问题）就是一类典型的难解性问题。

在计算复杂性理论中，将所有可以在多项式时间内求解的问题称为 *P* 类问题，而将所有在多项式时间内可以验证的问题称为 *NP* 类问题。由于 *P* 类问题采用的是确定性算法，*NP* 类问题采用的是非确定性算法，而确定性算法是非确定性算法的一种特例，因此，可以断定 $P\subseteq NP$。

2.4.4 证比求易算法

1. 证比求易算法

为了更好地理解计算复杂性的有关概念，我国学者洪加威曾经讲了一个被人称为“证比求易算法”的童话，用来帮助读者理解计算复杂性的有关概念，具体内容如下。

从前，有一个酷爱数学的年轻国王向邻国一位聪明美丽的公主求婚。公主出了这样一道题：求出 48 770 428 433 377 171 的一个真因子。若国王能在一天之内求出答案，公主便接受他的求婚。国王回去后立即开始逐个数地进行计算，他从早到晚，共算了三万多个数，最终还是没有结果。国王向公主求情，公主将答案相告：223 092 827 是它的一个真因子。国王很快就验证了这个数确能除尽 48 770 428 433 377 171。公主说：“我再给你一次机会，如果还求不出，将来你只好做我的证婚人了”。国王立即回国，并向时任宰相的大数学家求教，大数学家在仔细地思考后认为这个数为 17 位，则最小的一个真

因子不会超过 9 位，于是他给国王出了一个主意：按自然数的顺序给全国的老百姓每人编一个号发下去，等公主给出数目后，立即将它们通报全国，让每个老百姓用自己的编号去除这个数，除尽了立即上报，赏金万两。最后，国王用这个办法求婚成功。

2. 顺序算法和并行算法

在“证比求易算法”的故事中，国王最先使用的是一种顺序算法，其复杂性表现在时间方面，后面由宰相提出的是一种并行算法，其复杂性表现在空间方面。直觉上，我们认为顺序算法解决不了的问题完全可以用并行算法来解决，甚至会想，并行计算机系统求解问题的速度将随着处理器数目的不断增加而不断提高，从而解决难解性问题，其实这是一种误解。当将一个问题分解到多个处理器上解决时，由于算法中不可避免地存在必须串行执行的操作，从而大大地限制了并行计算机系统的加速能力。下面，用阿达尔（G.Amdahl）定律来说明这个问题。

3. 阿达尔定律

设 f 为求解某个问题的计算存在的必须串行执行的操作占整个计算的百分比，p 为处理器的数目，S_p 为并行计算机系统最大的加速能力（单位：倍），则

$$S_p \le \frac{1}{f+\frac{1-f}{p}}$$

设 f=1%，$p\to\infty$，则 S_p=100。这说明即使在并行计算机系统中有无穷多个处理器，解决这个串行执行操作仅占全部操作 1%，其解题速度与单处理器的计算机相比最多也只能提高一百倍。因此，对难解性问题而言，单纯的提高计算机系统的速度是远远不够的，而降低算法复杂度的数量级才是最关键的问题。

2.4.5 P=？NP

1. P = ？NP

在“证比求易算法”中，对公主给出的数进行验证，显然是在多项式时间内可以解决的问题，因此，这类问题属于 NP 类问题。

现在，P=NP 是否成立的问题是计算学科和当代数学研究中最大的悬而未决的问题之一。

如果 P=NP，则所有在多项式时间内可验证的问题都将是在多项式时间内可求解（或可判定）的问题。大多数人不相信 P=NP，因为人们已经投入了大量的精力为 NP 中的某些问题寻找多项式时间算法，但没有成功。然而，要证明 $P\neq NP$，目前还无法做到这一点。

在 P=?NP 问题上，库克（S.A.Cook）等人于 20 世纪 70 年代初取得了重大的进展，他们认为 NP 类中的某些问题的复杂性与整个类的复杂性有关，当这些问题中的任何一个存在多项式时间算法时，则所有这些 NP 问题都是多项式时间可解的，这些问题被称为 NP 完全性问题。

NP 完全性问题在理论和实践两方面都具有重要的研究意义。历史上第一个 NP 完全性问题是库克于 1971 年提出的可满足性问题。

可满足性问题就是判定一个布尔公式是否是可满足的。它可以形式化地表示为：

SAT={< ϕ > | ϕ 是可满足的布尔公式}

关于可满足性问题和 *NP* 问题的联系，库克给出并证明了这样的定理：

$SAT \in P$　当且仅当 $P=NP$

1982 年，库克因其在计算复杂性理论方面（主要是在 *NP* 完全性理论方面）的奠基性工作而荣获 ACM 图灵奖。

在库克工作的影响下，卡普（R.Karp）随后证明了 21 个有关组合优化的问题，也是 *NP* 完全性问题，从而加强和发展了 *NP* 完全性理论。卡普由于在计算复杂性理论、算法设计与分析、随机化算法等方面的创造性贡献，于 1985 年获 ACM 图灵奖。

现在，在计算科学、数学、逻辑学以及运筹学领域中已发现有总数多达数千个的 *NP* 完全性问题。其中有代表性的有：哈密尔顿回路问题、旅行商问题（也称货郎担问题）、划分问题、带优先级次序的处理机调度问题、顶点覆盖问题等。

2. 计算复杂性理论一个重要的应用领域——密码学

计算复杂性理论在密码学研究领域起了十分重要的作用，它给密码研究人员指出了寻找难计算问题的方向，并促使研究人员在该领域取得了革命性的成果。公钥密码系统就是其中的典型例子。

在公钥密码系统中，解密密钥不同于加密密钥，而且很难从加密密钥求出解密密钥。这样，每个人只要建立一对加密密钥和解密密钥，当甲要给乙发一条报文时，只要甲在公共密钥簿上找到乙的加密密钥，并用该加密密钥加密报文，然后发送到乙。由于乙是惟一知道解密密钥的人，因此，只有他能够解密这个报文，从而实现报文的安全传递，这一技术现已在民用和军用系统中得到了广泛的应用。

2.4.6　旅行商问题与组合爆炸问题

旅行商问题（Traveling Salesman Problem，简称 TSP）是威廉·哈密尔顿爵士和英国数学家克克曼（T.P.Kirkman）于 19 世纪初提出的一个数学问题。这是一个典型的 *NP* 完全性问题。其大意是：有若干个城市，任何两个城市之间的距离都是确定的，现要求一旅行商从某城市出发，必须经过每一个城市且只能在每个城市逗留一次，最后回到原出发城市。问如何事先确定好一条最短的路线，使其旅行的费用最少。

人们在考虑解决这个问题时，一般首先想到的最原始的一种方法就是：列出每一条可供选择的路线（即对给定的城市进行排列组合），计算出每条路线的总里程，最后从中选出一条最短的路线。假设现在给定的 4 个城市分别为 A、B、C 和 D，各城市之间的距离为已知数（如图 2.4 所示）。我们可以通过一个组合的状态空间图来表示所有的组合（如图 2.5 所示）。从图中不难看出，可供选择的路线共有 6 条，从中很快可以选出一条总距离最短的路线。由此推算，我们若设城市数目为 n 时，那么组合路径数则为 $(n-1)!$。很显然，当城市数目不多时要找到最短距离的路线并不难，但随着城市数目的不断增大，组合路线数将呈指数级数规律急剧增长，以至达到无法计算的地步，这就是所谓的“组合爆炸问题”。假设现在城市的数目增为 20 个，组合路径数则为 $(20-1)!\approx 1.216\times 10^{17}$，如此庞大的组合数目，若计算机以每秒检索 1000 万条路线的速度计算，也需要花上 386 年的时间。

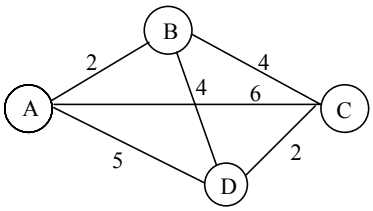
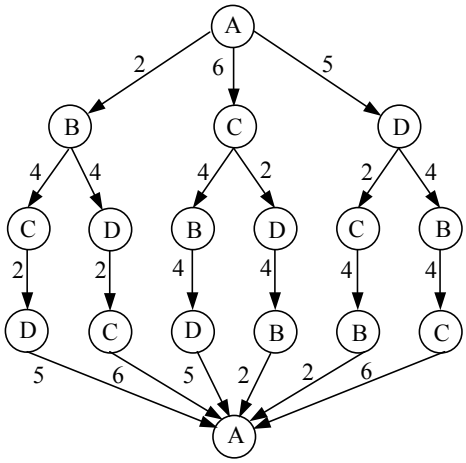


图 2.4　4 城市交通图



路径：ABCDA　总距离：13　路径：ABDCA　总距离：14
路径：ACBDA　总距离：19　路径：ACDBA　总距离：14
路径：ADCBA　总距离：13　路径：ADBCA　总距离：19

图 2.5　组合路径图

据文献介绍，1998 年，科学家们成功地解决了美国 13509 个城市之间的 TSP 问题，2001 年又解决了德国 15112 个城市之间的 TSP 问题。但这一工程代价也是巨大的，据报道，解决 15112 个城市之间的 TSP 问题，共使用了美国 Rice 大学和普林斯顿大学之间网络互连的、由速度为 500MHz 的 Compaq EV6 Alpha 处理器组成的 110 台计算机，所有计算机花费的时间之和为 22.6 年。

TSP 是最有代表性的优化组合问题之一，它的应用已逐步渗透到各个技术领域和我们的日常生活中，至今还有不少学者在从事这方面的研究工作，一些项目还得到美国军方的资助。就实际应用而言，一个典型的例子就是机器在电路板上钻孔的调度问题（注：在该问题中，钻孔的时间是固定的，只有机器移动时间的总量是可变的），在这里，电路板上要钻的孔相当于 TSP 中的“城市”，钻头从一个孔移到另一个孔所耗的时间相当于 TSP 中的“旅行费用”。在大规模生产过程中，寻找最短路径能有效地降低成本，这类问题的解决还可以延伸到其他行业中去，如运输业、后勤服务业等。然而，由于 TSP 会产生组合爆炸的问题，因此寻找切实可行的简化求解方法就成为问题的关键。

2.4.7　生产者－消费者问题与“哲学家共餐”问题

1. 生产者－消费者问题

1965 年，戴克斯特拉在他著名的论文《协同顺序进程》（*Cooperating Sequential*

Processes）中用生产者－消费者问题（Producer－Consumer Problem）对并发程序设计中进程同步的最基本问题，即对多进程提供（或释放）以及使用计算机系统中的软硬件资源（如数据、I/O 设备等）进行了抽象的描述，并使用信号灯的概念解决了这一问题。

在生产者－消费者问题中，所谓消费者是指使用某一软硬件资源时的进程，而生产者是指提供（或释放）某一软硬件资源时的进程。

在生产者－消费者问题中，有一个重要的概念，即信号灯，它借用了火车信号系统中的信号灯来表示进程之间的互斥。

在本书第 7 章的形式化技术中，我们还将对生产者－消费者问题作进一步的介绍，并用有限自动机和 Petri 网解决只有一个生产者和一个消费者的简单问题。

2. “哲学家共餐”问题

在提出生产者－消费者问题后，戴克斯特拉针对多进程互斥地访问有限资源（如 I/O 设备）的问题又提出并解决了一个被人称之为“哲学家共餐”（Dining Philosopher）的多进程同步问题。

对哲学家共餐问题可以作这样的描述：5 个哲学家围坐在一张圆桌旁，每个人的面前摆有一碗面条，碗的两旁各摆有一只筷子（注：戴克斯特拉原来提到的是叉子和意大利面条，因有人习惯用一个叉子吃面条，于是后来的研究人员又将叉子和意大利面条改写为中国筷子和面条）。

假设哲学家的生活除了吃饭就是思考问题（这是一种抽象，即对该问题而言其他活动都无关紧要），而吃饭的时候需要左手拿一只筷子，右手拿一只筷子，然后开始进餐。吃完后又将筷子摆回原处，继续思考问题。那么，一个哲学家的生活进程可表示为：

（1）思考问题；

（2）饿了停止思考，左手拿一只筷子（如果左侧哲学家已持有它，则需等待）；

（3）右手拿一只筷子（如果右侧哲学家已持有它，则需等待）；

（4）进餐；

（5）放右手筷子；

（6）放左手筷子；

（7）重新回到思考问题状态（1）。

现在的问题是：如何协调 5 个哲学家的生活进程，使得每一个哲学家最终都可以进餐。

考虑下面的两种情况：

（1）按哲学家的活动进程，当所有的哲学家都同时拿起左手筷子时，则所有的哲学家都将拿不到右手的筷子，并处于等待状态，那么哲学家都将无法进餐，最终饿死。

（2）将哲学家的活动进程修改一下，变为当右手的筷子拿不到时，就放下左手的筷子，这种情况是不是就没有问题？不一定，因为可能在一个瞬间，所有的哲学家都同时拿起左手的筷子，则自然拿不到右手的筷子，于是都同时放下左手的筷子，等一会，又同时拿起左手的筷子，如此这样永远重复下去，则所有的哲学家一样都吃不到饭。

以上两个方面的问题，其实反映的是程序并发执行时进程同步的两个问题，一个是死锁（Deadlock），另一个是饥饿（Starvation）。

为了提高系统的处理能力和机器的利用率，并发程序被广泛地使用，因此，必须彻底解决并发程序中的死锁和饥饿问题。于是，人们将 5 个哲学家问题推广为更一般性的

n 个进程和 *m* 个共享资源的问题，并在研究过程中给出了解决这类问题的不少方法和工具，如 Petri 网、并发程序语言等工具。

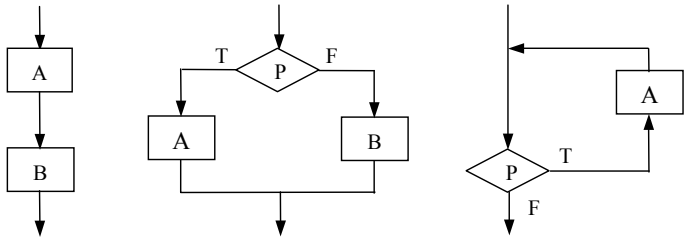
与程序并发执行时进程同步有关的经典问题还有：读－写者问题（Reader－Writer Problem）、理发师睡眠问题（Sleeping Barber Problem）等。

2.4.8 GOTO 语句的问题以及程序设计方法学

在计算机诞生的初期，计算机主要用于科学计算，程序的规模一般都比较小，那时的程序设计要说有方法的话，也只能说是一种手工式的设计方法。20 世纪 60 年代，计算机软硬技术得到了迅速的发展，其应用领域也急剧扩大，这给传统的手工式程序设计方法带来了挑战。

1966 年，C.BÖhm 和 G.Jacopini 发表了关于“程序结构”的重要论文《带有两种形成规则的图灵机和语言的流程图》（*Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules*），给出了任何程序的逻辑结构都可以用 3 种最基本的结构（如图 2.6 所示），即顺序结构、选择结构和循环结构来表示的证明。

以 BÖhm 和 Jacopini 的工作为基础，1968 年，戴克斯特拉经过深思熟虑后，在给《ACM 通讯》编辑的一封信中，首次提出了“GOTO 语句是有害的”（*Go to Statement Considered Harmful*）问题，该问题在《ACM 通讯》杂志上发表后，引发了激烈的争论，不少著名的学者参与了讨论。



(a) 顺序结构　　(b) 选择结构　　(c) 循环结构

图 2.6　程序的 3 种基本结构

经过 6 年的争论，1974 年，著名计算机科学家、图灵奖获得者克努特（D.E.Knuth）教授在他发表的有影响力的论文《带有 GOTO 语句的结构化程序设计》（*Structured Programming with Goto Statements*）中对这场争论作了较为全面而公正的论述：滥用 GOTO 语句是有害的，完全禁止也不明智，在不破坏程序良好结构的前提下，有控制地使用一些 GOTO 语句，就有可能使程序更清晰，效率也更高，关于“GOTO 语句”的争论，其焦点应当放在程序的结构上，好的程序应该是逻辑正确、结构清晰、朴实无华。

关于“GOTO 语句”问题的争论直接导致了一个新的学科分支领域，即程序设计方法学的产生。程序设计方法学是对程序的性质及其设计的理论和方法进行研究的学科，它是计算学科发展的必然产物，也是计算机科学与技术方法论中的重要内容。

2.5 人工智能中的若干哲学问题

在计算学科诞生后，为解决人工智能中一些激烈争论的问题，图灵和西尔勒又分别提出了能反映人工智能本质特征的两个著名的哲学问题，即“图灵测试”和西尔勒的“中文屋子”，沿着图灵等人对“智能”的理解，人们在人工智能领域取得了长足的进展，其中“深蓝（Deep Blue）”战胜国际象棋大师卡斯帕罗夫（G.Kasparov）就是一个很好的例证。

2.5.1 图灵测试

图灵于 1950 年在英国《心》（*Mind*）杂志上发表了《计算机器和智能》（*Computing Machinery and Intelligence*）一文，文中提出了“机器能思维吗？”这样一个问题，并给出了一个被称作“模仿游戏（Imitation Game）”的试验，后人称之为“图灵测试（Turing Test）”。

这个游戏需由 3 个人来完成：一个男人（A），一个女人（B），一个性别不限的提问者（C）。提问者（C）呆在与其他两个游戏者相隔离的房间里。游戏的目标是让提问者通过对其他两人的提问来鉴别其中哪个是男人，哪个是女人。为了避免提问者通过他们的声音、语调轻易地作出判断，最好是在提问者和两游戏者之间通过一台电传打字机来进行沟通。提问者只被告知两个人的代号为 X 和 Y，游戏的最后他要作出“X 是 A，Y 是 B”或“X 是 B，Y 是 A”的判断。

提问者可以提出下列问题：“请 X 回答，你的头发的长度？”，如果 X 实际上是男人（A），那么他为了给提问者造成错觉，可能会这样回答：“我的头发很长，大约有 9 英寸”。如果对女人（B）来说，游戏的目标是帮助提问者，那么她可能会作出真实的回答，并且在答案后面加上“我是女人，不要相信那个人”之类的提示。但也许这样也无济于事，因为男人（A）同样也可以加上类似的提示。

现在，把上面这个游戏中的男人（A）换成一部机器来扮演，如果提问者在与机器、女人的游戏中作出的错误判断与在男人、女人之间的游戏中作出错误判断的次数是相同的，那么，就可以判定这部机器是能够思维的。

图灵关于“图灵测试”的论文发表后引发了很多的争论，以后的学者在讨论机器思维时大多都要谈到这个游戏。

“图灵测试”不要求接受测试的思维机器在内部构造上与人脑一样，它只是从功能的角度来判定机器是否能思维，也就是从行为主义这个角度来对“机器思维”进行定义。尽管图灵对“机器思维”的定义是不够严谨的，但他关于“机器思维”定义的开创性工作对后人的研究具有重要意义，因此，一些学者认为，图灵发表的关于“图灵测试”的论文标志着现代机器思维问题讨论的开始。

根据图灵的预测，到 2000 年，此类机器能通过测试。现在，在某些特定的领域，如博弈领域，“图灵测试”已取得了成功，1997 年，IBM 公司研制的计算机“深蓝”就战胜了国际象棋冠军卡斯帕罗夫。

前面介绍过，符号主义者认为认知是一种符号处理过程，人类思维过程也可用某种符号来描述，即思维就是计算（认知就是计算），这种思想构成了人工智能的哲学基础之一。其实，历史上把推理作为人类精神活动的中心，把一切推理都归结于某种计算的想法一直吸引着西方的思想家。然而，由于人们对心理学和生物学的认识还很不成熟，

对人脑的结构还没有真正的了解，更无法建立起人脑思维完整的数学模型。因此，到目前为止，人们对人工智能的研究并无实质性的突破。

在未来，如果我们能像图灵揭示计算本质那样揭示人类思维的本质，即“能行”思维，那么制造真正思维机器的日子也就不长了。可惜要对人类思维的本质进行描述，还是相当遥远的事情。

2.5.2 西尔勒的“中文屋子”

美国哲学家约翰·西尔勒（J.R.Searle）根据人们在研究人工智能模拟人类认知能力方面的不同观点，将有关人工智能的研究划分为强人工智能（Strong Artificial Intelligence，简称强 AI）和弱人工智能（Soft Artificial Intelligence，简称弱 AI）两个派别。

在研究意识方面，弱 AI 认为：计算机的主要价值在于它为我们提供了一个强大的工具；强 AI 的观点则是：计算机不仅是一个工具，形式化的计算机是具有意识的。

1980 年，西尔勒在《行为科学和脑科学》（*Behavioral and Brain Sciences*）杂志上发表了《心、脑和程序》（*Minds*、*Brains and Programs*）的论文，在文中，他以自己为主角设计了一个“中文屋子（Chinese Room）”的假想试验来反驳强 AI 的观点：

假设西尔勒被单独关在一个屋子里，屋子里有序地堆放着足量的汉语字符，而他对中文可谓是一窍不通。这时屋外的人递进一串汉语字符，同时还附了一本用英文写的处理汉语字符的规则（注：英语是西尔勒的母语），这些规则将递进来的字符和屋子里的字符之间的处理作了纯形式化的规定，西尔勒按规则指令对这些字符进行一番搬弄之后，将一串新组成的字符送出屋外。事实上他根本不知道送进来的字符串就是屋外人提出的“问题”，也不知道送出去的就是所谓“问题的答案”。又假设西尔勒很擅长按照指令娴熟地处理一些汉字符号，而程序设计师（即制定规则的人）又擅长编写程序（即规则），那么，西尔勒的答案将会与一个地道的中国人作出的答案没什么不同。但是，我们能说西尔勒真的懂中文吗？

西尔勒借用语言学的术语非常形象地揭示了“中文屋子”的深刻寓意：形式化的计算机仅有语法，没有语义。因此，他认为，机器永远也不可能代替人脑。作为以研究语言哲学问题而著称的分析哲学家西尔勒来自语言学的思考，的确给人工智能涉及的哲学和心理学问题提供了不少启发。

2.5.3 计算机中的博弈问题

1. 博弈的历史简介

博弈问题属于人工智能中一个重要的研究领域。从狭义上讲，博弈是指下棋、玩扑克牌、掷骰子等具有输赢性质的游戏；从广义上讲，博弈就是对策或斗智。计算机中的博弈问题，一直是人工智能领域研究的重点内容之一。

1913 年，数学家策墨洛（E.Zermelo）在第五届国际数学会议上发表了《关于集合论在象棋博弈理论中的应用》（*On an Application of Set Theory to Game of Chess*）的著名论文，第一次把数学和象棋联系起来，从此，现代数学出现了一个新的理论，即博弈论。

1950 年，“信息论”创始人香农（A.Shannon）发表了《国际象棋与机器》（*A Chess－Playing Machine*）一文，并阐述了用计算机编制下棋程序的可能性。

1956 年夏天，由麦卡锡（J.McCarthy）和香农等人共同发起的，在美国达特茅斯

（Dartmouth）大学举行的夏季学术讨论会上，第一次正式使用了“人工智能”这一术语，该次会议的召开对人工智能的发展起到了极大的推动作用。当时，IBM 公司的工程师塞缪尔（A. Samuel）也被邀请参加了“达特茅斯”会议，塞缪尔的研究专长正是电脑下棋。早在 1952 年，塞缪尔就运用博弈理论和状态空间搜索技术成功地研制了世界上第一个跳棋程序。该程序经不断地完善于 1959 年击败了它的设计者塞缪尔本人，1962 年，它又击败了美国一个州的冠军。

1970 年开始，ACM 每年举办一次计算机国际象棋锦标赛直到 1994 年（1992 年间断过一次），每年产生一个计算机国际象棋赛冠军，1991 年，冠军由 IBM 的“深思Ⅱ（Deep Thought Ⅱ）”获得。ACM 的这些工作极大地推动了博弈问题的深入研究，并促进了人工智能领域的发展。

2. “深蓝”与卡斯帕罗夫之战

北京时间 1997 年 5 月初，在美国纽约公平大厦，“深蓝”与国际象棋冠军卡斯帕罗夫交战，前者以两胜一负三平战胜后者。

“深蓝”是美国 IBM 公司研制的一台高性能并行计算机，它由 256（32 node*8）个专为国际象棋比赛设计的微处理器组成，据估计，该系统每秒可计算 2 亿步棋。“深蓝”的前身是“深思”，始建于 1985 年。1989 年，卡斯帕罗夫首战“深思”，后者败北。1996 年，在“深思”基础上研制出的“深蓝”曾再次与卡斯帕罗夫交战，并以 2：4 负于对手。

“深蓝”战胜卡斯帕罗夫后，以“深蓝”主管谭崇仁（C.J.Tan，美籍华人）、设计师许峰雄（C.B.Hsu，美籍华人）、象棋顾问本杰明（GM.J.Benjamin）及其他科学家、工程师加盟的 “深蓝队”获得奖金 70 万美元，卡斯帕罗夫获 40 万美元。另外，IBM 公司也从中获得了大约 5000 万美元的广告收益。因此，与其说是“深蓝”战胜了卡斯帕罗夫，还不如说是“深蓝队”战胜了卡斯帕罗夫。

3．博弈树搜索

国际象棋、西洋跳棋与围棋、中国象棋一样都属于双人完备博弈。所谓双人完备博弈就是两位选手对垒，轮流走步，其中一方完全知道另一方已经走过的棋步以及未来可能的走步，对弈的结果要么是一方赢（另一方输），要么是和局。

对于任何一种双人完备博弈，都可以用一个博弈树（与或树）来描述，并通过博弈树搜索策略寻找最佳解。

博弈树类似于状态图和问题求解搜索中使用的搜索树。搜索树上的第一个结点对应一个棋局，树的分支表示棋的走步，根节点表示棋局的开始，叶节点表示棋局的结束。一个棋局的结果可以是赢、输或者和局。

对于一个思考缜密的棋局来说，其博弈树是非常大的，就国际象棋来说，有 10^{120} 个结点（棋局总数），而对中国象棋来说，估计有 10^{160} 个结点，围棋更复杂，盘面状态达 10^{768}。计算机要装下如此大的博弈树，并在合理的时间内进行详细的搜索是不可能的。因此，如何将搜索树修改到一个合理的范围，是一个值得研究的问题，“深蓝”就是这类研究的成果之一。

4. 结论

“深蓝”战胜人类最伟大的棋手卡斯帕罗夫后，在社会上引起了轩然大波。一些人认为，机器的智力已超越人类，甚至还有人认为计算机最终将控制人类。其实人的智力

与机器的智力根本就是两回事，因为，人们现在对人的精神和脑的结构的认识还相当缺乏，更不用说对它用严密的数学语言来进行描述了，而电脑是一种用严密的数学语言来描述的计算机器。

如果我们不考虑人的精神和脑的结构这样的哲学和生物学问题，那么许多问题解释起来就很容易了。其实计算机就如汽车、飞机一样，人要超过这些机器设备所具有的能力是不现实的。就计算机而言，人要在计算能力上超过机器是不现实的，而对博弈问题来说，人在未来要战胜机器也是不现实的。

以上认识有助于我们真正理解计算机器的本质。就像人们知道汽车、飞机在造福人类的同时，也会对人类产生灾难一样（美国的“9.11”事件就是其中一例），计算机器也是如此，这就需要我们去正视这些问题，并通过各种途径来避免这类灾难的发生。不仅如此，我们还应当自觉地将它们应用于人类社会的进步和发展之中。

思 考 题

1．什么是科学问题？科学问题具备哪些主要特征？它有哪些作用？

2．图灵是如何揭示计算的本质的？

3．给出罗素悖论的形式化描述，并简要介绍其大意。

4．什么是“希尔伯特纲领”？

5．为什么说离散数学对计算技术的发展起着十分重要的作用？

6．计算学科的定义是什么？什么是计算学科的根本问题？

7．计算学科的各个主领域包括哪些基本问题？

8．欧拉是如何对“哥尼斯堡七桥问题”进行抽象的？

9．简述“欧拉回路”与“哈密尔顿回路”的区别。

10．以“梵天塔问题”为例，说明理论上可行的计算问题实际上并不一定能行。

11．什么是顺序程序设计？什么是并行程序设计？

12．什么是 *NP* 类问题？请举例说明。

13．“生产者－消费者问题”和“哲学家共餐问题”反映的是计算学科中的什么问题？

14．用图表示程序的 3 种基本结构。

15．“GOTO 语句问题”的提出直接导致了计算学科哪一个分支领域的产生？

16．“图灵测试”和“中文屋子”是如何从哲学的角度反映人工智能本质特征的？

17．举例说明计算机中的博弈问题。

18．为什么说人要在计算能力上超过计算机是不现实的？

第 3 章　计算学科中的 3 个学科形态

方法论在层次上有哲学方法论、一般科学技术方法论、具体科学技术方法论之分，它们相互依存、互为作用。在一般科学技术方法论中，抽象、理论和设计是其研究的主要内容。在本章中，我们以一般科学技术方法论为指导，阐述计算学科中的抽象、理论和设计 3 个过程（形态）的内容。

抽象、理论和设计 3 个学科形态（或过程）概括了计算学科中的基本内容，是计算学科认知领域中最基本（原始）的 3 个概念。不仅如此，它还反映了人们的认识是从感性认识（抽象）到理性认识（理论），再由理性认识（理论）回到实践中来的科学思维方法。

3.1　一个关于学生选课的例子

众所周知，人类的认识过程是从感性认识到理性认识，再回到实践中去的过程。感性认识是采用一定的方法，以可感知的文字或图形等形式对客观事物的特征进行描述，并通过构建模型而实现的。感性认识包括两个方面的内容：一方面是感性认识的认识方法（或工具）的建立；另一方面是采用已建立起来的认识方法来实现对客观世界的感性认识。

下面，我们举一个“学生选课”的例子，为计算机初学者学习和掌握方法论中最基本的内容——感性认识（抽象）、理性认识（理论）和实践（设计）的学习作个铺垫。

例 3.1　现给出“学生”和“课程”两个实体，它们的联系为：一个学生可以选修若干门课程，每门课程可以被任一学生所选修。请建立一个信息管理系统，以实现对“学生选课”这一信息的管理。

3.1.1　对“学生选课”例子的感性认识

1. 概念模型

概念模型用于信息世界的建模，是客观世界到信息世界的抽象。概念模型中的主要概念有：实体、属性、码、域、联系等。

实体：客观存在并可相互区别的事物。

属性：实体所具有的某一种特性。

码：能惟一标识实体的属性集。

域：属性的取值范围。

联系：指不同实体集之间的联系。两个实体之间的联系分为：一对一（1:1）、一对多（1:*N*）、多对多（*N*:*M*）3 类。

2. E-R 模型

在计算学科发展的过程中，为了实现对客观世界的感性认识，人们创造了不少认识客观世界的方法，这些方法极大地促进了人们对客观世界的认识。在数据库领域中，最常用的抽象方法有 E - R 方法（Entity-Relationship Approach），它是美籍华人陈平山（Peter

Pingshan Chen）于 1976 年提出的，是一种用 E-R 模型来描述客观世界并建立概念模型的抽象方法，该方法也被称为实体—联系模型（或 E-R 图）。

在 E-R 图中，实体用矩形表示，属性用椭圆形表示，联系用菱形表示，实体间的联系有一对一（1:1）、一对多（1:*N*）、多对多（*N*:*M*）3 种情况。

要实现对客观事物的感性认识，必须将客观世界（在例中客观世界就是“学生选课”）抽象为信息世界。

下面，用 E-R 模型来建立“学生选课”的概念模型（如图 3.1 所示），从而实现对这一例子的感性认识。

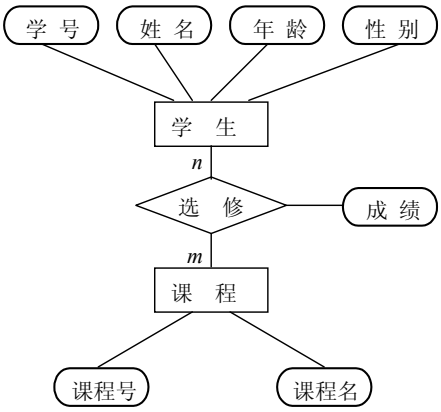


图 3.1　学生选课 E-R 图

3. 关系模型

概念模型不是机器世界中所支持的数据模型，而是客观世界到机器世界的一个中间层次，因此，它还需要转换成机器世界能支持的数据模型。在数据库领域中，数据库管理系统（DBMS）能支持的数据模型有：层次、网状、关系以及面向对象等数据模型。

本书只讨论最常用的关系模型。为此，还必须将例子的概念模型转变为关系数据库管理系统（RDBMS）所支持的数据模型，即关系模型。

关系模型支持的是一种二维表结构的数据模型，它由关系数据结构、关系数据操作和关系数据的完整性约束条件三部分组成。其中关系就是一张二维表。在关系模型中，客观世界的实体以及实体之间的各种联系均用关系来表示。常用的关系操作有：选择、投影、连接、并、交、差等查询操作以及更新操作。关系模型有 3 类完整性约束条件：实体完整性、参照完整性和用户定义的完整性。

根据概念模型向关系模型的转换规则，例子的概念模型（E-R 图）可以转换为下面的关系模型（关系的码用下划线标出）：

学生（学号，姓名，年龄，性别）；

课程（课程号，课程名）；

学生选课（学号，课程号，成绩）

概念模型是对现实原形的理想化，尽管理想化后的现实原形与现实事物有了质的区别，但它们总是现实事物的观念化，有现实背景，从严格意义上来说它们还是粗糙的、近似的。因此，将概念模型直接转换成关系模型，还不能说完全达到了对“学生选课”这一客观世界的理性认识，换言之，就是所转换的关系模型有可能还存在问题。

3.1.2 对“学生选课”例子的理性认识

1. 感性认识中存在的问题

在学生（学号，姓名，年龄，性别）关系中增加系名、系主任等属性时，即学生关系变为（学号，姓名，年龄，性别，系名，系主任）时，便开始出现以下问题。

（1）一个系刚成立，系主任已确定，但还未招学生时，则无法将系名和系主任的名字插入到数据库中（学生实体中学号为码，码不能缺），从而造成插入异常；

（2）当一个系的学生全部毕业，删除所有毕业生时，系名和系主任的名字也就删除了，从而造成删除异常。

（3）假若一个系有 1000 个学生，由于一个学生对应一个系名和系主任的名字，则该系系名和系主任的名字要重复 1000 次，冗余太大。

要解决以上一系列问题，必须使问题形式化，即内容与形式要分开。下面，给出一个简化了的关系模式的形式化定义，并在形式化的基础上讨论与上面问题有关的第一范式（1*NF*）、第二范式（2*NF*）和第三范式（3*NF*）等内容。

2. 关系模式的形式化定义

关系模式（*R*）是一个四元组，即

R=<*U*，*D*，*dom*，*F*>

其中：

（1）*U* 表示关系中所有属性的集合；

（2）*D* 表示属性集合 *U* 中属性所来自的域；

（3）*dom* 是属性到域的映射；

（4）*F* 是属性集合 *U* 上的一组数据依赖。

由于 *D*、*dom* 与模式设计关系不大，因此，可以将关系模式简单地表示为一个二元组 *R*=<*U*，*F*>。

1*NF* 的定义：作为一张二维表的关系，每一个分量必须是不可再分的数据项，满足这个条件的关系模式就属于 1*NF*。

2*NF* 的定义：若 *R*∈1*NF*，且每一个非主属性不存在对码的部分函数依赖，则 *R*∈2*NF*。在定义中，非主属性为不属于码的那些属性。

3*NF* 的定义：若 *R*∈2*NF*，且每一个非主属性不存在对码的传递函数依赖，则 *R*∈3*NF*。

3. 对“例子”问题的理性认识

显然例子最初是属于 1*NF*、2*NF*、3*NF*，但是当在学生属性集 *U* 中增加系名和系主任后，就出现了这样的传递函数依赖：学号（码）→系名，系名→系主任。因此，它就不属于 3*NF* 了。

不属于 3*NF* 的所有关系模型都会出现插入异常、删除异常和冗余的问题。因此，还必须依靠分解算法对模式进行分解，并满足 *3NF* 的要求。在数据依赖理论的指导下，可以完成模式的分解任务。就例子而言，可以再划分一个关系，即系（系号，系名，系主任名），从而满足了关系模式规范化的要求，实现了对例子的理性认识。

从概念模型向关系模型的转换，其实质是认识过程由感性认识（抽象）上升到理性认识（理论）的过程，这个过程包含两方面的内容：一方面是有

关理论的建立；另一方面是如何在理论的指导下，在具体的设计中，实现对客观世界的理性认识。前者是对科学研究而言的，而后者是对工程设计而言的。

3.1.3 “学生选课”系统的工程设计

在前面的例子中，在关系数据理论的指导下建立起正确的关系模型后，还要根据具体的关系数据库管理系统（如 Oracle、Informix、SyBase、FoxPro 等）对该模型进行定义，然后经过计算机处理，便可进行有关数据的输入、修改和查询工作。

下面，给出定义该模型的 SQL 语句：

```
CREATE TABLE STUDENT
( SNO   CHAR(9)   NOT NULL，
  SN      CHAR(16)，
  SAGE    INT，
  SEX     CHAR(1) )；
CREATE TABLE COURSE
( CNO   CHAR(6)   NOT NULL，
  CN      CHAR(22) )；
CREATE TABLE SC
( SNO   CHAR(9)   NOT NULL，
  CNO   CHAR(6)，
  GRADE   INT )；
CREATE TABLE DEPARTMENT
( DNO    CHAR(9)   NOT NULL，
  DN       CHAR(16)，
  DEAN   CHAR(8))；
```

以上语句为 SQL 语言的标准语句，可在任何支持 SQL 标准的数据库系统中运行，系统运行这些语句后，就将以上 3 个表的有关定义和约束条件存放在数据库中的数据字典中，供系统调用。接下来，便可以进行数据的输入、修改和查询，从而完成对“学生选课”的管理。下面介绍一个简单的查询：查询选修了“数据库”课程，并且成绩在 90 分以上的所有学生的学号和姓名。

```
SELECT SNO,SN
FROM STUDENT,SC,COURSE
WHERE CN='数据库' AND GRADE>90;
```

系统运行以上语句后，即可在屏幕上显示所求的结果。

以上“学生选课”管理系统的研制过程蕴含了人们对客观世界从感性认识（通过 E-R 图，实现对例子的抽象）到理性认识（在关系数据理论的指导下，通过建立更为适合的关系模型而实现对例子的理性认识），再由理性认识回到实践（在现实对“例子”的感性认识和理性认识后，编写程序完成“学生选课”管理信息系统的工作）中来的科学思维方法。

对“学生选课”这个简单例子的分析有助于我们从工程设计这个角度，来理解后面要介绍的计算学科中有关抽象、理论和设计 3 个过程的内容。

下面，分别从一般科学技术方法论的角度、计算学科的角度，对抽象、理论和设计3个过程（或Paradigm，即形态）进行论述，最后给出有关实例及3个过程（形态）的主要内容和简要分析，以加深读者对3个过程（形态）的理解。

3.2 抽象形态

3.2.1 一般科学技术方法论中有关抽象形态的论述

在一般科学技术方法论中，科学抽象是指在思维中对同类事物去除其现象的、次要的方面，抽取其共同的、主要的方面，从而做到从个别中把握一般，从现象中把握本质的认知过程和思维方法。科学抽象是科学认识由感性认识向理性认识飞跃的决定性环节。

抽象源于现实世界，源于经验，是对现实原形的理想化，尽管理想化后的现实原形与现实事物有了质的区别，但它们总是现实事物的概念化，有现实背景，从严格意义上来说还是粗糙的、近似的。因此，要实现对事物本质的认识还必须通过经验与理性的结合，实现从抽象到抽象的升华。尽管科学抽象还有待升华，但它仍然是科学认识的基础和决定性环节。

学科中的抽象形态包含着具体的内容，它们是学科中所具有的科学概念、科学符号和思想模型。

3.2.2 计算学科中有关抽象形态的论述

《计算作为一门学科》报告认为：理论、抽象和设计是我们从事本领域工作的3种主要形态（或称文化方式），它提供了我们定义计算学科的条件。按人们对客观事物认识的先后次序，我们将报告中的抽象列为第一个学科形态，理论列为第二个学科形态。抽象源于实验科学。按客观现象的研究过程，抽象形态包括以下4个步骤的内容：

（1）形成假设；

（2）建造模型并作出预测；

（3）设计实验并收集数据；

（4）对结果进行分析。

3.2.3 例子中有关抽象形态的主要内容及其简要分析

在“学生选课”例子中，有关抽象形态的内容可以用集合的方式表示为：

A={学生，属性，码，关系，学号，姓名，年龄，性别，课程，课程号，课程名，成绩，E-R图，“学生选课”E-R图，关系模型，“学生选课”关系模型，……}

对“学生选课”问题的抽象（感性认识）就是通过建立“学生选课”的E-R模型和关系模型来实现的，这一步是实现“学生选课”系统的关键。

3.3 理论形态

3.3.1 一般科学技术方法论中有关理论形态的论述

科学认识由感性阶段上升为理性阶段，就形成了科学理论。科学理论是经过实践检验的系统化了的科学知识体系，它是由科学概念、科学原理以及对这些概念、原理的理论论证所组成的体系。

理论源于数学，是从抽象到抽象的升华，它们已经完全脱离现实事物，不受现实事物的限制，具有精确的、优美的特征，因而更能把握事物的本质。

3.3.2 计算学科中有关理论形态的论述

在计算学科中，从统一合理的理论发展过程来看，理论形态包括以下 4 个步骤的内容：

（1）表述研究对象的特征（定义和公理）；

（2）假设对象之间的基本性质和对象之间可能存在的关系（定理）；

（3）确定这些关系是否为真（证明）；

（4）结论。

3.3.3 例子中有关理论形态的主要内容及简要分析

在与“学生选课”例子有关的关系数据库领域中，理论形态的主要内容可以用集合的方式表示为：

T={关系代数，关系演算，数据依赖理论……}

在数据库理论的指导下，我们就可以在“学生选课”关系模型（感性认识）的基础上，建立对“学生选课”问题的理性认识，从而为“学生选课”管理系统的设计奠定基础。

3.4 设计形态

3.4.1 一般科学技术方法论中有关设计形态的论述

1. 设计形态与抽象、理论两个形态存在的联系

设计源于工程，并用于系统或设备的开发，以实现给定的任务。

（1）设计形态（技术方法）和抽象、理论两个形态（科学方法）具有许多共同的特点。设计作为变革、控制和利用自然界的手段，必须以对自然规律的认识为前提（可以是科学形态的认识，也可以是经验形态的认识）。

（2）设计要达到变革、控制和利用自然界的目的，必须创造出相应的人工系统和人工条件，还必须认识自然规律在这些人工系统中和人工条件下的具体表现形式。所以，科学认识方法（抽象、理论两个形态），对具有设计形态的技术研究和技术开发是有作用的。

2. 设计形态的主要特征与抽象、理论两个形态的主要区别

（1）设计形态具有较强的实践性。

（2）设计形态具有较强的社会性。

（3）设计形态具有较强的综合性。

3.4.2 计算学科中有关设计形态的论述

在计算学科中，从为解决某个问题而实现系统或装置的过程来看，设计形态包括以下 4 个步骤的内容：

（1）需求分析；

（2）建立规格说明；

（3）设计并实现该系统；

（4）对系统进行测试与分析。

设计、抽象和理论 3 个形态针对具体的研究领域均起作用，在具体研究中，就是要在其理论的指导下，运用其抽象工具进行各种设计工作，最终的成果将是计算机的软硬件系统及其相关资料（如需求说明、规格说明和设计与实现方法说明等）。

3.4.3 例子中有关设计形态的主要内容及简要分析

在关系数据库中，有关设计形态的内容是指：在关系数据库理论的指导下，运用一定的抽象工具（如 E-R 图等）进行的各种设计工作。最终的内容包括 Oracle、Informix、SyBase、FoxPro 等各种关系数据库管理系统（DBMS）软件、应用软件（如学生、财务等具体的管理信息系统）以及相关资料（如需求说明、规格说明和设计与实现方法说明等资料）。

“学生选课”一例中，有关设计形态的内容是指：在数据库理论的指导下，运用 E-R 图和关系模型，实现对例子的感性认识和理性认识，最后借助某种关系 DBMS（如 Oracle 等），实现“学生选课”应用软件的编制。最终成果是“学生选课”应用软件以及相关资料（如需求说明书）。

就例子而言，其内容可以用集合的方式表示为：

D={“学生选课”应用软件，“学生选课”需求说明书……}

3.5 3个学科形态的内在联系

3.5.1 一般科学技术方法论中有关 3 个学科形态内在联系的简要论述

在计算机科学与技术方法论的原始命题中，蕴含着人类认识过程的两次飞跃，第一次飞跃是从物质到精神，从实践到认识的飞跃。这次飞跃包括两个决定性的环节：一个是科学抽象，另一个是科学理论。科学抽象是科学认识由感性阶段向理性阶段飞跃的决定性环节，当科学认识由感性阶段上升为理性阶段时，就形成了科学理论。第二次飞跃是从精神到物质，从认识到实践的飞跃。这次飞跃的实质对技术学科（计算学科就是一门技术学科）而言，其实就是要在理论的指导下，以抽象的成果为工具来完成各种设计工作。在设计（实践）工作中，又将遇到很多新的问题，从而又促使人们在新的起点上实现认识过程的新飞跃。

3.5.2 计算学科中有关 3 个学科形态内在联系的论述

1. 3 个学科形态的内在联系

《计算作为一门学科》报告的实质是学科方法论的思想，其关键问题是抽象、理论和设计 3 个过程相互作用的问题。

（1）抽象源于现实世界，它的研究内容表现在两个方面：一方面是建立对客观事物进行抽象描述的方法；另一方面是要采用现有的抽象方法，建立具体问题的概念模型，从而实现对客观世界的感性认识。

（2）理论源于数学，它的研究内容也表现在两个方面：一方面是建立完整的理论体系；另一方面是在现有理论的指导下，建立具体问题的数学模型，从而实现对客观世界的理性认识。

（3）设计源于工程，它的研究内容同抽象、理论一样，也表现在两个方面：一方面是在对客观世界的感性认识和理性认识的基础上，完成一个具体的任务；另一方面是要对工程设计中所遇到的问题进行总结，提出问题，由理论界去解决它。同时，也要将工程设计中所积累的经验和教训进行总结，最后形成方法（如计算机组成结构的设计方法：冯 • 诺依曼型计算机）以便以后的工程设计。

抽象、理论和设计 3 个学科形态的划分，有助于我们正确地理解学科 3 个形态的地位和作用。在计算学科中，人们还完全可以从抽象、理论和设计 3 个形态出发独立地开展工作，这种工作方式可以使研究人员将精力集中在所关心的学科形态中（如计算机科学侧重理论和抽象形态，计算机工程侧重设计和抽象形态），从而促进计算理论研究的深入和计算技术的发展。

2. 《计算作为一门学科》报告关于 3 个学科形态的论述

（1）抽象（建模）是自然科学的根本。科学家们认为，科学的进展过程主要是通过形成假说，然后系统地按照建模过程对假说进行验证和确认取得的。

（2）理论是数学的根本。应用数学家们认为，科学的进展都是建立在数学基础之上。

（3）设计是工程的根本。工程师们认为，工程的进展主要是通过提出问题，并系统地按照设计过程，通过建立模型而加以解决的。

3. 《计算作为一门学科》报告中有关 3 个学科形态局限性的论述

《计算作为一门学科》报告对 3 个学科形态的内在联系作了如下论述：

许多有关数学、科学和工程相对优劣的争论都隐含地基于抽象、理论和设计 3 个过程中某一个更为根本的假设。而更详细的研究揭示，在计算学科中，“3 个过程”是错综复杂地缠绕在一起的，以至于把任何一个作为根本都是不合理的……

以上的论述主要着眼于认识的局部过程以及学科发展中的细节问题，而忽视了认识的一般过程和总体过程，以及“实践是认识的基础，人们的认识归根结底产生于人类的社会实践之中”的科学思维方法。

正是《计算作为一门学科》报告对计算机科学与技术方法论中惟一原始命题认识的局限性，使得专家们忽视了 3 个过程的内在联系，从而削弱了人们对报告本质的理解，以致 CC2001 任务组也不得不承认，与《计算作为一门学科》报告密切相关的 CC1991 教学计划的执行，并没有达到预期的效果。

3.5.3 关系数据库领域中 3 个学科形态内在联系的有关内容

1. “学生选课”例子 3 个学科形态的内在联系

就“学生选课”例子而言，根据计算机科学与技术方法论的形式化定义，其 3 个学科形态的内在联系可以用函数的形式来表示：

① *F*(学号)=学号；

② *F*(“学生选课”应用软件)=“学生选课”应用软件；

③ *F*(“学生选课”应用软件)=“学生选课”E-R 图。“学生选课”应用软件属于设计形态方面的内容，“学生选课”E-R 图属于抽象形态方面的内容。在“学生选课”应用软件的设计中，首先要对问题有感性认识，即抽象（如“学生选课”E-R 图、“学生选课”关系模型等），故两者有内在联系；

④ *F*(“学生选课”应用软件)=数据依赖理论。数据依赖理论属于理论形态方面的内容。正如前面分析时所指出的那样，当增加某些字段时，例子中的关系模型将出现问题，解决例子中的这些问题应在数据依赖理论的指导下进行，故两者有内在联系；

⑤ *F*(关系代数)= 关系模型。作为理论形态的“关系代数”是建立在关系模型的基础上进行研究的，故两者有内在联系。

……

2. 关系数据库领域中 3 个学科形态的内在联系

关系模型是数据库中最常用的数据模型，“学生选课”例子采用的数据模型就是关系模型，关系数据库是数据库领域中最基本的内容，下面对该领域 3 个学科形态内在联系的有关内容作简要分析。

进入 20 世纪 90 年代以来，数据库系统在传统关系数据库的基础上广泛地采用了 C/S 模式，这种模式使数据和应用程序分别存放在网络环境中的服务器和客户机上。客户端的客户程序通过 ODBC 与服务器上的数据库相连，这种模式能在传统关系数据库模式不变的情况下，在客户端采用面向对象思想（一种抽象方法）编制应用程序，这类系统叫对象—关系数据库系统，它弥补了传统关系数据库不能用面向对象方法描述客观世界的局限性，为数据库应用系统的开发创造了更好的条件，实现了认识过程的一次飞跃。

然而，在实际应用中，还存在一系列的问题，如：当前面向对象建模语言种类繁多、内容繁杂、难学难教，从而要求使用者的素质较高；另外，面向对象建模语言的语言体系结构、语义等方面还存在理论上的缺陷。这就要求人们在现有基础上去创建更加完善的抽象工具和理论体系，另一方面，需要人们充分利用现有的抽象工具抽象现实世界的本质特性，并在现有理论的指导下，更好地完成工程设计的要求。

显然，数据库中的抽象、理论和设计 3 个过程的互为作用，推动了数据库领域的发展。在数据库领域中，当一个新的抽象工具或理论体系产生后，科学抽象或科学理论的成果取得了质的变化的时候，数据库技术的研究又将在另一个新的起点上实现 3 个过程的相互作用，并以螺旋式上升的方式推动整个数据库领域的发展。

3.6 各主领域中3个学科形态的主要内容

《计算作为一门学科》报告给出了最初划分的 9 个主领域中的抽象、理论和设计 3 个学科形态的主要内容，本书在此基础上，进一步给出 CC2001 报告划分的 14 个主领域中有关抽象、理论和设计 3 个学科形态的主要内容。

1. 离散结构

该主领域包括集合论、数理逻辑、近世代数、图论和组合数学等主要内容，它属于学科理论形态方面的内容。同时，它又具有广泛的应用价值，为计算学科各分支领域基本问题（或具体问题）的感性认识（抽象）和理性认识（理论）提供强有力的数学工具。

2. 程序设计基础

该主领域主要包括程序设计结构、算法和问题求解、数据结构等内容，它考虑的是如何对问题进行抽象。它属于学科抽象形态方面的内容，并为计算学科各分支领域基本问题的感性认识（抽象）提供方法。

3. 社会和职业的问题

该主领域属于学科设计形态方面的内容。根据一般科学技术方法论的划分，该领域中的价值观、道德观属于设计形态中技术评估方面的内容，知识产权属于设计形态中技术保护方面的内容，而 CC1991 报告提到的美学问题则属于设计形态中技术美学方面的内容。

4. 算法与复杂性

（1）抽象形态的主要内容：包括算法分析、算法策略（如蛮干算法、贪婪算法、启发式算法、分治法等）、并行和分布式算法等。

（2）理论形态的主要内容：包括可计算性理论、计算复杂性理论、*P* 和 *NP* 类问题、并行计算理论、密码学等。

（3）设计形态的主要内容：包括对重要问题类的算法的选择、实现和测试、对通用算法的实现和测试（如哈希法、图和树的实现与测试）、对并行和分布式算法的实现和测试、对组合问题启发式算法的大量实验测试、密码协议等。

5. 体系结构

（1）抽象形态的主要内容：包括布尔代数模型、基本组件合成系统的通用方法、电路模型和在有限领域内计算算术函数的有限状态机、数据路径和控制结构模型、不同的模型和工作负载的优化指令集、硬件可靠性（如冗余、错误检测、恢复与测试）、VLSI 装置设计中的空间、时间和组织的折衷，不同的计算模型的机器组织（如时序的、数据流、表处理、阵列处理、向量处理和报文传递）、分级设计的确定，即系统级、程序级、指令级、寄存器级和门级等。

（2）理论形态的主要内容：包括布尔代数、开关理论、编码理论、有限自动机理论等。

（3）设计形态的主要内容：包括快速计算的硬件单元（如算术功能单元、高速缓冲存储器）、冯•诺依曼机（单指令顺序存储程序式计算机）、RISC 和 CISC 的实现、存储和记录信息，以及检测与纠正错误的有效方法、对差错处理的具体方法（如恢复、诊断、重构和备份过程）、为 VLSI 电路设计的计算机辅助设计（CAD）系统和逻辑模拟、故障诊断、硅编译器等、在不同计算模型上的机器实现（如数据流、树、LISP、超立方结构 、向量和多处理器）、超级计算机等。

6. 操作系统

（1）抽象形态的主要内容：包括不考虑物理细节（如面向进程而不是处理器，面向

文件而不是磁盘）而对同一类资源上进行操作的抽象原则、用户接口可以察觉的对象与内部计算机结构的绑定（Binding）、重要的子问题模型（如进程管理、内存管理、作业调度、两级存储管理和性能分析）、安全计算模型（如访问控制和验证）等。

（2）理论形态的主要内容：包括并发理论、调度理论（特别是处理机调度）、程序行为和存储管理的理论（如存储分配的优化策略）、性能模型化与分析等。

（3）设计形态的主要内容：包括分时系统、自动存储分配器、多级调度器、内存管理器、分层文件系统和其他作为商业系统基础的重要系统组件、构建操作系统（如 UNIX、DOS、Windows）的技术、建立实用程序库的技术（如编辑器、文件形式程序、编译器、连接器和设备驱动器）、文件和文件系统等内容。

7. 网络计算

（1）抽象形态的主要内容：包括分布式计算模型（如 C/S 模式，合作时序进程、消息传递和远方过程调用）、组网（分层协议，命名、远程资源利用、帮助服务和局域网协议）、网络安全模型（如通信、访问控制和验证）等。

（2）理论形态的主要内容：包括数据通信理论、排队理论、密码学、协议的形式化验证等。

（3）设计形态的主要内容：包括排队网络建模和实际系统性能评估的模拟程序包、网络体系结构（如以太网、FDDI、令牌网）、包含在 TCP/IP 中的协议技术、虚拟电路协议、Internet、实时会议等。

8. 程序设计语言

（1）抽象形态的主要内容：包括基于语法和动态语义模型的语言分类（如静态型、动态型、函数式、过程式、面向对象的、逻辑、规格说明、报文传递和数据流），按照目标应用领域的语言分类（如商业数据处理、仿真、表处理和图形），程序结构的主要语法和语义模型的分类（如过程分层、函数合成、抽象数据类型和通信的并行处理），语言的每一种主要类型的抽象实现模型、词法分析、编译、解释和代码优化的方法，词法分析器、扫描器、编译器组件和编译器的自动生成方法等。

（2）理论形态的主要内容：包括形式语言和自动机、图灵机（过程式语言的基础）、POST 系统（字符串处理语言的基础）、λ-演算（函数式语言的基础）、形式语义学、谓词逻辑、时态逻辑、近世代数等。

（3）设计形态的主要内容：包括把一个特殊的抽象机器（语法）和语义结合在一起形成的统一的可实现的整体特定语言（如过程式的（COBOL，FORTURN，ALGOL，Pascal，Ada，C），函数式的（LISP），数据流的（SISAL，VAL），面向对象的（Smalltalk，CLU，C++），逻辑的（Prolog），字符串（SNOBOL）和并发（CSP，Concurrent Pascal，Modula 2））、特定类型语言的指定实现方法，程序设计环境、词法分析器和扫描器的产生器（如 YACC，LEX）、编译器产生器、语法和语义检查、成型、调试和追踪程序、程序设计语言方法在文件处理方面的应用（如制表、图、化学公式）、统计处理等。

9. 人机交互

（1）抽象形态的主要内容：包括人的表现模型（如理解、运动、认知、文件、通信和组织）、原型化、交互对象的描述、人机通信（含减少人为错误和提高人的生产力的交互模式心理学研究）等。

（2）理论形态的主要内容：包括认知心理学、社会交互科学等。

（3）设计形态的主要内容：交互设备（如键盘、语音识别器）、有关人机交互的常用子程序库、图形专用语言、原形工具、用户接口的主要形式（如子程序库、专用语言和交互命令）、交互技术（如选择、定位、定向、拖动等技术）、图形拾取技术、以“人为中心”的人机交互软件的评价标准等。

10. 图形学和可视化计算

（1）抽象形态的主要内容：包括显示图像的算法、计算机辅助设计（CAD）模型、实体对象的计算机表示、图像处理和加强的方法。

（2）理论形态的主要内容：包括二维和高维几何（包括解析、投影、仿射和计算几何）、颜色理论、认知心理学、傅立叶分析、线性代数、图论等。

（3）设计形态的主要内容：包括不同的图形设备上图形算法的实现、不断增多的模型和现象的实验性图形算法的设计与实现、在显示中彩色图的恰当使用、在显示器和硬拷贝设备上彩色的精确再现、图形标准、图形语言和特殊的图形包、不同用户接口技术的实现（含位图设备上的直接操作和字符设备的屏幕技术）、用于不同的系统和机器之间信息转换的各种标准文件互换格式的实现、CAD 系统、图像增强系统等。

11. 人工智能

（1）抽象形态的主要内容：包括知识表示（如规则、框架和逻辑）以及处理知识的方法（如演绎、推理）、自然语言理解和自然语言表示的模型（包括音素表示和机器翻译）、语音识别与合成、从文本到语音的翻译、推理与学习模型（如不确定、非单调逻辑、Bayesian 推理）、启发式搜索方法、分支界限法、控制搜索、模仿生物系统的机器体系结构（如神经网络）、人类的记忆模型以及自动学习和机器人系统的其他元素等。

（2）理论形态的主要内容：包括逻辑（如单调、非单调和模糊逻辑）、概念依赖性、认知、自然语言理解的语法和语义模型，机器人动作和机器人使用的外部世界模型的运动学和力学原理，以及相关支持领域（如结构力学、图论、形式语法、语言学、哲学与心理学）等。

（3）设计形态的主要内容：包括逻辑程序设计软件系统的设计技巧、定理证明、规则评估，在小范围领域中使用专家系统的技术，专家系统外壳程序、逻辑程序设计的实现（如 PROLOG）、自然语言理解系统、神经网络的实现、国际象棋和其他策略性游戏的程序、语音合成器、识别器、机器人等。

12. 信息系统

（1）抽象形态的主要内容：包括表示数据的逻辑结构和数据元素之间关系的模型（如 E-R 模型、关系模型、面向对象的模型），为快速检索的文件表示（如索引），保证更新时数据库完整性（一致性）的方法，防止非授权泄露或更改数据的方法，对不同类信息检索系统和数据库（如超文本、文本、空间的、图像、规则集）进行查询的语言，允许文档在多个层次上包含文本、视频、图像和声音的模型（如超文本），人的因素和接口问题等。

（2）理论形态的主要内容：包括关系代数、关系演算、数据依赖理论、并发理论、统计推理、排序与搜索、性能分析以及支持理论的密码学。

（3）设计形态的主要内容：包括关系、层次、网络、分布式和并行数据库的设计技

术，信息检索系统的设计技术，安全数据库系统的设计技术，超文本系统的设计技术，把大型数据库映射到磁盘存储器的技术，把大型的只读数据库映射到光存储介质上的技术等。

13. 软件工程

（1）抽象形态的主要内容：包括规约方法（如谓词转换器、程序设计演算、抽象数据类型和 Floyd-Hoare 公理化思想）、方法学（如逐步求精法、模块化设计）、程序开发自动化方法（如文本编辑器、面向语法的编辑器和屏幕编辑器）、可靠计算的方法学（如容错、安全、可靠性、恢复、多路冗余）、软件工具与程序设计环境、程序和系统的测度与评价、软件系统到特定机器的相匹配问题域、软件研制的生命周期模型等。

（2）理论形态的主要内容：包括程序验证与证明、时态逻辑、可靠性理论以及支持领域：谓词演算、公理语义学和认知心理学等。

（3）设计形态的主要内容：包括归约语言，配置管理系统，版本修改系统，面向语法的编辑器，行编辑器，屏幕编辑器和字处理系统，实际使用并受到支持的特定软件开发方法（如 HDM、Dijkstra、Jockson、Mills 和 Yourdon 倡导的方法），测试的过程与实践（如遍历、手工仿真、模块间接口的检查），质量保证与工程管理，程序开发和调试、成型、文本格式化和数据库操作的软件工具，安全计算系统的标准等级与确认过程的描述，用户接口设计，可靠、容错的大型系统的设计方法，“以公众利益为中心”的软件从业人员认证体系。

14. 科学计算

（1）抽象形态的主要内容：包括物理问题的数学模型（连续或离散）的形式化表示，连续问题的离散化技术，有限元模型等。

（2）理论形态的主要内容：数论、线性代数、数值分析，以及支持领域，包括微积分、实数分析、复数分析和代数等。

（3）设计形态的主要内容：用于线性代数的函数库与函数包、常微分方程、统计、非线性方程和优化的函数库与函数包、把有限元算法映射到特定结构上的方法等。

3.7 计算机语言的发展及其3个学科形态的内在联系

计算机语言在计算学科中占有特殊的地位，它是计算学科中最富有智慧的成果之一，它深刻地影响着计算学科各个领域的发展。不仅如此，计算机语言还是程序员与计算机交流的主要工具。因此，可以说如果不了解计算机语言，就谈不上对计算学科的真正了解。

下面，我们从自然语言与形式化语言、图灵机和冯·诺依曼型计算机、机器指令与汇编语言、计算机的层次结构、虚拟机的意义和作用、高级语言、应用语言和自然语言的形式化问题等方面，介绍计算机语言的发展历程及其在抽象、理论和设计 3 个学科形态取得的主要成果，从而揭示计算机语言发展过程中 3 个学科形态的内在联系。

3.7.1 自然语言与形式语言

科学思维是通过可感知的语言（符号、文字等）来完善并得以显示的。否则，人们

将无法使自己的思想清晰化，更无法进行交流和沟通。

1. 自然语言的定义

人类的语言（文字）是人类最普遍使用的符号系统。其最基本、最普遍的形式是自然语言符号系统。自然语言是某一社会发展中形成的一种民族语言。例如，汉语、英语、法语和俄语等。

2. 自然语言符号系统的基本特征

（1）歧义性；

（2）不够严格和不够统一的语法结构。

下面，我们用语言学家吕淑湘先生给出的两个例子来说明自然语言的歧义性问题。

例 3.2　他的发理得好。

这个例子至少有两种不同的解释：

① 他的理发水平高；

② 理发师理他的发理得好。

例 3.3　他的小说看不完。

这个例子至少有 3 种不同的解释：

① 他写的小说看不完；

② 他收藏的小说看不完；

③ 他是个小说迷。

3. 高级语言的歧义性问题

自然语言的语义有歧义性的问题，高级程序设计语言其实也有语义的歧义性问题，下面，我们给出一个典型的关于语义问题的例子。

例 3.4　IF (表达式 1) THEN IF (表达式 2) THEN 语句 1 ELSE 语句 2。

这个例子至少有两种不同的解释：

① IF (表达式 1) THEN (IF (表达式 2) THEN 语句 1 ELSE 语句 2);

② IF (表达式 1) THEN (IF (表达式 2) THEN 语句 1) ELSE 语句 2。

显然，自然语言和高级程序设计语言都存在歧义性的问题，只不过，高级程序设计语言存在较少的歧义性而已，而要用计算机对语言进行处理，则必须解决语言的歧义性问题，否则，计算机就无法进行判定。

4. 形式语言

随着科学的发展，人们在自然语言符号系统的基础上，逐步建立起了人工语言符号系统（也称科学语言系统），即各学科的专门科学术语（符号），使语言符号保持其单一性、无歧义性和明确性。

人工语言符号系统发展的第二阶段叫形式化语言，简称形式语言。形式语言是进行形式化工作的元语言，它是以数学和数理逻辑为基础的科学语言。

5. 形式语言的基本特点

（1）有一组初始的、专门的符号集；

（2）有一组精确定义的，由初始的、专门的符号组成的符号串转换成另一个符号串的规则。在形式语言中，不允许出现根据形成规则无法确定的符号串。

6. 形式语言的语法

形式语言中的转换规则被称为形式语言的语法。语法不包含语义，它们是两个完全不同的概念。在一个给定的形式语言中，可以根据需要，通过赋值或模型对其进行严格的语义解释，从而构成形式语言的语义。在形式语言中，语法和语义要作严格的区分。下面，给出几个例子，以加深读者对形式语言的理解。

例 3.5　语言 W 定义为：

初始符号集：{*a*，*b*，*c*，*d*，*e*}。形成规则：上述符号组成的有限符号串中，能组成一英语单词的为一公式；否则不是。

问：W 是否为一形式语言？

答：不是。因为，根据形成规则，无法精确地定义转换规则。原因：形成规则（语法）中包含了语义。

例 3.6　语言 X 定义为：

初始符号集：{*a*，*b*，*c*，*d*，*e*，(，)，+，−，×，÷}。形成规则：上述符号组成的有限符号串中，构成表达式的为一公式，否则不是。

问：X 是否为一形式语言？

答：不是。原因：与例 3.5 相同。

例 3.7　语言 Y 定义为：

初始符号集：{*a*，*b*，*c*，*d*，*e*，(，)，+，−，×，÷}。形成规则：上述符号组成的有限符号串中，凡以符号“(”开头且以“)”结尾的符号串，为一公式。

问：Y 是否为一形式语言？

答：不是。因为，根据形成规则，无法对不是以符号“(”开头且以“)”结尾的符号串进行判定。例如，(*a*+*b*)×*c*。

例 3.8　语言 Z 定义为：

初始符号集：{*a*，*b*，*c*，*d*，*e*，(，)，+，−，×，÷}。形成规则：上述符号组成的有限符号串中，凡以符号“(”开头且以“)”结尾的符号串，为一公式，否则不是。

问：Z 是否为一形式语言？

答：是。

由于技术科学（计算学科主要是一门技术科学）的语言从类型上说基本上是描述性、断定性而非评论性的，在描述性语言中又以分析陈述为主。这样，技术科学就更有可能充分运用形式语言来表达自己深刻而复杂的内容，并进行演算化的推理。

计算机语言是一种形式化语言。讲到计算机语言，就不可避免地要涉及到支撑语言的计算机，而计算机的诞生又与形式化研究的进程息息相关。其实，不论是计算机语言还是数字计算机，它们都是形式化的产物。

3.7.2　图灵机与冯・诺依曼型计算机

在关于形式化研究进程方面，第 2 章已给出“希尔伯特纲领”以及哥德尔关于形式系统的“不完备性定理”，正是在哥德尔的“不完备性定理”的影响下，图灵通过对人的计算过程的哲学分析，描述了计算一个数的过程，得出后来以他名字命名的通用计算机概念。由于图灵对计算科学所作出的杰出贡献，ACM 于 1966 年设立了以图灵名字命名的计算机科学大奖——图灵奖，以纪念这位杰出的科学家。后人也将图灵誉为计算机科学之父。

1. 图灵机

（1）图灵机及其他计算模型

根据图灵的观点可以得到这样的结论：凡是能用算法方法解决的问题，也一定能用图灵机解决；凡是图灵机解决不了的问题，任何算法也解决不了。

今天我们知道，图灵机与当时提出的用于解决可计算问题的递归函数、λ-演算和POST 规范系统等计算模型在计算能力上是等价的。它们于 20 世纪 30 年代共同奠定了计算科学的理论基础。相比于其他几种计算模型，图灵机是从过程这一角度来刻画计算的本质，其结构简单、操作运算规则也较少，从而为更多的人所理解。

（2）图灵机的特征

① 图灵机由一条两端可无限延长的带子、一个读写头以及一组控制读写头工作的命令组成，如图 3.2 所示。图灵机的带子被划分为一系列均匀的方格。读写头可以沿带子方向左右移动，并可以在每个方格上进行读写。

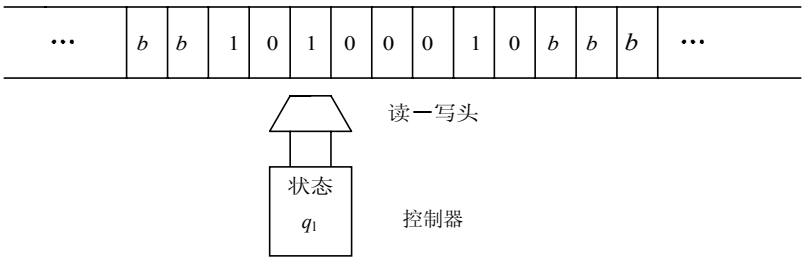


图 3.2　图灵机

② 写在带子上的符号为一个有穷字母表：{S_0，S_1，S_2，…，S_p}。通常，可以认为这个有穷字母表仅有 S_0、S_1 两个字符，其中 S_0 可以看作是“0”，S_1 可以看作是“1”，它们只是两个符号，要说有意义的话，也只有形式的意义。众所周知，由字符“0”和“1”组成的字母表可以表示任何一个数。由于“0”和“1”只有形式的意义，因此，也可以将 S_0 改称为“白”，S_1 改称为“黑”，甚至，还可以改称为“桌子”和“老虎”，这样改称的目的在于割断与直感的联系，并加深对布尔域中的值{真，假}，以及二进制机器本质的理解。

③ 机器的控制状态表为：{q_1，q_2，…，q_m}。通常，将一个图灵机的初始状态设为 q_1，在每一个具体的图灵机中还要确定一个结束状态 q_w。

一个给定机器的“程序”认为是机器内的五元组（$q_iS_jS_kR$（或 L 或 N）q_1）形式的指令集，五元组定义了机器在一个特定状态下读入一个特定字符时所采取的动作。5 个元素的含义如下：

- q_i 表示机器目前所处的状态；
- S_j 表示机器从方格中读入的符号；
- S_k 表示机器用来代替 S_j 写入方格中的符号；
- R、L、N 分别表示向右移一格、向左移一格、不移动；
- q_1 表示下一步机器的状态。

（3）图灵机的工作原理

机器从给定带子上的某起始点出发，其动作完全由其初始状态及机内五元组来决定。就某种意义而言，一个机器其实就是它作用于纸带上的五元组集。

一个机器计算的结果是从机器停止时带子上的信息得到的。容易看出，$q_1S_2S_2Rq_3$ 指令和 $q_3S_3S_3Lq_1$ 指令如果同时出现在机器中，当机器处于状态 q_1，第一条指令读入的是 S_2，第二条指令读入的是 S_3，那么机器会在两个方块之间无休止地工作。

另外，如果 $q_3S_2S_2Rq_4$ 和 $q_3S_2S_4Lq_6$ 指令同时出现在机器中，当机器处于状态 q_3 并在带子上扫描到符号 S_2 时，就产生了二义性的问题，机器就无法判定。

以上两个问题是进行程序设计时要注意避免的问题。

（4）实例

例 3.9　设 b 表示空格，q_1 表示机器的初始状态，q_4 表示机器的结束状态，如果带子上的输入信息是 10100010，读入头位对准最右边第一个为 0 的方格，状态为初始状态 q_1。按照以下规则执行之后，输出正确的计算结果。

$$q_1\ 0\ 1\ L\ q_2$$
$$q_1\ 1\ 0\ L\ q_3$$
$$q_1\ b\ b\ N\ q_4$$
$$q_2\ 0\ 0\ L\ q_2$$
$$q_2\ 1\ 1\ L\ q_2$$
$$q_2\ b\ b\ N\ q_4$$
$$q_3\ 0\ 1\ L\ q_2$$
$$q_3\ 1\ 0\ L\ q_3$$
$$q_3\ b\ b\ N\ q_4$$

显然，最后的结果是 10100011，即对给定的数加 1。其实，以上命令计算的是这样一个函数：$S(x)=x+1$。当没有输入时，即初始状态所指的方格为空格（b）时，不改变空格符，读写头不动并停机。

现在，尽管我们仅给出了图灵机的非形式化描述，但也不难理解图灵机作为一个数学机器的事实。

（5）小结

图灵机不仅可以计算 $S(x)=x+1$（后继函数），显然还可以计算 $N(x)=0$（零函数），甚至 $U_i^{(n)}(x_1, x_2, \cdots, x_n)=x_i$，$1\leqslant i\leqslant n$（投影函数）以及这 3 个函数的任意组合。从递归论中，我们知道这 3 个函数属于初始递归函数，而任何原始递归函数都是从这 3 个初始递归函数经有限次的复合、递归和极小化操作得到的。从可计算理论中，我们知道每一个原始递归函数都是图灵机可计算的。

尽管图灵机就其计算能力而言，可以模拟现代任何计算机，甚至，图灵机还蕴含了现代存储程序式计算机的思想（图灵机的带子可以看作是具有可擦写功能的存储器）。但是，它毕竟不同于实际的计算机，在实际计算机的研制中，还需要有具体的实现方法与实现技术。

2. 冯·诺依曼型计算机

由于应用的需求以及电子技术的发展，1946 年 2 月 14 日，世界上第一台数字电子计算机 ENIAC 在美国宾夕法尼亚大学研制成功。

ENIAC 是第一台使用电子线路来执行算术和逻辑运算以及信息存储的真正工作的计算机器，它的成功研制显示了电子线路的巨大优越性。但是，ENIAC 的结构在很大程度上是依照机电系统设计的，还存在重大的线路结构等问题。在图灵等人工作的影响

下，1946 年 6 月，美国杰出的数学家冯 • 诺依曼（Von Neumann）及其同事完成了关于“电子计算装置逻辑结构设计”的研究报告，具体介绍了制造电子计算机和程序设计的新思想，给出了由控制器、运算器、存储器、输入和输出设备 5 类部件组成的，被称为冯 • 诺依曼型计算机（或存储程序式计算机）的组织结构（如图 3.3 所示），以及实现它们的方法，为现代计算机的研制奠定了基础。至今为止，大多数计算机采用的仍然是冯 • 诺依曼型计算机的组织结构，只是作了一些改进而已。因此，冯 • 诺依曼被人们誉为“计算机器之父”。

（1）输入设备和输出设备

输入和输出设备是人与计算机进行交互的两大部件，其作用分别是将信息输入计算机和输出计算机。常用的文字输入设备是键盘，当在键盘上按下一个键时，按下的键通过编码变换成机器可读的数据形式，如字符“A”变换成 ASCII 码“1000001”，该编码数据随即存入存储器等待处理，同时，通过与“1000001”对应的字符点阵数据在屏幕上显示一个字符“A”。此外，还有扫描仪、穿孔卡片读入机和鼠标等专用输入设备。

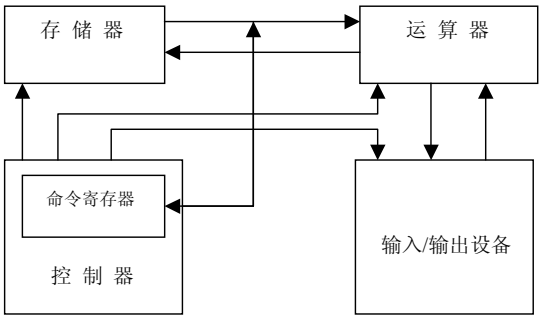


图 3.3　冯 • 诺依曼型计算机的组织结构

输出设备通过数字、字符、图形、图像、声音等形式将计算结果输出。常用的输出设备有打印机、显示器、绘图仪、磁记录设备等。

（2）存储器

存储器是一种数据或信息的存储部件，它分成很多存储单元，并按照一定的方式进行排列。每个单元都编了号，称为存储地址。指令和数据存放在存储器中,而且对指令和数据同等对待，都不加区别地送到运算器中运算。指令在存储器中基本上是按执行顺序存储的，由指令计数器指明要执行的指令在存储器中的地址。

早期的存储器采用水银延迟线、磁芯等材料研制，现在基本上采用半导体材料研制。存储器一般分为内存储器（简称内存）与外存储器（简称外存）两大类。内存一般安装在主机板上，根据材料和工作原理的不同，内存可分为随机存储器（RAM）和只读存储器（ROM）两种。前者可以随时读写信息，关机后信息消失，后者存储系统的固有程序和数据，其信息一般作为引导系统的一部分，信息只能读不能写，关机后信息不消失。控制器和运算器只能接受在内存中存放的指令和数据。

外存一般安装在主机板之外，例如磁盘就是一种常用的外存。外存上面的信息可长久保存，但这些信息必须读入内存之后才能被控制器和运算器所利用。磁盘按其材料的不同，又可分为软盘和硬盘两种。

（3）运算器和控制器

运算器又称为算术逻辑单元（ALU），它由很多逻辑电路组成。当控制器把数据带

给 ALU 后，它能根据指令完成算术运算或逻辑运算。

控制器由时序电路和逻辑电路组成，它的任务是负责从存储器中取出指令，确定指令的类型并对指令进行译码，控制整个计算机系统一步一步地完成各种操作。

随着技术的发展，运算器和控制器现在一般都做在一个集成块中，合称为中央处理机（CPU）。那么计算机中的各种控制和运算便都由 CPU 来完成，因此，人们把 CPU 称为计算机的心脏。

3.7.3 机器指令与汇编语言

1. 机器指令

每台数字电子计算机在设计中，都规定了一组指令，这组机器指令集合，就是所谓的机器指令系统。用机器指令形式编写的程序，称为机器语言，支撑机器语言的理论基础是图灵机等计算模型。

2. 计算机语言在裸机级所取得的主要成果

在裸机级，计算机语言中的抽象、理论和设计 3 个形态的主要内容和成果如表 3.1 所示。

表 3.1　　裸机级计算机语言中有关抽象、理论和设计形态的主要内容

计算机语言	抽象	理论	设计
裸机级的主要内容和成果	语言的符号集为：{0，1}； 用机器指令对算法进行描述	图灵机（过程语言的基础）、波斯特系统（字符串处理语言的基础）、λ-演算（函数式语言的基础）等计算模型	冯•诺依曼型计算机等实现技术； 数字电子计算机产品

表 3.1 所描述的是，在裸机级，计算机语言关于算法的描述采用的是实际机器的机器指令，它的符号集是{0，1}，即所有指令由“0”和“1”组成；支撑实际机器的理论是图灵机等计算模型；在图灵机等计算模型理论的指导下，有关设计形态的主要成果有冯•诺依曼型计算机等具体实现思想和技术，以及各类数字电子计算机产品。

3. CISC

在实际机器研制的过程中，同时也要对指令系统进行设计。为了使机器具有更强的功能、更好的性能价格比，人们对机器指令系统进行了研究：最初人们采用的是进一步增强原有指令的功能，并设置更为复杂的指令的方法，按照这种思路，机器指令系统将变得越来越庞杂，采用这种设计思路的计算机被称为复杂指令系统计算机（CISC）。CISC 的思路是由 IBM 公司提出的，并以 1964 年 IBM 研制的 IBM 360 系统为代表。

20 世纪 70 年代，通过仔细的研究，人们发现，80%的指令只在 20%的运行时间里用到；一些指令非常繁杂，而执行效率甚至比用几条简单的基本指令组合的实现还要慢。另外，庞杂的指令系统也给超大规模集成电路（VLSI）的设计带来了困难，它不但不利于设计自动化技术的应用，延长了设计周期，增加了成本，同时，也容易增加设计中出现错误的机会，从而降低了系统的可靠性。

4. RISC

为了解决以上问题，Patterson 等人提出了 RISC 的设计思路，这种设计思路主要是

通过减少指令总数和简化指令的功能来降低硬件设计的复杂度，从而提高指令的执行速度。按照这种思路，机器指令系统将得到进一步精简，采用这种设计思路的计算机被称为精简指令系统计算机（RISC）。RISC 技术现已成为计算机结构设计中的一种重要思想，与 CISC 技术相比，它有如下优点：

① 简化了指令系统，适合超大规模集成电路的实现；

② 提高了机器执行的速度和效率；

③ 降低了设计成本，提高了系统的可靠性；

④ 提供了直接支持高级语言的能力，简化了编译程序的设计。

5. 汇编语言

在计算机发展的早期，人们最初使用机器指令来编写程序。然而，由于以二进制表示的机器指令编写的程序很难阅读和理解，于是，在机器指令的基础上，人们提出了采用字符和十进制数来代替二进制代码的思想，产生了将机器指令符号化的汇编语言。

例 3.10　对 2+6 进行计算的算法描述

（1）用机器指令对“2+6”进行计算的算法描述：

1011000000000110
0000010000000010
10100010010100000000000

第一条指令表示将“6”送到寄存器 AL 中，数字“6”放在指令后八位；第二条指令表示数“2”与寄存器 AL 中的内容相加，结果仍存在 AL 中；第三条指令表示把 AL 中的内容送到地址为 5 的单元中。

以上是一个非常简单的例子，从例子中可以看到，机器指令由一连串的“0”和“1”组成，很难辨认。显然，要用机器指令进行程序设计是非常困难的。

（2）汇编语言对“2+6”进行计算的算法描述：

MOV AL，6
ADD AL，2
MOV VC，AL

汇编语言语句与特定的机器指令有一一对应的关系，但是它毕竟不同于由二进制组成的机器指令，它还需要经汇编程序翻译为机器指令后才能运行。汇编语言源程序经汇编程序翻译成机器指令，再在实际的机器中执行。这样，就汇编语言的用户而言，该机器是可以直接识别汇编语言的，从而产生了一个属于抽象形态的重要概念，即虚拟机的概念。

3.7.4　以虚拟机的观点来划分计算机的层次结构

1. 虚拟机

虚拟机（Virtual Machine，也被译为如真机）是一个抽象的计算机，它由软件实现，并与实际机器一样，都具有一个指令集并可以使用不同的存储区域。例如，一台机器上配有 C 语言和 Pascal 语言的编译程序，对 C 语言用户来说，这台机器就是以 C 语言为机器语言的虚拟机，对 Pascal 用户来说，这台机器就是以 Pascal 语言为机器语言的虚拟机。略有区别的是 Java 虚拟机，它不识别 Java 程序设计语言，它识别的是字节码（ByteCode，Java 源程序经 Java 编译器编译后产生，也称为 Java 虚拟机指令）。

2. 虚拟机的层次之分

虚拟机可分为固件虚拟机、操作系统虚拟机、汇编语言虚拟机、高级语言虚拟机和应用语言虚拟机等几个层次。下面，我们从语言的角度给出的计算机系统的层次结构图（如图 3.4 所示）。

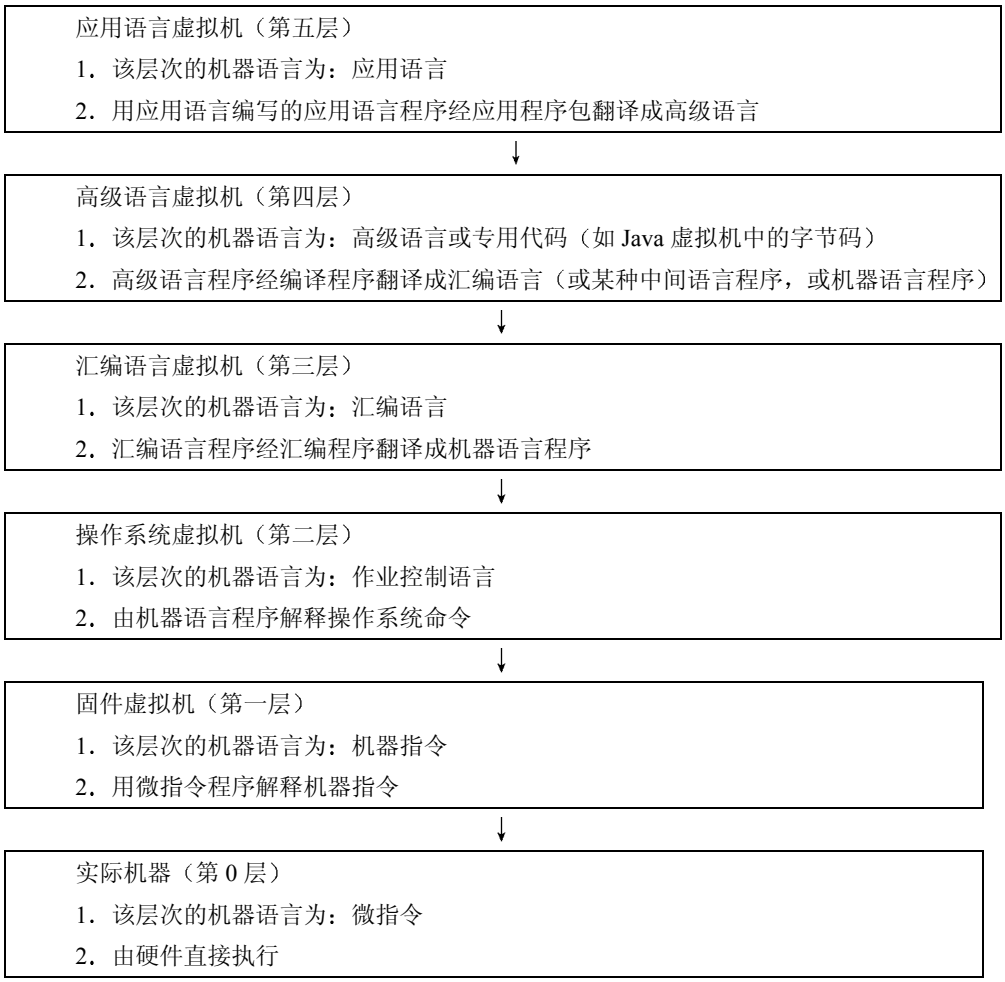


图 3.4　计算机系统的层次结构图

3. 虚拟机的意义和作用

当机器（实际机器或虚拟机）确定下来后，所识别的语言也随之确定；反之，当一种语言形式化后，所需要支撑的机器也可以确定下来。从计算机系统的层次结构图中可以清晰地看到这种机器与语言的关系。

虚拟机是计算学科中抽象的重要内容。引入虚拟机的概念，就计算机语言而言，有以下意义和作用：

（1）有助于我们正确理解各种语言的实质和实现途径

微指令、机器指令、作业控制语言主要是为支撑更高层次虚拟机所必须的解释程序和翻译程序而设计的，它们是更高层次虚拟机设计与实现的基础。汇编语言、高级语言、应用语言主要是为应用程序员设计的，它们需通过翻译变成低级语言，或由低级语言解释来执行。为了对上一层次语言进行较为方便的翻译和解释，相邻层次语言的语义差距不能太大。虚拟机的引入，有助于我们正确理解各种语言的实质和实现途径，从而更好

地进行语言的研究和应用。

（2）推动了计算机体系结构以及计算机语言的发展

虚拟机的引入使计算机体系结构得到了极大的发展，由于各层次虚拟机均可以识别相应层次的计算机语言，从而摆脱了这些语言必须在同一台实际机器上执行的状况，为多处理计算机系统、分布式处理系统以及计算机网络、并行计算机系统等新的计算机体系结构的出现奠定了基础。

计算机体系结构发展的同时，也极大地促进了计算机语言的发展，相继出现了一系列支持多处理机、分布式、计算机网络、并行计算机的高级语言。例如，Java 语言，它经 Java 编译器编译产生的字节码可以在网上任何一台装有 Java 虚拟机的机器上，由该虚拟机解释执行。值得注意的是，由于支持网络的高级语言可以在网上任何一台装配有该语言虚拟机的机器上运行，从而给网络安全带来了巨大的隐患。

反过来，我们也可以根据需要设计支持特定语言的虚拟机或实际机器。在计算机中，软件和硬件在逻辑功能上是等效的。一般来说，软件实现的功能可以由硬件来完成，硬件实现的功能也可以由软件来模拟完成。用软件还是硬件来实现一个逻辑功能，这要根据其实现的性能、价格和难易程度等情况进行折衷。随着技术的发展，硬件成本的不断下降，可以考虑用硬件和固件来实现各层次虚拟机的功能，甚至可以用真正存在的处理机来代替各个层次的虚拟机，例如，高级语言机器等，从而又促进了计算机体系结构的发展。

（3）有助于各层次计算机语言自身的完善

虚拟机的层次之分，有助于各层次计算机语言相对独立地发展，使研制者可以将注意力主要放在本层次语言上，使之不断地得到完善和发展。各种语言的不同升级版本就是这种不断完善的产物。

在图 3.4 中，有一个特殊的虚拟机，那就是操作系统虚拟机，它属于操作系统研究的范畴。虽然它作用于各个层次，但从本质上来看，它是固件虚拟机的引伸，因此，一般将它放在固件虚拟机之上、汇编语言虚拟机之下来认识。它提供了固件机器所没有的，但为汇编语言和高级语言使用和实现所需的某些基本操作和数据结构，如文件结构与文件管理的基本操作、存储体系以及多道程序和多重处理所用的某些操作等，这些操作可以看成是操作系统的指令系统，它们经机器语言程序解释实现。另外，需要指出的是，有些机器指令，如某些 I/O 操作指令是被操作系统“挡”住的，高级语言不能调用这些指令，但大部分机器指令，如运算类指令等，操作系统不作限制，并且这些指令基本包括在操作系统的指令系统之内。

在操作系统虚拟机的上一层是汇编语言虚拟机，由于汇编语言可以直接控制机器的所有操作，以及它与机器指令的一一对应关系所带来的高效性，因此，在一些软件的研制中，汇编语言和机器指令至今仍被人们所使用。

4. 汇编语言中有关抽象、理论和设计 3 个形态的主要内容（表 3.2）

表 3.2　　汇编语言中有关抽象、理论和设计形态的主要内容

抽象	理论	设计
常用的符号有：数字(0~9)，大小写字母(A~Z、a~z)等； 虚拟机； 用汇编语言对算法进行描述	与裸机级中理论形态的内容相同	CISC 设计思想； RISC 设计思想； 翻译方法和技术； 汇编程序

3.7.5 高级语言

1. 高级语言的产生

虽然与机器语言相比，汇编语言的产生是一个很大的进步，但是用它来进行程序设计仍然比较困难。于是人们着手对它进行改进。一是发展宏汇编，即用一条宏指令代替若干条汇编指令，从而提高编程效率。现在人们使用的汇编语言，大多数都是宏汇编语言；二是创建高级语言，使编程更加方便。如用高级语言对例子 2+6 进行计算的算法描述，其描述与数学描述一样，即 2+6。

在汇编语言虚拟机的上一层是高级语言虚拟机。高级语言的语句与特定机器的指令无关，又比较接近自然语言，因此，用高级语言进行程序设计就方便多了。

20 世纪 50 年代是高级语言兴起的年代，早期的有 Forturn、Algol、Cobol、Lisp 等高级语言，随着语言学理论研究的进展以及计算技术的迅猛发展，在原有基础上又产生了大量新的高级语言。

2. 高级语言的分类

按语言的特点，可以将高级语言划分为：过程式语言（如 Cobol，Forturn，Algol，Pascal，Ada，C）、函数式语言（如 Lisp）、数据流语言（如 SISAL，VAL）、面向对象语言（如 Smalltalk，CLU，C++）、逻辑语言（如 Prolog）、字符串语言（如 SNOBOL）和并发程序设计语言（如 Concurrent Pascal，Modula 2）等类型的语言。

3. 高级语言的形式化

计算机要处理高级语言，就必须使其形式化。20 世纪 50 年代，在高级语言发展的早期，计算机语言的设计往往强调其“方便”的一面，而较忽略其“严格”的一面，因而对语言的语义（注意：这里所指的“语义”与西尔勒在“中文屋子”中所说的“语义”有本质的不同，这里指的是计算机语言中一类特定的转换规则，若无特别说明，本书均指这一意义），甚至语法，都未下严格的定义，从而语言的设计者、语言的实现者和语言的使用者对同一语言的语义缺乏共同的理解，造成了一定程度的混乱。

20 世纪 50 年代，美国语言学家乔姆斯基（Noam Chomsky）关于语言分层的理论，以及巴科斯（Backus）、瑙尔（Naur）的关于“上下文无关方法表示形式”的研究成果推动了语法形式化的研究。其结果是，在 ALGOL60 的文本设计中第一次使用了 BNF 范式来表示语法，并且第一次在语言文本中明确提出应将语法和语义区分开来。20 世纪 50 年代至 60 年代间，面向语法的编译自动化理论得到了很大发展，使语法形式化研究的成果达到实用化的水平。

语法形式化问题基本解决以后，人们逐步把注意力集中到语义形式化的研究方面。20 世纪 60 年代，相继诞生了操作语义学、指称语义学、公理语义学、代数语义学等语义学理论，这些理论与乔姆斯基等人关于语法形式化的形式语言与自动机理论一起为高级语言的发展奠定了基础。

相对于汇编语言和机器语言，高级语言的数据类型的抽象层次有了很大地提高，出现了整型、实型、字符型、布尔型、用户自定义类型以及抽象数据类型等新的数据类型，极大地方便了用户对数据的抽象描述，为实现软件设计的工程化奠定了基础。

4. 高级语言中有关抽象、理论和设计 3 个形态的主要内容（表 3.3）

表 3.3　　高级语言中有关抽象、理论和设计形态的主要内容

抽象	理论	设计
常用的符号：数字(0~9)，大小写字母(A~Z、a~z)，括号，运算符(+，–，*，/)等; 用高级语言对算法进行的描述; 语言的分类方法; 各种数据类型的抽象实现模型; 词法分析、编译、解释和代码优化的方法; 词法分析器、扫描器、编译器组件和编译器的自动生成方法	形式语言和自动机理论; 形式语义学：操作、指称、公理、代数、并发和分布式程序的形式语义	特定语言：过程式的COBOL，FORTURN，ALGOL，Pascal，Ada，C），函数式的（LISP），数据流的（SISAL，VAL），面向对象的（Smalltalk，C++），逻辑的（Prolog），字符串（SNOBOL），和并发（Concurrent Pascal，Modula 2）等语言; 词法分析器和扫描器的产生器（如 YACC，LEX），编译器产生器; 语法和语义检查，成型、调试和追踪程序

3.7.6　应用语言

1. 应用语言

在高级语言虚拟机之上，还有应用语言虚拟机，它是为使计算机系统满足某种特定应用而专门设计的，如商业管理系统等。应用语言虚拟机的机器语言为应用语言。用应用语言编写的程序一般经应用程序包翻译成高级语言程序后，再逐级向下实现。

2. 第四代语言

在计算机界，根据计算机语言接近人类语言的程度，一般将它划分为 5 代：第一代为机器语言；第二代为汇编语言；第三代为高级语言；第四代为“非过程性语言”；第五代为自然语言。

本书中，我们将第四代语言划归到应用语言之中。第四代语言（4GL）是 20 世纪 80 年代随着大型管理信息系统开发的需要而产生的，这类语言提供了功能强大的非过程化问题定义手段，用户只需告知系统“做什么”，而无需说明“怎么做”，因此，极大地提高了软件的生产效率。

4GL 以数据库管理系统所提供的功能为核心，进一步构造了开发高层软件系统的开发环境，如报表生成、多窗口表格设计、菜单生成系统等，为用户提供了一个良好的应用开发环境。4GL 的代表性软件系统有：PowerBuilder、Delphi 和 INFORMOX-4GL 等。

3. 应用语言中有关抽象、理论和设计形态的主要内容（表 3.4）

表 3.4　　应用语言中有关抽象、理论和设计形态的主要内容

抽象	理论	设计
用应用语言对算法进行描述	特定应用领域的支撑理论：如数据库等领域的支撑理论	在文件处理等方面的应用：如表生成，图、数据处理，统计处理等; 第四代语言（4GL），如PowerBuilder、Delphi、

		Informox-4GL 等； 程序设计环境

3.7.7 自然语言

除了以上层次的计算机语言外，还有自然语言。自然语言的计算机处理是计算学科中最富有挑战性的课题之一。

1. 自然语言计算机处理层次的划分

自然语言的计算机处理可以分为以下四个层次：

（1）第一层次是文字和语音，即基本语言信息的构成；

（2）第二层次是语法，即语言的形态结构；

（3）第三层次是语义，即语言与它所指的对象之间的关系；

（4）第四层次是语用，即语言与它的使用者之间的关系。

2. 自然语言的输入问题

目前，自然语言的输入问题已基本解决，各种自然语言的文字（如英文、中文等）都可以通过多种方式，例如键盘（汉字的编码输入也是通过键盘进行输入）、扫描、手写、语音等方式进入计算机。计算机可以对输入的文字进行各种加工和处理（如放大、变形等），现在大多数的报纸、杂志、书籍等就是这一处理的产物。

3. 自然语言的形式化问题

文字输入计算机后，要使计算机对自然语言进行处理, 就必须使其形式化。因此，如何解决自然语言语法和语义的形式化问题，成为计算机处理自然语言的关键。

乔姆斯基用了一个寻常的，但不为人们所注意的事实回答了这个问题：一个说本族语的人具有一种理解他过去从未听到过的句子的能力，他也能十分贴切地说出大量新的句子，而说同一种语言的人对听懂这些句子是毫不困难的。这个事实表明：人不仅具有创造新句子的能力，而且还有创造“合格”句子的能力。

乔姆斯基把人所具有的创造和理解正确句子的能力称为语言的“创造性”（Creativity）。而语言“创造性”过程的本质，其实就是由有限数量的词根据一定的规则产生正确句子的过程，进一步而言，其实质也就是一个字符串到另一个字符串的变换过程。显然，语言“创造性”过程的本质与计算过程的本质是一致的，因此，可以将自然语言也看作是一种计算，从而自然语言能否实现形式化的争论也就不存在了。

4. 自然语言形式化的方法及实例

自然语言能否形式化的问题解决以后，接下来的问题是“如何使其形式化？”。自然语言形式化的内容非常丰富，本节仅给出一个简单的例子，以便大家理解。

现有一个具体的形式语法 $G_0=<V_n，V_t，P_o，S>$，

其中：

V_n 为非终结符号的有限集合；

V_t 为终结符号的有限集合；

P_o 为生成式（或称产生式）的有限集合，即形式规则；

S 为开始符号。

该形式语法的定义为：

V_n={S，NP，VP，N，V}

V_t={我，他，学，教，英语，汉语，希望}

P_o={S→NP VP，NP→N，VP→V NP，VP→V S，N→我，N→他，V→学，V→教，V→希望，N→英语，N→汉语}。

其中：

（1）S 表示句子；

（2）NP 表示名词短语；

（3）VP 表示动词短语；

（4）N 表示名词；

（5）V 表示动词；

（6）S→NP VP 表示：句子由名词短语和动词短语组成；

（7）NP→N 表示：名词短语由名词构成。

其他转换规则依此类推。根据以上形式语法的定义，可以产生以下一些“合格”的句子：

（1）我学英语；

（2）他学汉语；

（3）我教他学汉语；

（4）他教我学英语；

（5）我希望他教我学英语；

（6）他希望我教他学汉语；……。

下面，按照以上语法规则给出句子（6）的派生过程：

S							
NP	VP						根据 S→NP VP
N	VP						根据 NP→N
N	V	S					根据 VP→V S
N	V	NP	VP				根据 S→NP VP
N	V	N	V	S			根据 VP→V S
N	V	N	V	NP	VP		根据 S→NP VP
N	V	N	V	N	VP		根据 NP→N
N	V	N	V	N	V	NP	根据 VP→VNP
N	V	N	V	N	V	N	根据 NP→N
他	V	N	V	N	V	N	根据 N→他
他	希望	N	V	N	V	N	根据 V→希望
他	希望	我	V	N	V	N	根据 N→我
他	希望	我	教	N	V	N	根据 V→教
他	希望	我	教	他	V	N	根据 N→他
他	希望	我	教	他	学	N	根据 V→学
他	希望	我	教	他	学	汉语	根据 N→汉语

就此，可以理解，以上句子完全可以用上述形式化方式描述，并且以上形式语法还

可以构造出更多符合以上语法规定的句子。

通过以上例子，可以知道汉语是可以用数学模型（形式语法和形式语义均是数学模型）来表示的。

就目前而言，自然语言的语法形式化和语义形式化的研究，已取得了较为丰硕的成果，现在已有一些计算机程序能在受限制的领域内“懂得”英语等自然语言，比如 SQL（Structured English Query Language）语言可以根据数据库中的信息，按照其受到严格限制的英语语言的命令来回答问题或处理事务。但要实现更多领域的自然语言理解，不仅还有很多的工作要做，而且还面临巨大的挑战。例如，在自然语言中就有一类是关于人的常识（Commonsense）和常识性推理（Commonsense Reasoning）的描述，由于常识具有不确定性，因此，要实现自然语言描述的常识和常识性推理的形式化就非常困难。针对常识性推理的形式化问题，20 世纪 70 年代末，科学家们提出了非单调逻辑（Non-Monotonic Logic）的思想，即依据常识进行通常的逻辑推理，但保留对常识的不确定性及环境的变迁造成的推理失误的修正权。非单调逻辑是机器模拟人的智能活动取得实质性突破的关键，因此，它已引起学术界的高度重视，并成为计算学科中最富有挑战性的研究方向之一。

至于语言研究中的第四个层次——语用的问题，这是一个较语法和语义更为复杂的问题，本书不再进行讨论。

3.7.8 小结

综上所述，计算机语言经历了从机器语言、汇编语言、高级语言、应用语言到自然语言的发展阶段，其功能变得越来越强大，人机交互也变得越来越方便，这种发展过程反映了人们的认识从感性认识（抽象）上升到理性认识（理论），再回到实践（设计）中来的科学思维方法。

由于高层次语言总要转为低层次语言来解释或执行，因此，带有更高抽象层次的语言系统将更加庞大，对软硬件资源的消耗也就更加严重，应用也将越来越受到硬件的限制，运行效率不可避免地也将越来越低，这就是一般新的系统软件为什么越来越庞大，而又往往在原有机器上运行效率较低（或不能运行）的主要原因之一。在整体能力上，不同层次的语言也有一定的差异，比如，汇编语言所具有的与机器有关的某些功能（如某些 I/O 功能），高级语言就不具备，同时，又由于汇编语言和机器指令的高效性，因此，在一些软件的研制中，汇编语言和机器指令至今仍被人们所使用。

为了人与机器的交流更加友好、方便，计算机语言必然要朝着更高层次的方向发展。随着计算机硬件技术的迅猛发展，高层次语言对计算机软硬件资源相对消耗较大的缺点也就相对减弱了，从而为计算机语言的发展，实现人们认识的螺旋式上升提供了必要的条件。

对计算机语言抽象、理论和设计 3 个学科形态的研究，有助于我们正确理解计算机语言的本质，以及更好地把握它的研究方向，从而能更好地进行计算学科的研究。

最后，讨论计算机语言局限性的问题。

计算机语言是一种形式系统，由于形式系统所固有的局限性，因此，我们可以想象在计算机语言中必然存在一个表达式既不为真，也不为假，它的真假对一个形式系统（计算机语言）而言是不可判定的。

计算机语言方面的内容极其丰富，本节只是从计算机发展的角度作了比较简单的概

述，详细内容请读者查阅有关资料。

思 考 题

1．以“学生选课”为例，分析人们对客观世界的认识过程。

2．什么是概念模型和关系模型？

3．简述计算学科中 3 个学科形态的主要内容。

4．什么是形式语言？试举例说明。

5．图灵机有什么特点？它的工作原理是什么？

6．计算题：在图灵的带子机中，设 b 表示空格，q_1 表示机器的初始状态，q_4 表示机器的结束状态，如果带子上的输入信息是 11100101，读入头对准最右边第一个为 1 的方格，状态为初始状态 q_1。执行以下命令后，请写出计算结果。

$q_1\ 0\ 0\ L\ q_2$
$q_1\ 1\ 0\ L\ q_3$
$q_1\ b\ b\ N\ q_4$
$q_2\ 0\ 0\ L\ q_2$
$q_2\ 1\ 0\ L\ q_2$
$q_2\ b\ b\ N\ q_4$
$q_3\ 0\ 0\ L\ q_2$
$q_3\ 1\ 0\ L\ q_3$
$q_3\ b\ b\ N\ q_4$

7．简述冯·诺伊曼型计算机的组织结构。

8．什么是机器语言？什么是汇编语言？

9．简述 CISC 和 RISC 的设计思想。

10．什么是虚拟机？引入“虚拟机”这一概念有何意义？

11．如何用虚拟机的观点来划分计算机的层次结构？

12．为什么说自然语言的“创造性”过程的本质与计算过程的本质是一致的？

13．自然语言的计算机处理分为哪 4 个层次？

第 4 章　计算学科中的核心概念

认知学科终究是通过概念来实现的，而掌握和应用学科中具有方法论性质的核心概念正是成熟的计算机科学家和工程师的标志之一。

本章分别介绍计算学科中最具有方法论性质的核心概念——算法，以及数据结构、程序、软件、硬件等基本概念，最后还给出了 CC1991 提取的 12 个核心概念。

4.1　算　法

算法是计算学科中最具有方法论性质的核心概念，也被誉为计算学科的灵魂。算法设计的优劣决定着软件系统的性能，对算法进行研究能使我们深刻地理解问题的本质以及可能的求解技术。

4.1.1　算法的历史简介

“算法”（Algorithm）一词在 1957 年之前的《韦氏新世界词典》（*Webster's New World Dictionary*）中还没有出现。现代的数学史学者发现了这一名词的真实来源：公元 825 年，阿拉伯数学家阿科瓦里茨米（AlKhowarizmi）写了著名的《波斯教科书》（*Persian Textbook*），书中概括了进行四则算术运算的法则。“算法”（Algorithm）一词就来源于这位数学家的名字。后来，《韦氏新世界词典》将其定义为“解某种问题的任何专门的方法”。而据考古学家发现，古巴比伦人在求解代数方程时，就已经采用了“算法”的思想。

在算法的研究中，人们常提到丢番图方程，希尔伯特著名的 23 个数学问题中的第 10 个问题就是关于“丢番图方程的可解性问题”。

古希腊数学家丢番图（Diophantus）对代数学的发展有极其重要的贡献，并被后人称之为“代数学之父”。他在《算术》（Arithmetica）一书中提出了有关两个或多个变量整数系数方程的有理数解问题。对于具有整数系数的不定方程，如果只考虑其整数解，这类方程就叫做丢番图方程。

“丢番图方程可解性问题”的实质为：能否写出一个可以判定任意丢番图方程是否可解的算法？本书仅讨论线性丢番图方程，至于非线性丢番图方程，不在本书讨论范围之内。

对于只有一个未知数的线性丢番图方程而言，求解很简单，如 $ax=b$，只要 a 能整除 b，就可判定其有整数解，该整数解即 b/a。

对于有两个未知数的线性丢番图方程判定其是否有解的方法也很简单，如 $ax+by=c$，先求出 a 和 b 的最大公因子 d，若 d 能整除 c，则该方程有解（整数解）。

例 4.1　问：方程 $13x+26y=52$ 有无整数解？

答：13 和 26 的最大公因子是 13，13 又可整除 52，故该方程有整数解（如 $x=2$，$y=1$ 即方程的解）。

例 4.2　问：方程 $2x+4y=15$ 有无整数解？

答：2 和 4 的最大公因子是 2，2 不能整除 15，故该方程无整数解。

因此可以看出，对于两个未知数的线性丢番图方程来说，求解的关键就是求最大公因子。公元前 300 年左右，欧几里德在其著作《几何原本》（*Elements*）第七卷中阐述了关于求解两个数最大公因子的过程，这就是著名的欧几里德算法：给定两个正整数 *m* 和 *n*，求它们的最大公因子，即能同时整除 *m* 和 *n* 的最大正整数。

步骤如下：

（1）以 *n* 除 *m*，并令所得余数为 *r*（*r* 必小于 *n*）；

（2）若 *r*=0，算法结束，输出结果 *n*；否则，继续步骤（3）；

（3）将 *n* 置换为 *m*，*r* 置换为 *n*，并返回步骤（1）继续进行。

例 4.3　设 *m*=56，*n*=32，求 *m*、*n* 的最大公因子。

算法如下:

（1）32 除 56 余数为 24；

（2）24 除 32 余数为 8；

（3）8 除 24 余数为 0，算法结束，输出结果 8。

答：*m*、*n* 的最大公因子为 8。

欧几里德算法既表述了一个数的求解过程，同时，它又表述了一个判定过程，该过程可以判定“*m* 和 *n* 是互质的”（即除 1 以外，*m* 和 *n* 没有公因子）这个命题的真假。

4.1.2　算法的定义和特征

有关算法的定义不少，其内涵基本上是一致的，其中最为著名的是计算机科学家克努特在其经典巨著——《计算机程序设计的艺术》（*The Art of Computer Programming*）第一卷中对算法的定义和特性所作的有关描述。

1. 算法的非形式化定义

一个算法，就是一个有穷规则的集合，其中之规则规定了一个解决某一特定类型问题的运算序列。

2. 算法的重要特性

（1）有穷性：一个算法在执行有穷步之后必须结束。也就是说，一个算法，它所包含的计算步骤是有限的。如在欧几里德算法中，由于 *m* 和 *n* 均为正整数，在步骤（1）之后，*r* 必小于 *n*，若 *r*≠0，下一次进行步骤（1）时，*n* 的值已经减小，而正整数的递降序列最后必然要终止。因此，无论给定 *m* 和 *n* 的原始值有多大，步骤（1）的执行都是有穷次。

（2）确定性：算法的每一个步骤必须要确切地定义。即算法中所有有待执行的动作必须严格而不含混地进行规定，不能有歧义性。如欧几里德算法中，步骤（1）中明确规定“以 *n* 除 *m*”，而不能有类似“以 *n* 除 *m* 或以 *m* 除 *n*”这类有两种可能做法的规定。

（3）输入：算法有零个或多个的输入，即在算法开始之前，对算法最初给出的量。如欧几里德算法中，有两个输入，即 *m* 和 *n*。

（4）输出：算法有一个或多个的输出，即与输入有某个特定关系的量，简单地说就是算法的最终结果。如在欧几里德算法中只有一个输出，即步骤（2）中的 *n*。

（5）能行性：算法中有待执行的运算和操作必须是相当基本的，换言之，它们都是能够精确地进行的，算法执行者甚至不需要掌握算法的含义即可根据该算法的每一步骤要求进行操作，并最终得出正确的结果。

3. 算法的形式化定义

算法的形式化定义：算法是一个四元组，即（Q，I，Ω，F）。

其中：

（1）Q 是一个包含子集 I 和 Ω 的集合，它表示计算的状态；

（2）I 表示计算的输入集合；

（3）Ω 表示计算的输出集合；

（4）F 表示计算的规则，它是一个由 Q 到它自身的函数，且具有自反性，即对于任何一个元素 $q \in Q$，有 $F(q)=q$。

一个算法是对于所有的输入元素 x，都在有穷步骤内终止的一个计算方法。在算法的形式化定义中，对于任何一个元素 $x \in I$，x 均满足以下性质：

$x_0=x$，$x_{k+1}=F(x_k)$，$(k \geqslant 0)$，该性质表示任何一个输入元素 x 均为一个计算序列，即 x_0，x_1，x_2，…，x_k。对任何输入元素 x，该序列表示算法在第 k 步结束。

4.1.3 算法实例

下面，我们再介绍几个简单的算法实例，以加深读者对算法思想的理解。

例 4.4 求 1+2+3+……+100

设变量 X 表示加数，Y 表示被加数，用自然语言将算法描述如下：

（1）将 1 赋值给 X；

（2）将 2 赋值给 Y；

（3）将 X 与 Y 相加，结果存放在 X 中；

（4）将 Y 加 1，结果存放在 Y 中；

（5）若 Y 小于或等于 100，转到步骤（3）继续执行；否则，算法结束，结果为 X。

例 4.5 求解调和级数 H_n

$$H_n=\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$$

调和级数在算法分析中有重要作用。直觉上，当 n 很大时，H_n 也未必会得到很大的值。其实不然，只要 n 充分大，则 H_n 就能得到我们所需要的不论多大的数。这个例子与梵天塔问题一样清晰地表明：在算法的研究中，不能依靠人的直觉，而只能依靠严密的数学方法。

下面，给出求解调和级数的算法。

设变量 X 表示累加和，变量 I 表示循环的次数，自然语言描述算法如下：

（1）将 0 赋值给 X；

（2）将 1 赋值给 I；

（3）将 X 与 $1/I$ 相加，然后把结果存入 X；

（4）将 I 加 1；

（5）若 I 大于等于 N，算法结束，结果为 X；否则转到步骤（3）继续执行。

例 4.6 求解斐波那契数

0，1，1，2，3，5，8，13，21，34，…　（1）

数列（1）即著名的斐波那契数列，它来源于 1202 年意大利数学家斐波那契（L.P.Fibonacci）在其《珠算之书》（*Liber Abaci*）中提出的一个“兔子问题”：

假设一对刚出生的兔子一个月后就能长大，再过一个月就能生下一对兔子，并且此

后每个月都能生一对兔子，且新生的兔子在第二个月后也是每个月生一对兔子。问：一对兔子一年内可繁殖成多少对兔子？

在序列（1）中，每个数都是它的前两个数之和，F_n 表示这个序列的第 n 个数，该序列可以形式化的定义为：

$F_0=0$，$F_1=1$，$F_{n+2}=F_{n+1}+F_n$，$n\geqslant 0$

斐波那契数列不仅包含着一个有趣的“兔子问题”，而且还是一个关于加法算法的典型实例。下面，我们给出求解前 n 个斐波那契数的算法。

设变量 X 表示前一个数的值，即定义中的 F_n，变量 Y 表示当前数的值，即定义中的 F_{n+1}，变量 Z 表示后一个数的值，即定义中的 F_{n+2}。那么求解问题的自然语言描述如下：

（1）如果 n =0，那么将 0 赋值给 Y，并输出 Y，转步骤（11）继续执行；

（2）将 0 赋给 X，将 1 赋值给 Y；

（3）输出 X、Y；

（4）将 1 赋值给 I；

（5）如果 I 大于 n -1，则转到步骤（11），否则继续执行；

（6）将 X 和 Y 的和赋值给 Z；

（7）将 Y 赋值给 X；

（8）将 Z 赋值给 Y；

（9）将 Y 输出；

（10）将 I 加 1，转步骤（5）继续执行；

（11）算法结束。

4.1.4 算法的表示方法

算法是对解题过程的精确描述，这种描述是建立在语言基础之上的，表示算法的语言主要有自然语言、流程图、伪代码、计算机程序设计语言等。

1. 自然语言

前面关于欧几里德算法以及算法实例的描述使用的都是自然语言，自然语言是人们日常所用的语言，如汉语、英语、德语等。使用这些语言不用专门训练，所描述的算法也通俗易懂。然而，其缺点也是明显的：

（1）由于自然语言的歧义性，容易导致算法执行的不确定性；

（2）自然语言的语句一般太长，从而导致了用自然语言描述的算法太长；

（3）由于自然语言表示的串行性，因此，当一个算法中循环和分支较多时就很难清晰地表示出来；

（4）自然语言表示的算法不便翻译成计算机程序设计语言理解的语言。

2. 流程图

流程图是描述算法的常用工具，它采用美国国家标准化协会 ANSI（American National Standard Institute）规定的一组图形符号来表示算法。流程图可以很方便地表示顺序、选择和循环结构，而任何程序的逻辑结构都可以用顺序、选择和循环结构来表示，因此，流程图可以表示任何程序的逻辑结构。另外，用流程图表示的算法不依赖于任何具体的计算机和计算机程序设计语言，从而有利于不同环境的程序设计。就算法的描述

而言，流程图优于其他描述算法的语言。下面，分别给出求解例 4.4、例 4.5 和例 4.6 的流程图算法描述。

（1）求解例 4.4 的算法流程图

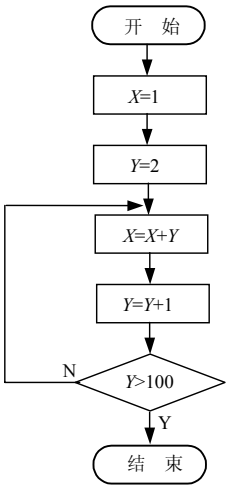


图 4.1　例 4.4 算法流程图

（2）求解例 4.5 的算法流程图

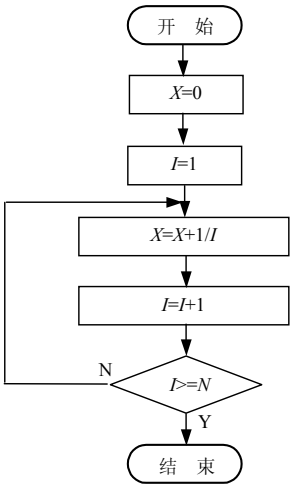


图 4.2　例 4.5 算法流程图

（3）求解例 4.6 的算法流程图

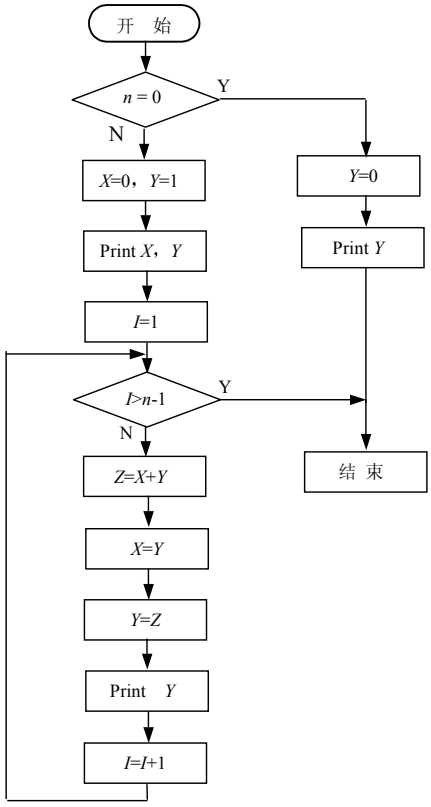


图 4.3　例 4.6 算法流程图

3. 伪代码

伪代码是用介于自然语言和计算机语言之间的文字和符号来描述算法的工具。它不用图形符号，因此，书写方便，格式紧凑，易于理解，便于向计算机程序设计语言算法（程序）过渡。下面，分别给出求解例 4.4、例 4.5 和例 4.6 的伪代码算法描述。

（1）求解例 4.4 的伪代码算法描述：

```
BEGIN(算法开始)
1⇒X
2⇒Y
while（Y<=100）
  {
     X+Y⇒X
     Y+1⇒Y
  }
    Print  X
END（算法结束）
```

（2）求解例 4.5 的伪代码算法描述：

```
BEGIN（算法开始）
0⇒X
1⇒I
```

```
do
  {
       X+1/I ⇒X
       I+1⇒I
   }while(I>=n)
END（算法结束）
```

（3）例 4.6 的伪代码算法描述：

```
BEGIN（算法开始）
if   n = =0
  {
       0⇒Y
       Print Y
  }
else {
     0⇒X
     1⇒Y
     Print   X，Y
     for(i=1;i<=n-1;i++)
     {
        X+Y⇒Z
        Y⇒X
        Z⇒Y
        Print   Y
     }
  }
END（算法结束）
```

4. 计算机程序设计语言

计算机不识别自然语言、流程图和伪代码等算法描述语言，而设计算法的目的就是要用计算机解决问题，因此，用自然语言、流程图和伪代码等语言描述的算法最终还必须转换为具体的计算机程序设计语言描述的算法，即转换为具体的程序。

一般而言，计算机程序设计语言描述的算法（程序）是清晰的、简明的，最终也能由计算机处理的。然而，就使用计算机程序设计语言描述算法而言，它还存在以下几个缺点：

（1）算法的基本逻辑流程难于遵循。与自然语言一样，程序设计语言也是基于串行的，当算法的逻辑流程较为复杂时，这个问题就变得更加严重；

（2）用特定程序设计语言编写的算法限制了与他人的交流，不利于问题的解决；

（3）要花费大量的时间去熟悉和掌握某种特定的程序设计语言；

（4）要求描述计算步骤的细节，而忽视算法的本质。

下面，分别给出求解例 4.4、例 4.5 和例 4.6 的计算机程序设计语言（C 语言）的算

法描述。

（1）求解例 4.4 的计算机程序设计语言（C 语言）的算法描述：

```
main()
{
        int X,Y;
        X=1;
        Y=2;
        while(Y<=100)
        {
           X=X+Y;
           Y=Y+1;
        };
        printf("%d",X);
}
```

（2）求解例 4.5 的计算机程序设计语言（C 语言）的算法描述：

```
main()
{
        int    n;
        float X,I;
        printf("Please input n:");
        scanf("%d",&n);
        X=0;
        I=1;
        do
       {
           X=X+1/I;
           I=I+1;
       }while(I<=n);
          printf("\n%f",X);
}
```

（3）求解例 4.6 的计算机程序设计语言（C 语言）的算法描述：

```
main()
{
      int    X,Y,Z,I,j,n;
      printf("please input n:");
      scanf("%d",&n);
      printf("\n");
      if (n==0)
        {
          Y=0;
          printf("%d    " , Y);
```

```
        }
      else
        {
          X=0;
          Y=1;
          printf("%d    %d    ", X ,Y);
          for(I=1;I<=n-1;I++)
             {
               Z=X+Y;
               X=Y;
               Y=Z;
               printf("%d    ", Y);
             }
        }
}
```

4.1.5 算法分析

解一个问题，往往有若干不同的算法，这些算法决定着根据该算法编写的程序性能的好坏。那么，在保证算法正确性的前提下，如何确定算法的优劣就是一个值得研究的课题。

在算法的分析中，一般应考虑以下 3 个问题：

（1）算法的时间复杂度；

（2）算法的空间复杂度；

（3）算法是否便于阅读、修改和测试。

算法时间复杂度是指算法中有关操作次数的多少，它用 $T(n)$表示，T为英文单词Time的第一个字母，$T(n)$中的 n 表示问题规模的大小。如在累加求和中，n 表示待加数的个数；在矩阵相加问题中，n 表示矩阵的阶数；在图中，n 表示顶点数等。

在算法的复杂度分析中，经常使用一个记号O（读作“大O”），该记号是保罗·巴克曼（P.Bachmann）于 1892 年在《解析数论》（*Analytische Zahlentheorie*）一书引进的，它为 Order（数量级）的第一个字母，它允许使用“=”代替“≈”。如 $n^2+n+1=O(n^2)$，该表达式表示，当 n 足够大时表达式左边约等于 n^2。

设 $f(n)$是一个关于正整数 n 的函数，若存在一个正整数 n_0和一个常数 C，当 $n \geqslant n_0$ 时，$|T(n)| \leqslant |C f(n)|$均成立，则称 $f(n)$为 $T(n)$的同数量级的函数。于是，算法时间复杂度 $T(n)$可表示为：

$$T(n)= O(f(n))$$

常见的大O表示形式有：

- $O(1)$称为常数级；
- $O(\log n)$称为对数级；
- $O(n)$称为线性级；
- $O(n^c)$称为多项式级；
- $O(c^n)$称为指数级；

- $O(n!)$称为阶乘级。

用以上表示方法，在第 2 章的梵天塔问题中，需要移动的盘子次数为 $h(n)=2^n-1$，则该问题的算法时间复杂度表示为$O(2^n)$；例 4.4 的算法时间复杂度表示为$O(1)$；例 4.5 的算法时间复杂度表示为$O(n)$；例 4.6 的算法时间复杂度表示为$O(n)$等等。

一般而言，对于较复杂的算法，应将它分成容易估算的几个部分，然后用O的求解原则计算整个算法的时间复杂度，最好不要采用指数级和阶乘级的算法，而应尽可能选用多项式级或线性级等时间复杂度较小的算法。另外，还要在算法最好、平均和最坏的情况下区别执行效率的不同。

在阶乘级的算法中，如果问题规模 n 为 10，则算法时间复杂度为 10!（3，628，800）。若要检验 10! 种情况，设每种情况需要 1 毫秒的计算时间，则整个计算将需 1 小时左右。一般来说，如果选用了阶乘级的算法，则当问题规模等于或者大于 10 时，就要认真考虑算法的适用性问题。

算法的空间复杂度是指算法在执行过程中所占存储空间的大小，它用 $S(n)$表示，S 为英文单词 Space 的第一个字母。与算法的时间复杂度相同，算法的空间复杂度 $S(n)$也可表示为：$S(n)= O(g(n))$。

4.2 数据结构

数学模型有定量模型和定性模型两类之分。定量模型指的是可以用数值方程表示的一类模型，而定性模型则是指非数值性的数据结构（如表、树和图等）及其运算。

在计算领域中，数据结构（Data Structure）指的是一类定性数学模型，它是计算机算法设计的基础，它在计算科学中占有十分重要的地位。本节将介绍数据结构的基本概念和常用的几种数据结构，如线性表、数组、树和二叉树以及图等。

4.2.1 数据结构的基本概念

1. 数据结构的组成

数据结构是一类定性的数学模型，它由数据的逻辑结构、数据的存储结构（或称物理结构）及其运算 3 部分组成。

2. 数据逻辑结构的形式化定义

$DS=<D，R>$

其中：

D 表示数据的集合；

R 表示数据 D 上关系的集合。

3. 数据的存储结构

数据的存储结构是指在反映数据逻辑关系的原则下，数据在存储器中的存储方式。数据存储结构的基本组织方式有顺序存储结构和链式存储结构。

（1）顺序存储结构：借助元素在存储器中的相对位置来表示数据元素的逻辑关系。

（2）链式存储结构：借助指针来表示数据元素之间的逻辑关系，通常在数据元素上增加一个或多个指针类型的属性来实现这种表示方式。

4. 数据结构的基本运算内容

（1）建立数据结构；
（2）清除数据结构；
（3）插入数据元素；
（4）删除数据元素；
（5）更新数据元素；
（6）查找数据元素；
（7）按序重新排列；
（8）判定某个数据结构是否为空，或是否已达到最大允许的容量；
（9）统计数据元素的个数。

4.2.2 常用的几种数据结构

1. 线性表

线性表是 n 个数据元素的有限序列，即（X[1]，X[2]，X[3]，…，X[i]，…，X[n]）。插入、删除和存取数据元素是所有数据结构的基本操作，若对线性表的这些基本操作加一定的限制，则形成下面几种特殊的线性表：

（1）后进先出（Last In First Out，简称 LIFO）的线性表。它的所有插入、删除和存取都是在线性表的表尾进行的；

（2）先进先出（First In First Out，简称 FIFO）的线性表。它的所有插入都是在线性表的一端进行的，而所有的删除和存取又都在线性表的另一端进行；

（3）限定所有插入、删除和存取都在表的两端进行的线性表。

2. 数组

数组是线性表的推广形式之一。如在一个 $m \times n$ 的二维数组中，每个元素 $A[i,j]$都分别属于两个线性表，即（$A[i,1]$，$A[i,2]$，…，$A[i,n]$）和（$A[1,j]$，$A[2,j]$，…，$A[m,j]$）。

4.2.3 树和二叉树

树和二叉树是一种具有层次关系的非线性结构，在计算机领域中有广泛的应用，尤其以二叉树最为常用。

1. 树

树（Tree）是由 n（$n \geqslant 0$）个结点组成的有限集合。若 $n=0$，则称为空树，任何一个非空树均满足以下两个条件：

（1）仅有一个称为根的结点；

（2）当 $n>0$ 时，其余结点可分为 m（$m \geqslant 0$）个互不相交的有限集合，其中每个集合又是一棵树，并称为根的子树。

例如，图 4.4 所示的树中有 12 个结点，A 是根结点，该树又可再分为若干不相交的子树，如 T1={B，E，F，K}，T2={C，G}，T3={D，H，I，J，L}等。

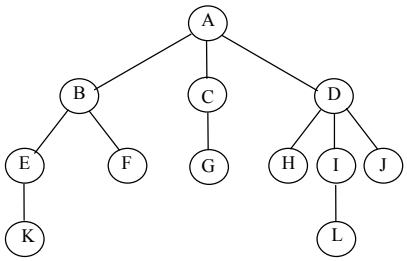


图 4.4　树

2. 二叉树

二叉树是 n（$n \geqslant 0$）个结点组成的有限集合，它或者是空集（$n=0$），或者由一个结点及两棵互不相交的子树组成，且这两个子树有左、右之分，其次序不能任意颠倒。

4.2.4　图

图是由结点和连接这些结点的边所组成的集合。在图形结构中，结点之间的关系可以是任意的，图中任意两个数据元素之间都可能相关。

图的形式化定义为：$G=<V，E>$

其中：

V 是一个非空结点的集合；

E 是连结结点的边的集合。

例　$G=<V，E>$

其中 $V=\{A，B，C，D\}$，$E=\{(A，B)，(A，C)，(B，D)，(B，C)，(D，C)，(A，D)\}$。

该图也可以用图 4.5 来表示。

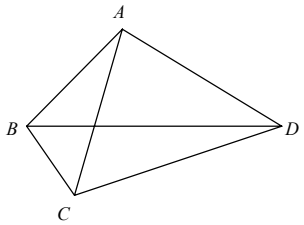


图 4.5　图的表示

4.3　程　序

“程序”一词，从广义上讲可以认为是一种行动方案或工作步骤。在日常生活中，我们经常会碰到这样的程序：某个会议的日程安排、一条旅游路线的设计、手工小制作的说明书等，这些程序表示的都是我们在做一件事务时按时间的顺序应先做什么后做什么。在本书中所说的程序指的是计算机程序，它实际上表示的也是一种处理事务的时间顺序和处理步骤。由于组成计算机程序的基本单位是指令，因此，计算机程序就是按照工作步骤事先编排好的、具有特殊功能的指令序列。

一个程序具有一个单一的、不可分的结构，它规定了某个数据结构上的一个算法。瑞士著名计算机科学家尼可莱·沃思（Nikiklaus Wirth）在 1976 年曾提出这样一个公式：

算法+数据结构=程序

由此看来，我们前面提到的算法和数据结构是计算机程序的两个最基本的概念。算法是程序的核心，它在程序编制、软件开发，乃至在整个计算机科学中都占据重要地位。数据结构是加工的对象，一个程序要进行计算或处理总是以某些数据为对象的，而要设计一个好的程序就需将这些松散的数据按某种要求组成一种数据结构。然而，随着计算机科学的发展，人们现在已经意识到程序除了以上两个主要要素外，还应包括程序的设计方法以及相应的语言工具和计算环境。

4.4 软　件

软件是与程序密切相关的一个概念，在计算机发展的初期，硬件设计和生产是主要问题，那时的软件就是程序。后来，随着计算技术的发展，传统软件的生产方式已不适应发展的需要，于是人们将工程学的基本原理和方法引入软件设计和生产中。现在计算机软件一般指计算机系统中的程序及其文档，也可以指在研究、开发、维护，以及使用上述含义下的软件所涉及的理论、方法、技术所构成的分支学科。软件一般分为系统软件、支撑软件、应用软件 3 类。

（1）系统软件是计算机系统中最靠近硬件层次的软件。如操作系统、编译程序等。

（2）支撑软件是支撑其他软件的开发与维护的软件。如数据库管理系统、网络软件、各种接口软件和开发工具等。

（3）应用软件是特定应用领域的专用软件。如商业会计软件、教学软件等。

4.5 硬　件

计算机硬件是构成计算机系统的所有物理器件、部件、设备，以及相应的工作原理与设计、制造、检测等技术的总称。广义的硬件包含硬件本身及其工程技术两部分。其中：

计算机系统的物理元器件包括：集成电路、印制电路板，以及其他磁性元件、电子元件等。

计算机系统的部件和设备包括：控制器、运算器、存储器、输入输出设备、电源等。

硬件工程技术包括：印制电路板制造、高密度组装、抗环境干扰、抗恶劣环境破坏等技术，以及在设计和制造过程中为提高计算机性能所采取的措施等。

计算机就是由计算机硬件和计算机软件组成的。硬件是计算机的“躯体”，软件是计算机的“灵魂”。

4.6 CC1991报告提取的核心概念

核心概念是 CC1991 报告首次提出的，是具有普遍性、持久性的重要思想、原则和方法。它的基本特征有：

（1）在学科中多处出现；

（2）在各分支领域及抽象、理论和设计的各个层面上都有很多示例；

（3）在技术上有高度的独立性；

（4）一般都在数学、科学和工程中出现。

CC1991 报告的重要贡献之一就是提取了计算学科中具有方法论性质的 12 个反复出现的核心概念。

1．绑定（Binding）

绑定指的是通过将一个对象（或事物）与其某种属性相联系，从而使抽象的概念具体化的过程。例如：将一个进程与一个处理机，一个变量与其类型或值分别联系起来。这种联系的建立，实际上就是建立了某种约束。

2．大问题的复杂性（Complexity of Large Problems）

大问题的复杂性是指随着问题规模的增长而使问题的复杂性呈非线性增加的效应。这种非线性增加的效应是区分和选择各种现有方法和技术的重要因素。

3．概念和形式模型（Conceptual and Format Models）

概念和形式模型是对一个想法或问题进行形式化、特征化、可视化思维的方法。抽象数据类型、语义数据类型以及指定系统的图形语言，如数据流图和 E-R 图等都属于概念模型。而逻辑、开关理论和计算理论中的模型大都属于形式模型。概念模型和形式模型以及形式证明是将计算学科各分支统一起来的重要的核心概念。

4．一致性和完备性（Consistency and Completeness）

一致性包括用于形式说明的一组公理的一致性、事实和理论的一致性，以及一种语言或接口设计的内部一致性。完备性包括给出的一组公理，使其能获得预期行为的充分性、软件和硬件系统功能的充分性，以及系统处于出错和非预期情况下保持正常行为的能力等。

在计算机系统设计中，正确性、健壮性和可靠性就是一致性和完备性的具体体现。

5．效率（Efficiency）

效率是关于空间、时间、人力、财力等资源消耗的度量。在计算机软硬件的设计中，要充分考虑某种预期结果所达到的效率，以及一个给定的实现过程较之替代的实现过程的效率。

6．演化（Evolution）

演化指的是系统的结构、状态、特征、行为和功能等随着时间的推移而发生的更改。这里主要是指了解系统更改的事实和意义及应采取的对策。在软件进行更改时，不仅要充分考虑更改时对系统各层次造成的冲击，还要充分考虑到软件的有关抽象、技术和系统的适应性问题。

7．抽象层次（Levels of Abstraction）

抽象层次指的是通过对不同层次的细节和指标的抽象对一个系统或实体进行表述。在复杂系统的设计中，隐藏细节，对系统各层次进行描述（抽象），从而控制系统的复杂程度。例如，在软件工程中，从规则说明到编码各个阶段（层次）的详细说明，计算

机系统的分层思想，计算机网络的分层思想等。

8. 按空间排序（Ordering in Space）

按空间排序指的是各种定位方式，如物理上的定位（如网络和存储中的定位），组织方式上的定位（如处理机进程、类型定义和有关操作的定位）以及概念上的定位（如软件的辖域、耦合、内聚等）。按空间排序是计算技术中一个局部性和相邻性的概念。

9. 按时间排序（Ordering in Time）

按时间排序指的是事件的执行对时间的依赖性。例如，在具有时态逻辑的系统中，要考虑与时间有关的时序问题；在分布式系统中，要考虑进程同步的时间问题；在依赖于时间的算法执行中，要考虑其基本的组成要素。

10. 重用（Reuse）

重用指的是在新的环境下，系统中各类实体、技术、概念等可被再次使用的能力。如软件库和硬件部件的重用等。

11. 安全性（Security）

安全性指的是计算机软硬件系统对合法用户的响应及对非法请求的抗拒，以保护自己不受外部影响和攻击的能力。如为防止数据的丢失、泄密而在数据库管理系统中提供的口令更换、操作员授权等功能。

12. 折衷和结论（Tradeoff and Consequences）

折衷指的是为满足系统的可实施性而对系统设计中的技术、方案所作出的一种合理的取舍。结论是折衷的结论，即选择一种方案代替另一种方案所产生的技术、经济、文化及其他方面的影响。折衷是存在于计算学科领域各层次上的基本事实。如在算法的研究中，要考虑空间和时间的折衷；对于矛盾的设计目标，要考虑诸如易用性和完备性、灵活性和简单性、低成本和高可靠性等方面所采取的折衷等。

思 考 题

1．什么是算法？算法有何特征？

2．表示算法的语言有哪几种？

3．判定方程 $3x+5y=2$ 是否有整数解。

4．用欧几里德算法分别求下列自然数的最大公因子：

（1）18，12　（2）21，9　（3）83，19　（4）201，81　（5）216，78

5．设 $e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\cdots$，请分别用自然语言、流程图和伪代码写出求解 e 的近似值的算法。

6．在算法的分析中应考虑哪些问题？

7．什么是数据结构？数据结构的基本运算内容是什么？

8．常用的数据结构有哪几种？

9．什么是程序？程序包括哪些基本要素？

10．什么是软件？什么是硬件？

11．CC1991 报告提取了哪些核心概念？

第 5 章　计算学科中的数学方法

数学方法是指解决数学问题的策略、途径和步骤，它是计算学科中最根本的研究方法。理论上，凡能被计算机处理的问题均可以转换为一个数学问题，换言之，所有能被计算机处理的问题均可以用数学方法解决；反之，凡能以离散数学为代表的构造性数学方法描述的问题，当该问题所涉及的论域为有穷，或虽为无穷但存在有穷表示时，这个问题也一定能用计算机来处理。

本章主要介绍数学的基本特征、数学方法的作用，计算学科中常用的数学概念和术语、证明方法、递归和迭代、公理化方法、形式化方法以及 Armstrong 公理系统等内容。

5.1　数学的基本特征

数学是研究现实世界的空间形式和数量关系的一门科学。它具有以下三个基本特征：

（1）高度的抽象性

抽象是任何一门科学乃至全部人类思维都具有的特性，然而，数学的抽象程度大大超过自然科学中一般的抽象，它最大的特点在于抛开现实事物的物理、化学和生物学等特性，而仅保留其量的关系和空间的形式。

（2）逻辑的严密性

数学高度的抽象性和逻辑的严密性是紧密相关的。若数学没有逻辑的严密性，在自身理论中矛盾重重，漏洞百出，那么用数学方法对现实世界进行抽象就失去了意义。正是由于数学的逻辑严密性，需要我们在运用数学工具解决问题时，只有严格遵守形式逻辑的基本法则，充分保证逻辑的可靠性，才能保证结论的正确性。

（3）普遍的适用性

数学的高度抽象性决定了它的普遍适用性。数学广泛地应用于其他科学与技术，甚至人们的日常生活之中。

5.2　数学方法的作用

数学方法在现代科学技术的发展中已经成为一种必不可少的认识手段，它在科学技术方法论中的作用主要表现在以下 3 个方面：

（1）为科学技术研究提供简洁精确的形式化语言

人类在日常交往中使用的语言称自然语言，它是人与人之间进行交流和对现实世界进行描述的一般的语言工具。而随着科学技术的迅猛发展，对于微观和宏观世界中存在的复杂的自然规律，只有借助于数学的形式化语言才能抽象地表达。许多自然科学定律，如牛顿的万有引力定律等，都是用简明的数学公式表示的。数学模型就是运用数学的形

式化语言，在观测和实验的基础上建立起来的，它有助于人们认识和把握超出感性经验之外的客观世界。

（2）为科学技术研究提供数量分析和计算的方法

一门科学要从定性分析发展到定量分析，数学方法从中起了杠杆的作用。计算机的问世更为科学的定量分析和理论计算提供了必要条件，使一些过去无法解决的数学课题找到了解决的可能性。如原子能的研究和开发、空间技术的发展等都是借助于精确的数值计算和理论分析进行的。

（3）为科学技术研究提供了逻辑推理的工具

数学的逻辑严密性这一特点使它成为建立一种理论体系的手段，在这方面最有意义的就是公理化方法。数学逻辑用数学方法研究推理过程，把逻辑推理形式加以公理化、符号化，为建立和发展科学的理论体系提供了有效的工具。

5.3 计算学科中常用的数学概念和术语

5.3.1 集合

1. 集合的概念

集合是数学的基本概念，它是构造性数学方法的基础。集合就是一组无重复的对象的全体。集合中的对象称为集合的元素。如：计算机专业学生全部必修课程可以组成一个集合，其中的每门课程就是这一集合中的元素。

2. 集合的描述方法

通常用大写字母表示集合，用小写字母表示元素，描述集合的方式主要有以下 3 种：

（1）枚举法：列出所有元素的表示方法。

如 1 至 5 的整数集合可表示为：

A={1，2，3，4，5}；

（2）外延表示法：当集合中所列元素的一般形式很明显时，可只列出部分元素，其他则用省略号表示。

如斐波那契数列可表示为：

{ 0，1，1，2，3，5，8，13，21，34，… }；

（3）谓词表示法：用谓词来概括集合中元素的属性。

如斐波那契数列可表示为：

$\{F_n \mid F_{n+2}=F_{n+1}+F_n，F_0=0，F_1=1，n\geqslant 0\}$

3. 集合的运算

集合的基本运算有并、差、交、补和乘积等运算。

（1）集合的并

设 A、B 为两个任意集合，所有属于 A 或属于 B 的元素构成的集合 C，称为 A 和 B 的并集。可表示为：$C=A\cup B=\{x \mid x\in A\vee x\in B\}$。

求并集的运算称为并（运算）。

例 5.1　若 A={a，b，c，d}，B={b，d，e}，求集合 A 和 B 的并。

解：$A \cup B = \{a, b, c, d, e\}$

（2）集合的差

设 A、B 为两个任意集合，所有属于 A 而不属于 B 的一切元素构成的集合 S，称为 A 和 B 的差集。可表示为：$S=A-B=\{x \mid x \in A \vee x \in B\}$。

求差集的运算称为差（运算）。

例 5.2　若 $A=\{a, b, c, d\}$，$B=\{b, d, e\}$，求集合 A 和 B 的差。

解：$A-B=\{a, c\}$

（3）集合的交

设 A、B 为两个任意集合，由 A 和 B 的所有相同元素构成的集合 C，称为 A 和 B 的交集。可表示为：$C=A \cap B = \{x \mid x \in A \wedge x \in B\}$。

求交集的运算称为交（运算）。

例 5.3　若 $A=\{x \mid x > -5\}$，$B=\{x \mid x<1\}$，求集合 A 和 B 的交。

解：$A \cap B = \{x \mid x > -5\} \cap \{x \mid x<1\} = \{x \mid -5<x<1\}$

（4）集合的补

设 I 为全集，A 为 I 的任意一子集，$I-A$ 则为 A 的补集，记为 $\overline{A}$。可表示为

$\overline{A}=I-A=\{x \mid x \in I, x \notin A\}$

求补集的运算称为补（运算）。

例 5.4　若 $I=\{x \mid -5 < x < 5\}$，$A=\{x \mid 0 < x < 1\}$，求 $\overline{A}$。

解：$\overline{A}=I-A=\{x \mid -5 < x < 0 \vee 1 < x < 5\}$

（5）集合的乘积

集合 A_1，A_2，…，A_n 的乘积一般用法国数学家笛卡尔（Rene Descartes）的名字命名，即笛卡尔积。该乘积表示如下：

$A_1 \times A_2 \times \cdots A_n = \{(a_1, a_2, \cdots, a_n) \mid a_i \in A_i, i=1, 2, \cdots, n\}$

$A_1 \times A_2 \times \cdots A_n$ 的结果是一个有序 n 元组的集合。

例 5.5　若 $A=\{1, 2, 3\}$，$B=\{a, b\}$，求 $A \times B$。

解：$A \times B = \{(1, a), (1, b), (2, a), (2, b), (3, a), (3, b)\}$

5.3.2　函数和关系

1. 函数

函数又称映射，是指把输入转变成输出的运算，该运算也可理解为从某一“定义域”的对象到某一“值域”的对象的映射。函数是程序设计的基础，程序定义了计算函数的算法，而定义函数的方法又影响着程序语言的设计，好的程序设计语言一般都便于函数的计算。

设 f 为一个函数，当输入值为 a 时输出值为 b，则记作：

$$f(a) = b$$

2. 关系

关系是一个谓词，其定义域为 k 元组的集合。通常的关系为二元关系，其定义域为有序对的集合，在这个集合中，我们说有序对的第一个元素和第二个元素有关系。如学生选课（如图 5.1 所示）。

学生	课程	成绩
张三	文学	90
张三	哲学	95
李四	数学	80
李四	艺术	85
王五	历史	92
王五	文学	88

图 5.1　学生选课表

在图 5.1 中，我们用关系的术语可以说，张三与文学以及哲学课有关系（选课关系），李四与艺术以及数学课有关系，王五与历史以及文学课有关系。

3. 等价关系

在关系中，有一种特殊的关系，即等价关系，它满足以下 3 个条件：

（1）自反性，即对集合中的每一个元素 a，都有 aRa；

（2）对称性，即对集合中的任意元素 a，b，aRb 成立当且仅当 bRa 成立；

（3）传递性，即对集合中的任意元素 a，b，c，若 aRb 和 bRc 成立，则 aRc 一定成立。

等价关系的一个重要性质是：集合 A 上的一个等价关系 R 可将 A 划分为若干个互不相交的子集，称为等价类。

例 5.6　证明非负整数 N 上的模 3 的同余关系 R 为等价关系

证明

首先将该关系形式化地表示为：

$R=\{(a, b) \mid a, b \in N, a \bmod 3=b \bmod 3\}$

（1）自反性证明

对集合中的任何一个元素 $a \in N$，都有 $a \bmod 3= a \bmod 3$；

（2）对称性证明

对集合中的任意元素 a，$b \in N$，若 $a \bmod 3=b \bmod 3$，则有

$b \bmod 3=a \bmod 3$；

（3）传递性证明

对集合中的任意元素 $a, b, c \in N$，若 $a \bmod 3=b \bmod 3$，$b \bmod 3=c \bmod 3$，则有 $a \bmod 3=c \bmod 3$。

综上所述，该关系满足自反性、对称性和传递性，因此该关系为等价关系。

例 5.7　假设某人在唱歌（事件 e_1）的同时，还可以开车（事件 e_2）或者步行（事件 e_3），但一个人不能同时开车和步行。

问：以上反映的并发现象，如用关系来表示时，是否是等价关系？

答：以上反映的是一种并发（co）现象，如用关系来表示，则这种并发关系具有自反性和对称性，即可表示为：$e_1\ co\ e_1$，$e_2\ co\ e_2$，$e_3\ co\ e_3$；以及 $e_1\ co\ e_2$（或 $e_2\ co\ e_1$），$e_1\ co\ e_3$（或 $e_3\ co\ e_1$），但不满足传递性，即（$e_2\ co\ e_1$）∧（$e_1\ co\ e_3$）不能推出 $e_2\ co\ e_3$，即不能在开车的同时，又步行。因此，以上并发关系不是等价关系。

5.3.3　字母表、字符串和语言

所有的计算机程序设计语言都是形式语言，其构成基础同一般自然语言一样，也是符号或字母。常用的符号有数字（0~9）、大小写字母（A～Z，a～z）、括号、运算符（+，-，*，/）等。

有限字母表指的是由有限个任意符号组成的非空集合，简称为字母表，用∑表示。字母表上的元素称作字符或符号，用小写字母或数字表示，如 a、b、c、1、2、3 等。

字母表可以理解为计算机输入键盘上符号的集合。字母可以理解为键盘上的每一个英文字母、数字、标点符号、运算符号等。

字符串，也称为符号串，指的是由字符组成的有限序列，常用小写希腊字母表示。字母表∑上的字符串以下列方式生成：

（1）ε 为∑上的一个特殊串，称为空串，对任何 $a\in\Sigma$，$a\varepsilon=\varepsilon a=a$；

（2）若 σ 是∑上的符号串，且 $a\in\Sigma$，则 σa 是∑上的符号串；

（3）若 α 是∑上的符号串，当且仅当它由（1）和（2）导出。

直观来说，∑上的符号串是由其上的符号以任意次序拼接起来构成的，任何符号都可以在串中重复出现。ε 作为一个特殊的串，由零个符号组成。应当指出的是，空串 ε 不同于我们计算机键盘上的空格键。

语言指的是给定字母表∑上的字符串的集合。例如，当∑={a，b}，则{ab，$aabb$，$abab$，bba}、{ε}、（$a^nb^n \mid n\geqslant 1$）都是∑上的语言。不包含任何字符串的语言称作空语言，用 Φ 表示。注意：{ε} 不同于 Φ，前者表示由空串组成的语言，后者表示空语言。

语言是字符串的集合，因此，传统的集合运算（如并、交、差、补、笛卡尔积）对语言都适用。除此之外，语言还有一种重要的专门的运算，即闭包运算。

语言、文法以及自动机有着密切的关系。语言由文法产生。短语结构语言、上下文有关语言、上下文无关语言和正规语言分别由 0 型文法、1 型文法、2 型文法和 3 型文法产生。自动机是识别语言的数学模型，各类文法所对应的自动机分别是图灵机、线性有界自动机、下推自动机和有限状态自动机。

需要指出的是，语言与数学模型不是一一对应的关系，一种语言可以由不同的文法产生，也可以由不同的自动机识别。

5.3.4 布尔逻辑

布尔逻辑是建立在 TRUE 和 FALSE 两个值上的数学体系。值 TRUE 和 FALSE 称作布尔值，一般用“0”和“1”表示。

最基本的布尔运算是“非（NOT，符号为¬）”和“并（AND，符号为∧）”运算，其他的布尔运算如“或（OR，符号为∨）”、“异或（XOR，符号为⊕）”、“等值（Equivalence，符号为↔）”、“蕴含（Implication，符号为→）”运算均可以用最基本的“非”和“并”运算来表示。

下面，综合列出各运算的定义：

¬0=1；　¬1=0；

0∧0=0；0∧1=0；1∧0=0；1∧1=1；

0∨0=0；0∨1=1；1∨0=1；1∨1=1；

0⊕0=0；　0⊕1=1；1⊕0=1；1⊕1=0；

0↔0=1；0↔1=0；1↔0=0；1↔1=1；

$0 \to 0=1$；$0 \to 1=1$；$1 \to 0=0$；$1 \to 1=1$；

5.3.5 定义、定理和证明

定义、定理和证明是数学的核心，也是计算学科理论形态的核心内容。其中，定义是蕴含在公理系统之中的概念和命题；定理是被证明为真的数学命题；证明是为使人们确信一个命题是真的而作的一种逻辑论证。

例 5.8　在欧氏几何中，平面角的定义为：平面角是在一平面内，但不在一直线上的两条相交线的相互倾斜度；等腰三角形的定理为：两边相等的三角形为等腰三角形。

5.4 证明方法

5.4.1 直接证明法和间接证明法

1. 直接证明

假定 p 为真，通过使用公理或已证明的定理以及正确的推理规则证明 q 也为真，以此证明蕴含式 $p \to q$ 为真。这种证明方法为直接证明法。

例 5.9　用直接证明法证明“若 p 是偶数，则 p^2 是偶数”。

证明：假定 p 是偶数为真，设 $p=2k$（k 为整数）。由此可得，$p^2=2$（$2k^2$）。因此，p^2 是偶数（它是一个整数的 2 倍）。

2. 间接证明

因为蕴含式 $p \to q$ 与其逆否命题 $\neg q \to \neg p$ 等价，因此可以通过证明 $\neg q \to \neg p$ 来证明蕴含式 $p \to q$ 为真。这种证明方法为间接证明法。

例 5.10　用间接证明法证明“若 p^2 是偶数，则 p 是偶数”。

证明：假定此蕴含式后件为假，即假定 p 是奇数。则对某个整数 k 来说有 $p=2k+1$。由此可得 $p^2=4k^2+4k+1=2$（$2k^2+2k$）$+1$，因此，p^2 是奇数（它是一个整数的 2 倍加 1）。因为对这个蕴含式后件的否定蕴含着前件为假，因此该蕴含式为真。

5.4.2 反证法

首先假定一个与原命题相反的命题成立，然后通过正确的推理得出与已知（或假设）条件、公理、已证过的定理等相互矛盾或自相矛盾的结果，以此证明原命题正确。这种证明方法就是反证法，也称归谬法，是一种常用的数学证明方法。

例 5.11　证 $\sqrt{2}$ 是无理数

$\sqrt{2}$ 作为无理数的发现，是数学史上的一个重要事件，公元前 500 年，毕达哥拉斯（Pythagoras）学派提出了“万物皆数”的命题，把数归结为整数或整数之比。而 $\sqrt{2}$ 的发现，使当时人们的认识产生了混乱，并导致了第一次数学危机。

传说毕氏学派的希帕索斯（Hippasus）因最先发现了 $\sqrt{2}$ 为不可公度比数，而被该学派投入大海。鉴于该学派迫害数学人才的“无理行为”，15 世纪的达 • 芬奇（L.Da.Vinci）将不可公度比数（既无限不循环小数）称为无理数。

本例是数学反证法的一个典型实例，下面证明之。

证明：

假设$\sqrt{2}$是有理数，则$\sqrt{2}=\frac{q}{p}$（p、q为正整数，且p、q无公因子）。

两边同时平方：$(\sqrt{2})^2=(\frac{q}{p})^2$

$$2=\frac{q^2}{p^2}$$

$$q^2=2p^2\cdots\cdots\cdots（1）$$

因为q^2为偶数，故q一定为偶数，对某个整数m来说则有

$$q=2m\cdots\cdots\cdots（2）$$

将（2）代入（1），可得$(2m)^2=2p^2$

$$p^2=2m^2$$

p^2为偶数，则p也为偶数，p和q均为偶数与原假设p和q无公因子相矛盾。因此，$\sqrt{2}$是无理数为真。

5.4.3 归纳法

1. 归纳法的定义

所谓归纳法，是指从特殊推理出一般的一种证明方法。归纳法可分为不完全归纳法、完全归纳法和数学归纳法。

2. 不完全归纳法

不完全归纳法是根据部分特殊情况作出推理的一种方法，该方法多用于无穷对象的论证，然而，论证的结果不一定正确。因此，不完全归纳法不能作为严格的证明方法。

3. 完全归纳法

完全归纳法也称穷举法，它是对命题中存在的所有特殊情况进行考虑的一种方法，用该方法论证的结果是正确的，然而，它只能用于“有限”对象的论证。

4. 数学归纳法

（1）数学归纳法的概念

数学归纳法是一种用于证明与自然数n有关的命题正确性的证明方法，该方法能用“有限”的步骤解决无穷对象的论证问题。数学归纳法广泛地应用于计算理论研究之中，如算法的正确性证明、图与树的定理证明等方面。

（2）数学归纳法的基本原理

数学归纳法由归纳基础和归纳步骤两个部分组成，其基本原理如下：

假定对一切正整数n，有一个命题$P(n)$，若以下证明成立，则$P(n)$为真：

① 归纳基础：证明$P(1)$为真；

② 归纳步骤：证明对任意的$i\geqslant 1$，若$P(i)$为真，则$P(i+1)$为真。

（3）数学归纳法的形式化定义

根据数学归纳法的原理，可以对数学归纳法形式化地定义为：

$$P(1)\wedge(\forall n)(P(n)\rightarrow P(n+1))\rightarrow \forall n\,P(n)$$

（4）实例

例 5.12　求证命题$P(n)$：“从1开始连续n个奇数之和是n的平方”，即公式

$1+3+5+\cdots+(2n-1)=n^2$ 成立。

证明：

① 归纳基础：当 $n=1$ 时，等式成立，即 $1=1^2$

② 归纳步骤：

设对任意 $k\geqslant 1$，$P(k)$成立，即：

$$1+3+5+\cdots+(2K-1)=K^2$$

而

$$1+3+5+\cdots+(2K-1)+(2(K+1)-1)$$
$$=K^2+2K+1=(K+1)^2$$

则当 $P(k)$成立时，$P(K+1)$也成立，根据数学归纳法，该命题得证。

5.4.4 构造性证明

1. 存在性证明

存在一个 x 使命题 $P(x)$成立可表示为：$\exists xP(x)$。对形如$\exists xP(x)$的命题的证明称为存在性证明。

2. 构造性证明

通过找出一个使得命题 $P(a)$为真的元素 a，从而完成该函数值的存在性证明称为构造性证明。构造性证明方法是计算科学中广泛使用的一种证明方法，本章 Armstrong 公理系统的完备性证明就采用了构造性的证明方法。

5.5 递归和迭代

构造性是计算机软硬件系统的最根本特征，而递归和迭代是最具代表性的构造性数学方法，它广泛地应用于计算学科各个领域，理解递归和迭代的基本思想，有助于今后的学习。

递归和迭代密切相关，实现递归和迭代的基础基于以下一个事实。

很多序列项常常可以以这样的方式得到：由 a_{n-1} 得到 a_n，按这样的法则，可以从一个已知的首项开始，有限次地重复做下去，最后产生一个序列，该序列是实现递归和迭代的基础。

5.5.1 递归

1. 递归及其有关概念

递归不仅是数学中的一个重要概念，也是计算技术中重要的概念之一。20 世纪 30 年代，正是可计算的递归函数理论与图灵机、λ 演算和 POST 规范系统等理论一起为计算理论的建立奠定了基础。

在计算技术中，与递归有关的概念有：递归关系、递归数列、递归过程、递归算法、递归程序、递归方法。

（1）递归关系指的是：一个数列的若干连续项之间的关系。

（2）递归数列指的是：由递归关系所确定的数列。

（3）递归过程指的是：调用自身的过程。

（4）递归算法指的是：包含递归过程的算法。

（5）递归程序指的是：直接或间接调用自身的程序。

（6）递归方法（也称递推法），是一种在“有限”步骤内，根据特定的法则或公式对一个或多个前面的元素进行运算，以确定一系列元素（如数或函数）的方法。

2. 递归与数学归纳法

递归是一个重要的概念，然而，对于刚入大学的同学来说，理解起来却有一定的困难，为了更好地理解递归思想，我们先给出一个简单的例子。

例 5.13　计算 5×6

计算方法之一：6，6+6=12，12+6=18，18+6=24，24+6=30；

计算方法之二：5×6，4×6，3×6，2×6，1×6；1×6+6=12，12+6=18，18+6=24，24+6=30；

方法之二从 5×6 开始计算，假设一个刚学乘法的小学生计算不出这个数，那么，这个小学生一般会先计算 4×6，然后再加 6 就可以了，若仍计算不出，则会再追溯到 3×6，直到 1×6，然后，再依次加 6，最后得到 30。这种计算方法其实就反映了一种递归的思想，这个例子还可以用更一般的递归关系表示：

a_n=C$a_{n\text{-}1}$+$g(n)$，　$n=2$，3，4，…

其中，C 是已知常数，$\{g(n)\}$是一个已知数列。如果已知 $a_{n\text{-}1}$ 就可以确定 a_n。从数学归纳法的角度来看，这相当于数学归纳法归纳步骤的内容。但仅有这个关系，还不能确定这个数列，若使它完全确定，还应给出这个数列的初始值 a_1，这相当于数学归纳法归纳基础的内容。

与数学归纳法相对应，递归由递归基础和递归步骤两部分组成。数学归纳法是一种论证方法，而递归是算法和程序设计的一种实现技术，数学归纳法是递归的基础。

3. 递归的定义功能

递归不仅应用于算法和程序设计之中，它还广泛地应用于定义序列、函数和集合等各个方面。下面，举例说明之。

（1）定义序列

例 5.14　现有序列：2，5，11，23，…，$a_n=2a_{n-1}+1$，…，请给出其递归定义。

解　该序列的递归定义如下：

$a_1=2$；　　递归基础

$a_n=2a_{n-1}+1$，n=2，3，4，…；　　递归步骤

（2）定义函数

例 5.15　给出阶乘 $F(n)=n!$的递归定义

解　阶乘 $F(n)=n!$的递归定义如下：

$F(0)=1$；　　递归基础

$F(n)=n\times F(n-1)$，n=1，2，3，…；　　递归步骤

（3）定义集合

例 5.16　现有文法 G 的生成式如下：

$S\rightarrow 0A1$；　　S 是文法 G 的开始符号

$A\rightarrow 01$；　　递归基础

$A\rightarrow 0A1$；　　递归步骤

以上这个文法的生成式采用了递归方法，该文法其实定义了这样一个集合：$L(G)=\{0^n1^n \mid n\geq 1\}$，这是一个以相同个数的“0”和“1”组成的字符串的集合，即一种特殊的语言。以后，我们将学习到该语言可以由多种文法产生（如 0 型文法、2 型文法等），而图灵机与 0 型文法相对应，因此，图灵机可以识别该语言。

4. 阿克曼函数

在递归函数论和涉及集合的并的某些算法的复杂性研究中，有一个起重要作用的递归函数——阿克曼（Ackermann）函数，该函数是由希尔伯特的学生、德国著名数学家威尔海姆 • 阿克曼于 1928 年发现的。这是一个图灵机可计算的，但不是原始递归的函数。下面，我们介绍这个经典的递归函数，并给出相应的计算过程。

阿克曼函数：

$$A(m,n)=\begin{cases} n+1 & \text{若 } m=0 \\ A((m-1),1) & \text{若 } n=0 \\ A(m-1,A(m,n-1)) & \text{若 } m,n>0 \end{cases}$$

解阿克曼函数的递归算法:

```
Begin
if m=0 then n+1
else if n=0 then A(m-1,1)
else A(m-1,A(m,n-1))
End
```

例 5.17　计算 $A(1,2)$

解: $A(1,2)=A(0,A(1,1))$

$=A(0,A(0,A(1,0))$

$=A(0,A(0,A(0,1))$

$=A(0,A(0,2))$

$=A(0,3)$

$=4$

5.5.2　迭代

“迭”是屡次和反复的意思，“代”是替换的意思，合起来，“迭代”就是反复替换的意思。在程序设计中，为了处理重复性计算的问题，最常用的方法就是迭代方法，主要是循环迭代。

迭代与递归有着密切的联系，甚至，一类如 $X_0=a$，$X_{n+1}=f(n)$的递归关系也可以看作是数列的一个迭代关系。可以证明，迭代程序都可以转换为与它等价的递归程序，反之，则不然。就效率而言，递归程序的实现要比迭代程序的实现耗费更多的时间和空间。因此，在具体实现时，又希望尽可能将递归程序转化为等价的迭代程序。

在第 4 章中，我们给出了一个斐波那契数的迭代算法，显然，就斐波那契数的求解算法而言，可以使用迭代方法或递归方法来解决。然而，一些递归算法，如求解梵天塔问题的算法就不能用迭代方法，而只能用递归方法。

5.6 公理化方法

5.6.1 理论体系

1. 什么是理论

从数学的角度来说，理论是基本概念、基本原理或定律（联系这些概念的判断）以及由这些概念与原理逻辑推理出来的结论组成的集合，该概念可以形式化的定义为：

T=<C,P,S>，

其中：

（1）T 表示理论；

（2）C 表示基本概念的集合；

（3）P 表示基本原理或定律的集合；

（4）S 表示由这些概念与原理逻辑推理出来的结论组成的集合。

2. 构建理论体系的常用方法

每一个理论都由一组特定的概念和一组特定的命题组成。在一个理论中，基本概念（原始概念）和基本命题（原始命题）必须是明确的，否则就会出现“循环定义”和“循环论证”的严重问题。因此，构建一个理论体系必须采用科学的方法。构建理论体系的常用方法有：公理化方法、逻辑和历史相统一的方法、从抽象上升到具体的方法。本书仅介绍与计算学科密切相关的公理化方法。

5.6.2 公理化方法

1. 公理化方法的概念

公理化方法，我们在第 1 章已作过简单介绍，这是一种构造理论体系的演绎方法，它是从尽可能少的基本概念、公理出发，运用演绎推理规则，推出一系列的命题，从而建立整个理论体系的思想方法。

2. 公理系统的 3 个条件

用公理化构建的理论体系称为公理系统，该公理系统需要满足无矛盾性、独立性和完备性的条件。

（1）无矛盾性。这是公理系统的科学性要求，它不允许在一个公理系统中出现相互矛盾的命题，否则这个公理系统就没有任何实际的价值。

（2）独立性。公理系统所有的公理都必须是独立的，即任何一个公理都不能从其他公理推导出来。

（3）完备性。公理系统必须是完备的，即从公理系统出发，能推出该领域所有的命题。

为了保证公理系统的无矛盾性和独立性，一般要尽可能使公理系统简单化。简单化将使无矛盾性和独立性的证明成为可能，简单化是科学研究追求的目标之一。一般而言，正确的一定是简单的（注意，这句话是单向的，反之不一定成立）。

关于公理系统的完备性要求，自哥德尔发表关于形式系统的“不完备性定理”的论文后，数学家们对公理系统的完备性要求大大放宽了。也就是说，能完备更好，即使不

完备，同样也具有重要的价值。

3. 公理化方法在计算学科中的应用

公理化方法主要用于计算学科理论形态方面的研究。在计算学科各分支领域，如形式语义学、关系数据库理论、分布式代数系统以及计算认知领域的研究中均采用了公理化方法。

5.6.3 实例

例 5.18　正整数的公理化概括

原始概念：1；

原始命题（公理）：任何正整数 n 或者等于 1，或者可以从 1 开始，重复地“加 1”来得到它。

例 5.19　平面几何的公理化概括（欧氏几何）

在欧几里德的《几何原本》中，欧几里德用公理化方法对当时的数学知识（平面几何）作了系统化、理论化的总结。在书中，他以点、线、面为原始概念，以 5 条公设和 5 条公理为原始命题，给出了平面几何中的 119 个定义，465 条命题及其证明，构成了历史上第一个数学公理体系。

原始概念：点、线、面

原始命题（公设和公理）如下：

公设 1：两点之间可作一条直线；

公设 2：一条有限直线可不断延长；

公设 3：以任意中心和直径可以画圆；

公设 4：凡直角都彼此相等；

公设 5：在平面上，过给定直线之外的一点，存在且仅存在一条平行线，即所谓的“平行公设（公理）”。

公理 1：等于同量的量彼此相等；

公理 2：等量加等量，和相等；

公理 3：等量减等量，差相等；

公理 4：彼此重合的图形全等；

公理 5：整体大于部分。

5.7 形式化方法

5.7.1 具体公理系统和抽象公理系统

在欧氏几何公理系统中，原始概念（点、线、面）和所有的公理都有直观的背景或客观的意义，像这样有现实世界背景的公理系统，一般被称为具体公理系统。由于非欧几何的出现，使人们感到具体公理系统过于受直觉的局限。因而，在 19 世纪末和 20 世纪初，一些杰出的数学家和逻辑学家开始了对抽象公理系统的研究。

在抽象公理系统中，原始概念的直觉意义被忽视，甚至没有任何预先设定的意义，而公理也无需以任何实际意义为背景，它们无非是一些形式约定的符号串。这时，抽象

公理系统可以有多种解释。例如，形式化的运算规则 1+1 可以解释为一个苹果加一个苹果，或者为一本书加一本书；布尔代数抽象公理系统可以解释为有关命题真值的命题代数，或者有关电路设计研究的开关代数。在这些解释中，抽象符号 X 可以分别被看作“命题 X”、“电路 X”等，“0”和“1”分别被用于表示命题的“假”和“真”，或者电路的“开”和“闭”。

5.7.2 形式化方法

1. 形式化方法

王元元教授在他的著作《计算机中的逻辑学》中，将形式化方法定义为彻头彻尾的“符号化+抽象公理化”。本书所指的形式化方法特指这一含义。

2. 形式系统的组成部分

形式系统由下面几个部分组成：

（1）初始符号。初始符号不具有任何意义。

（2）形式规则。形式规则规定一种程序，借以判定哪些符号串是本系统中的公式，哪些不是。

（3）公理。即在本系统的公式中，确定不加推导就可以断定的公式集。

（4）变形规则。变形规则亦称演绎规则或推导规则。变形规则规定，从已被断定的公式，如何得出新的被断定公式。被断定的公式又称为系统中的定理。

在以上 4 个组织部分中，前两个部分定义了一个形式语言，后两个部分在该形式语言上定义了一个演绎结构。形式系统由形式语言和定义于其上的演绎结构组成。

在计算机系统中，它的软硬件都是一种形式系统，它们的结构也可以用形式化方法描述，程序设计语言更是不折不扣的形式语言系统。

3. 形式系统的基本特点

（1）严格性

形式系统中，初始符号和形式规则都要进行严格的定义，不允许出现在有限步内无法判定的公式。形式系统采用的是一种纯形式的机械方法，它的严格性高于一般的数学推导。

（2）抽象性

抽象性不是形式系统的专利，抽象是人们认识客观世界的基本方法，只不过形式系统具有更强的抽象性。

形式系统的抽象性表现在它自身仅仅是一个符号系统，除了表示符号间的关系（字符号串的变换）外，不表示任何别的意义。

4. 形式系统的局限性

一个形式系统，如果它是无矛盾的，那么，它就具有下面两个局限性：

（1）不完备性

1931 年，哥德尔提出的关于形式系统的“不完备性定理”指出：如果一个形式的数学理论是足够复杂的（复杂到所有的递归函数在其中都能够表示），而且它是无矛盾的，那么在这一理论中存在一个语句，而这一语句在这一理论中是既不能证明，也不能否证的。

（2）不可判定性

如果对一类语句 C 而言，存在一个算法 AL，使得对 C 中的任一语句 S 而言，可以利用算法 AL 来判定其是否成立，则 C 称为可判定的，否则，称为不可判定的。

著名的“停机问题”就是一个不可判定性的问题。该问题是指，一个任给的图灵机对于一个任给的输入而言是否停机的问题。图灵证明这类问题是不可判定的。

需要指出的是：计算机系统就是一种形式系统，因此，计算机系统一样也具有形式系统的局限性。

5. 形式化与公理化

形式化不一定导致公理化，公理系统也不一定是形式系统。如欧氏几何公理系统就不是形式系统。形式化与公理化虽然不同，但在近代数学中，形式系统大都是形式化的公理系统。

5.8 一个实例~~Arms~~trong公理系统*

为便于读者理解计算学科中的公理化思想，我们给出一个数据库领域中有关模式分解的重要的公理系统—Armstrong 公理系统及其有关的预备知识。

5.8.1 预备知识

1. 定义

定义 1：设 $R=<U, F>$是一个关系模式，其中，R 是关系名，U 为组成该关系属性名的集合，F 为 R 上的一个函数依赖关系的集合。设 X，$Y \subseteq U$，$r \in Ins(R)$，则对任何 u，$v \in r$，只要 $u[X]=v[X]$，就必有 $u[Y]=v[Y]$，则称 r 满足 $X \rightarrow Y$，也称 r 中存在函数依赖 $X \rightarrow Y$。

注：关系模式 $R=<U, F>$上的所有实例的集合记作 $Ins(R)$。

定义 2：在关系模式 $R=<U, F>$中，为 F 所逻辑蕴含的函数依赖全体的集合称为 F 的闭包（Closure），记为 F^+。

定义 3：在关系模式 $R=<U, F>$中，若 $X \subseteq U$，则 $X^+=\{A \mid X \rightarrow A$ 能从 F 出发，由 Armstrong 公理系统推导出来$\}$，X^+称为属性集 X 关于函数依赖集 F 的闭包。

2. 算法

在实际工作中，一般不直接计算 F^+，而是通过计算 X^+而达到同样的目的。下面，给出计算属性闭包的算法和一个例子。

算法：在关系模式 $R=<U, F>$中，求属性集 $X(X \subseteq U)$关于函数依赖 F 的属性闭包 X^+。

输入：关系模式 $R=<U, F>$中的全部属性集 U，在 U 上的函数依赖 F，U 的子集 X。

输出：属性闭包 X^+。

步骤：

（1）令 $X^{(0)}=X$，$i=0$

（2）令 $X^{(i+1)}=X^{(i)} \cup \{A \mid V \subseteq X^{(i)}$ ，$V \rightarrow W \in F$，$A \in W\}$

（3）判断等式 $X^{(i+1)}=X^{(i)}$是否成立，若成立则转（4），否则转（2）

（4）令 $X^{+}=X^{(i)}$，输出 X^{+}。

例 5.20　F 由下列 5 个函数依赖组成：

F=｛A→C，B→C，AB→D，BC→DE，D→E｝

计算：$(BD)^{+}$

（1）$X^{(0)}=BD$

（2）$X^{(1)}=BD\cup C$，即 $X^{(1)}=BDC$（在此，将属性集 BD 和 C 的并集 $BD\cup C$ 简记为 BDC，下同），则 $X^{(1)}\neq X^{(0)}$，继续

（3）$X^{(2)}=BDC\cup E$，则 $X^{(2)}\neq X^{(1)}$，继续

（4）$X^{(3)}=BDCE$，则 $X^{(3)}=X^{(2)}$，所以$(BD)^{+}=X^{(2)}=BDCE$。

5.8.2　Armstrong 公理系统

1974 年，Armstrong 在总结各种推理规则的基础上，将以下 3 个推理规则作为函数依赖的 3 条公理，提出了函数依赖公理系统的思想，并给出了该公理系统正确性和完备性的证明。

1. Armstrong 公理系统的 3 条公理

设 U 为关系模式 $R<U，F>$的属性名集合，F 是 U 上的一组函数依赖的集合。对关系模式 $R=<U，F>$而言有以下推理规则：

（1）若 $Y\subseteq X\subseteq U$，则 $X\rightarrow Y$（自反律 Reflexivity）

（2）若 $X\rightarrow Y$，且 $Z\subseteq U$，则 $XZ\rightarrow YZ$（增广律 Augmentation）

（3）若 $X\rightarrow Y$ 及 $Y\rightarrow Z$，则 $X\rightarrow Z$（传递律 Transitivity）

以上 3 条公理构成了函数依赖的一个有效而完备的公理系统，该公理系统是数据库技术中模式分解算法的理论基础。

2. Armstrong 公理系统的正确性和完备性要求

建立函数依赖公理系统的目的在于有效而准确地计算函数依赖的逻辑蕴含，即从已知的函数依赖推出未知的函数依赖。这里有两个要求，一个要求是用公理系统推出的所有函数依赖都是正确的；另一个要求是保证每一个函数依赖都能够用该公理系统推出，如果 F^{+}中居然存在一个函数依赖不能用这些公理推出，那么这些公理就不够用，就不完全，就得补充新的公理。

显然，以上第一个要求是要证明公理系统的正确性，第二个要求是要证明公理系统的完备性。

3. Armstrong 公理系统的正确性证明

证明：分别证明自反律、增广律和传递律的正确性。

（1）设 $r\in Ins(R)$，t，$s\in r$，X，$Y\subseteq U$，$Y\subseteq X$

若 t[X]=s[X]，根据条件 Y ⊆X，则必有 t[Y]=s[Y]。

根据函数依赖的定义，若 $t[X]=s[X]$，必有 $t[Y]=s[Y]$，则 $X\rightarrow Y$ 成立(自反律得证)。

（2）设 $r\in Ins(R)$，t，$s\in r$，X，Y，$Z\subseteq U$，$X\rightarrow Y$

若 $t[XZ]=s[XZ]$，则有 $t[X]=s[X]$及 $t[Z]=s[Z]$　　…… ①。

再根据条件 $X\rightarrow Y$ 以及 $t[X]=s[X]$，则必有 $t[Y]=s[Y]$…… ②。

由①、②可知，$t[Y]=s[Y]$，$t[Z]=s[Z]$，即 $t[YZ]=s[YZ]$。

根据函数依赖的定义，若 $t[XZ]=s[XZ]$，必有 $t[YZ]=s[YZ]$，则 $XZ\rightarrow YZ$ 成立（增广律得证）。

（3）设 $r\in Ins(R)$，t，$s\in r$，X，Y，$Z\subseteq U$，$X\rightarrow Y$，$Y\rightarrow Z$

若 $t[X]=s[X]$，根据条件 $X\rightarrow Y$，则必有 $t[Y]=s[Y]$。

再根据条件 $Y\rightarrow Z$ 及 $t[Y]=s[Y]$，则必有 $t[Z]=s[Z]$。

根据函数依赖的定义，若 $t[X]=s[X]$，必有 $t[Z]=s[Z]$，则 $X\rightarrow Y$ 成立（传递律得证）。

4. Armstrong 公理系统的完备性证明

函数依赖公理系统的公理完备性还可以这样理解：所有从 F 出发，用公理系统不能推出的函数依赖都不为真，即它不被 F^+蕴含，或者说，存在一个具体关系 r，若 $X\rightarrow Y$ 不能从 F 出发用 Armstrong 公理系统推出，则 $X\rightarrow Y$ 在 r 中不成立，即 F^+不能逻辑蕴含 $X\rightarrow Y$。

首先，我们构造一个特定的关系 r，如表 5.1 所示，该表只包含两个元组。

表 5.1　　关系 r

X^+ 的	$(U-X^+)$的属性
$a_1a_2a_3$ …… a_m	$a_{m+1}a_{m+2}a_{m+3}$ …… a_n
$a_1a_2a_3$ …… a_m	$b_{m+1}b_{m+2}b_{m+3}$ …… b_n

设 r 中第一个元组为 $u1$，第二个元组为 $u2$，根据 r 的构造，这两个元组在 X^+的属性上取相同的值，在其他属性，即$(U-X^+)$上取不相同的值。

针对关系 r，如果我们能证明以下两点，则 Armstrong 公理系统的完备性也就证明了。

（1）r 必是 $R<U$，$F>$的一个关系，即 F 中的全部函数依赖在 r 上都成立。

（2）在 r 中，若 $X\rightarrow Y$ 从 F 出发，不能用 Armstrong 公理系统导出，则 $X\rightarrow Y$ 在 r 中不成立。

下面，分别证明这两点。

（1）设 $V\rightarrow W$ 为 F^+中任一函数依赖，分以下两种情况进行讨论：

① 若 $V\subseteq X^+$，则 $X\rightarrow V$，并从 r 中可知 $u1[V]=u2[V]$，再根据假设 $V\rightarrow W$，用 Armstrong 公理系统的第三条公理（传递律）可得 $X\rightarrow W$。

$\because X\rightarrow W$，　$\therefore W\subseteq X^+$，则 $u1[W]=u2[W]$。

根据函数依赖的定义，在 r 中，$V\rightarrow W$ 成立。

② 若 $V\not\subset X^+$，则 $u1[V]\neq u2[V]$。根据函数依赖的定义，在 r 中 $V\rightarrow W$ 也成立。

因此，在关系 r 中，F^+中的任一函数依赖都成立。

（2）在 r 中，若 $X\rightarrow Y$ 由 F 出发，不能用 Armstrong 公理系统推出，则由 r 的构造可知 $Y\not\subset X^+$。因此，必有 Y 的子集 Y'满足 $Y'\subseteq U-X^+$，则在 r 中，若 $u1[X]=u2[X]$，而 $u1[Y]\neq u2[Y]$，即在关系 r 中 $X\rightarrow Y$ 不成立。

因此，凡是从 F 出发，不能用 Armstrong 公理系统推出的函数依赖，在 r 中都不成立，即不被 F^+所蕴含。换言之，凡是 F^+所蕴含的函数依赖，从 F 出发，都能用 Armstrong 公理系统推出。从而 Armstrong 公理系统的完备性得到证明。

思 考 题

1．数学有哪些基本特征？
2．数学方法有什么作用？
3．什么叫集合？集合的基本运算有哪几种？
4．什么是函数？
5．什么是关系？等价关系要满足哪些条件？
6．并发关系是否是等价关系?试举例说明。
7．什么是字符串？什么是语言？语言、文法与自动机有何关系？
8．什么是布尔值？布尔运算的定义？
9．数学方法中有哪几种证明方法？
10．请用伪代码给出求解斐波那契数的递归算法。
11．求下列阿克曼函数值：
（1）$A(0,1)$　（2）$A(1,0)$　（3）$A(1,1)$　（4）$A(2,1)$　（5）$A(2,2)$
12．什么是递归和迭代？二者有何联系？
13．给出“理论”的形式化定义。
14．构建理论体系的常用方法有哪些？
15．简述公理系统的 3 个条件。
16．分别对正整数、平面几何（欧氏几何）进行公理化概括。
17．什么是形式化方法？形式化系统由哪几部分组成？

第 6 章　计算学科中的系统科学方法

系统科学方法是指用系统的观点来认识和处理问题的各种方法的总称，它是一般科学方法论中的重要内容。系统科学方法为现代科学技术的研究带来了革命性的变化，并在社会、经济和科学技术等各个方面都得到了广泛的应用。

模型方法是系统科学的基本方法，研究系统具体来说就是研究它的模型。模型是对系统原型的抽象，是科学认识的基础和决定性环节。

模型与实现是认识与实践的一种具体体现，在计算学科中，它反映了抽象、理论和设计 3 个过程的基本内容。模型与实现包括建模、验证和实现 3 方面的内容。其中，建模主要属于学科抽象形态方面的内容，模型的验证主要属于学科理论形态方面的内容，而模型的实现则主要属于学科设计形态方面的内容。

本章主要介绍学科中有关系统建模（抽象）和实现（设计）两方面的内容，至于模型的验证（理论），本书不加赘述，有兴趣的读者可参阅其他专业文献。

为了更好地理解计算学科中的系统科学方法，下面首先介绍系统科学的基本思想，然后，介绍计算学科中最常用的两种系统科学方法，即结构化方法和面向对象方法。

6.1　系统科学的基本思想

系统科学起源于人们对传统数学、物理学和天文学的研究，诞生于 20 世纪 40 年代。系统科学的崛起被认为是 20 世纪现代科学的两个重大突破性成就之一。

建立在系统科学基础之上的系统科学方法开辟了探索科学技术的新思路，它是认识、调控、改造和创造复杂系统的有效手段，它为系统形式化模型的构建提供了有效的中间过渡模式。现代计算机普遍采用的组织结构，即冯•诺依曼型计算机组织结构就是系统科学在计算技术领域所取得的应用成果之一。今天，随着计算技术的迅猛发展，计算机软硬件系统变得越来越复杂，因此，系统科学方法在计算学科中的作用也日显重大。

6.1.1　系统科学的基本概念

系统科学是探索系统的存在方式和运动变化规律的学问，是对系统本质的理性认识，是人们认识客观世界的一个知识体系。计算学科中一些重要的系统方法，如结构化方法、面向对象方法都沿用了系统科学的思想方法。如何更好地借鉴系统科学的思想方法，是计算科学界应引起重视的问题，而了解系统科学的基本概念和方法是我们自觉运用系统科学方法的基础。

1. 系统（System）和子系统（Subsystem）

系统是指由相互联系、相互作用的若干元素构成的，具有特定功能的统一整体。系统可以形式化地定义为：$S=<A, R>$

其中：

A 表示系统 S 中所有元素的集合；

R 表示系统 S 中所有元素之间关系的集合。

一个大的系统往往是复杂的，它通常可以划分为一系列较小的系统，这些系统称为子系统。子系统可以形式化地定义为：$S_i=<A_i, R_i>$

其中：

$S_i \subset S$；

$A_i \subset A$；

$R_i \subset R$。

2. 结构（Structure）和结构分析（Structure Analysis）

所谓结构是指系统内各组成部分（元素和子系统）之间相互联系、相互作用的框架。

结构分析的重要内容就是划分子系统，并研究各子系统的结构以及各子系统之间的相互关系。

3. 层次（Hierarchy）和层次分析（Hierarchy Analysis）

层次是划分系统结构的一个重要工具，也是结构分析的主要方式。系统的结构可以表示为各级子系统和系统要素的层次结构形式。一般来说，在系统中，高层次包含和支配低层次，低层次隶属和支撑高层次。明确所研究的问题处在哪一层次上，可以避免因混淆层次而造成的概念混乱。

层次分析的主要内容有：系统是否划分层次，划分了哪些层次，各层次的内容，层次之间的关系以及层次划分的原则等。

4. 环境（Environment）、行为（Behavior）和功能（Function）

系统的环境是指一个系统之外的一切与它有联系的事物组成的集合。系统要发挥它应有的作用，达到应有的目标，系统自身一定要适应环境的要求。

系统的行为是指系统相对于它的环境所表现出来的一切变化。行为属于系统自身的变化，同时又反映环境对系统的影响和作用。

系统的功能是指系统行为所引起的、有利于环境中某些事物乃至整个环境存在与发展的作用。

5. 状态（State）、演化（Evolution）和过程（Process）

状态是系统科学中的基本概念之一，它是指系统的那些可以观察和识别的形态特征。状态一般可以用系统的定量特征来表示，如温度 T、体积 V 等。

演化是指系统的结构、状态、特征、行为和功能等随着时间的推移而发生的变化。系统的演化性是系统的基本特性。

过程是指系统的演化所经过的发展阶段，它由若干子过程组成。过程的最基本元素是动作，动作不能再分。

6.1.2 系统科学遵循的一般原则

1. 整体性原则

整体性原则，是基于系统要素对系统的非还原性或非加和性关系（注：非还原性是指系统的整体具有但还原为部分便不存在的特性，非加和性是指

整体不能完全等于各部分之和），是系统方法的根据和出发点。这一原则要求人们在研究系统时，应从整体出发，立足于整体来分析其部分以及部分之间的关系，进而达到对系统整体的更深刻的理解。

在讲到系统的整体性时，我们要谈到“涌现性”（Emergent Property）一词。系统科学把整体具有而部分不具有的东西（即新质的涌现），称为“涌现性”。从层次结构的角度看，涌现性是指那些高层次具有而还原到低层次就不复存在的属性、特征、行为和功能。

简单地借用亚里士多德的名言“整体大于部分之和”来表述整体涌现性是不够的。在某些特殊情况下，当部分构成整体时，出现了部分所不具有的某些性质，同时又可能丧失了组成部分单独存在时所具有的某些性质。这个规律叫做“整体不等于部分之和”原理，也称为“贝塔朗菲定律”。系统的整体功能是否大于或小于部分功能之和，关键取决于系统内部诸要素相互联系、相互综合的方式如何。

2. 动态原则

动态原则是指系统总是动态的，永远处于运动变化之中。人们在科学研究中经常采用理想的“孤立系统”或“闭合系统”的抽象，但在实际中，系统无论是在内部各要素之间，还是在内部环境与外部环境之间，都存在着物质、能量及信息的交换和流通。因此，实际系统都是活系统，而非静态的死系统、死结构。我们在研究系统时，应从动态的角度去研究系统发展的各个阶段，以准确把握其发展过程及未来趋势。

3. 最优化原则

亦称整体优化原则，就是运用各种有效方法，从系统多种目标或多种可能的途径中选择最优系统、最优方案、最优功能、最优运动状态，达到整体优化的目的。

4. 模型化原则

模型化原则就是根据系统模型说明的原因和真实系统提供的依据，提出以模型代替真实系统进行模拟实验，达到认识真实系统特性和规律性的方法。模型化方法是系统科学的基本方法。

系统科学研究主要采用的是符号模型而非实物模型。符号模型包括概念模型、逻辑模型、数学模型，其中最重要的是数学模型。数学模型是指描述元素之间、子系统之间、层次之间以及系统与环境之间相互作用的数学表达式，如树结构、图、代数结构等。数学模型是系统定性和定量分析的工具。研究系统的模型化方法，通常是指通过建立和分析系统的数学模型来解决问题的方法和程序。

用计算机程序定义的模型称为基于计算机的模型（Computer-Based Model）。所有数学模型均可转化为基于计算机的模型，并通过计算来研究系统。而一些复杂的、无法建立数学模型的系统，如生物、社会和行为过程等，也可建立基于计算机的模型。计算实验对一些无法用真实实验来检验的系统是惟一可行的检验手段。

6.1.3 常用的几种系统科学方法

1. 系统分析法

系统分析法是以运筹学和计算机为主要工具，通过对系统各种要素、过程和关系的考察，确定系统的组成、结构、功能、效用的方法。系统分析法广泛应用于计算机硬件的研制和软件的开发，技术产品的革新，环境科学和生态系统的研究，以及城市管理规划等方面。

2. 信息方法

信息方法是以信息论为基础，通过获取、传递、加工、处理、利用信息来认识和改造对象的方法。

3. 功能模拟方法

功能模拟方法是以控制论为基础，根据两个系统功能的相同或相似性，应用模型来模拟原型功能的方法。

4. 黑箱方法

黑箱是指内部要素和结构尚不清楚的系统。黑箱方法就是通过研究黑箱的输入和输出的动态系统，确定可供选择的黑箱模型进行检验和筛选，最后推测出系统内部结构和运动规律的方法。

5. 整体优化方法

整体优化方法是指从系统的总体出发，运用自然选择或人工技术等手段，从系统多种目标或多种可能的途径中选择最优系统、最优方案、最优功能、最优运动状态，使系统达到最优化的方法。

6.1.4 实例

为便于理解系统的基本概念，下面，举几个例子进行说明。

例 6.1　科学的分类

根据科学知识本质特征的不同，我国著名科学家钱学森开创性地将科学划分为工程技术、技术科学、基础科学和哲学 4 个层次。

4 个科学层次是相互联系、相互作用的。其中，工程技术泛指一切应用和技术领域，技术科学是为工程技术提供工程理论的科学；基础科学是揭示客观世界运动规则和本质关系的科学，哲学是对科学知识总的概括，是最高一层的科学。

例 6.2　生命系统

美国心理学家米勒（S.Miller）把生物圈看作是一个生命系统，他认为一切活着的具体系统都是“生命系统”，并将生命系统划分为 7 个层次，即细胞、器官、生物体、群体、组织、社会和超国家系统，以及 19 个关键的子系统。20 世纪 50 年代，米勒创立了一般生命系统理论，该理论对解决生命世界的统一性问题有十分重要的意义。

例 6.3　化学元素周期表

进入 19 世纪后，由于化学分析方法的改进，到 1869 年，人们已经发现了 63 种化学元素。随着新元素发现的增加，以及对这些元素性质的更多了解，人们反而对眼前纷繁复杂的化学世界产生了一种迷惑：难道世界上的化学物质就是这样杂乱无章地凑到一起的吗？

为了寻找化学元素之间的内在联系，许多科学家开始致力于这方面的探索。1869年3月，俄国化学家门捷列夫发表了《元素属性和原子量的关系》的论文，首创了化学元素周期表，揭示了化学元素性质呈周期性变化的内在规律，并指明了发现新元素的方向。

化学元素周期表的建立，使化学科学走上了系统化的道路，成为化学发展的主要基石之一。

例 6.4　整数

当把整数看作是一个系统时，根据等价关系，可以将整数划分为若干互不相交的子集。

如可以将整数划分为奇数和偶数。再比如，若以3为模，可将非负整数 S 划分为下面3类具有同余关系（同余关系是一种等价关系）的集合 $S1$、$S2$ 和 $S3$。

若余数为0，则具有同余关系的数据构成第一个集合：$S1=\{0，3，6，\cdots，3n，\cdots\}$。

若余数为1，则具有同余关系的数据构成第二个集合：$S2=\{1，4，7，\cdots，3n+1，\cdots\}$。

若余数为2，则具有同余关系的数据构成第三个集合：$S3=\{2，5，8，\cdots，3n+2，\cdots\}$。

以上整数的划分，体现出了集合论中等价关系（满足自反性、对称性和传递性的关系）的一个重要性质，即将“整数”推广为更一般性的“元素”时，只要元素之间的关系为等价关系，则可将这些元素组成的集合划分为若干互不相交的子集。等价关系的这种性质具有重要的理论和应用价值。

例 6.5　计算机网络

计算机网络是计算机系统中一个有代表性的复杂系统，需要高度协调的工作才能保证系统的正常运行。为此，必须精确定义网络中数据交换的所有规则（网络协议），然而由这些规则组成的集合却相当庞大和复杂。

为了解决复杂网络协议的设计问题，国际标准化组织（ISO）采用系统科学的思想，定义了现在被广泛使用的开放系统互连模型（Open System Interconnection，简称OSI），该模型将整个网络协议划分为7个层次，即物理层、数据链路层、网络层、运输层、会话层、表示层和应用层，从而有效地降低了网络协议的复杂性，推动了网络技术的发展。

6.2　结构化方法

结构化方法（Structured Methodology）是计算学科的一种典型的系统开发方法。它采用了系统科学的思想方法，从层次的角度，自顶向下地分析和设计系统。结构化方法包括结构化分析（Structured Analysis，简称SA）、结构化设计（Structured Design，简称SD）和结构化程序设计（Structured Program Design，简称SP）三部分内容。其中，SA和SD主要属于学科抽象形态的内容，SP则主要属于学科设计形态方面的内容。

在结构化方法中，有两大类典型方法，一类是以Yourdon的结构化设计、Gane/Sersor结构化分析方法以及Demarco结构化分析方法为代表的面向过程（面向数据流）的方法；另一类是以Jackson方法和Warnier-Orr方法为代表的面向数据结构的方法。

6.2.1　结构化方法的产生和发展

1. 结构化程序设计方法的形成

结构化方法起源于结构化程序设计语言。在使用 SP 之前，程序员都是按照各自的习惯和思路来编写程序，没有统一的标准，这样编写的程序可读性差，更为严重的是程序的可维护性极差，经过研究发现，造成这一现象的根本原因是程序的结构问题。

1966 年，C.BÖhm 和 G.Jacopini 提出了关于“程序结构”的理论，并给出了任何程序的逻辑结构都可以用顺序结构、选择结构和循环结构来表示的证明。在程序结构理论的基础上，1968 年，戴克斯特拉提出了“GOTO 语句是有害的”的问题，并引起普遍重视，SP 逐渐形成，并成为计算机软件领域的重要方法，对计算机软件的发展具有重要的意义。伴随着 SP 的形成，相继出现了 Modula-2、C 以及 Ada 等结构化程序设计语言。

2. 结构化设计方法的形成

结构化程序设计需要事先设计好每一个具体的功能模块，然后将这些设计好的模块组装成一个软件系统。接下来的问题是，如何设计模块。

源于结构化程序设计思想的结构化设计方法就是要解决模块的构建问题。1974 年，W.Stevens、G.Myers 和 L.Constantine 等人在《IBM 系统》（*IBM System*）杂志上发表了《结构化设计》（*Structured Design*）论文，为结构化设计方法奠定了思想基础。此后这一思想不断发展，最终成为一种流行的系统开发方法。

3. 结构化分析方法的形成

结构化设计方法建立在系统需求明确的基础上。如何明确系统的需求，就是结构化分析所要解决的问题。结构化分析方法产生于 20 世纪 70 年代中期，最初的倡导者有 Tom Demarco、Ed Yourdon 等人。结构化分析在 20 世纪 80 年代又得到了进一步的发展，并随着 Ed Yourdon 于 1989 年所著的《现代结构化分析》（*Modern Structured Analysis*）的出版而流行开来。现代结构化分析更强调建模的重要性。

6.2.2 结构化方法遵循的基本原则

结构化方法的基本思想就是将待解决的问题看作一个系统，从而用系统科学的思想方法来分析和解决问题。结构化方法遵循以下基本原则。

（1）抽象原则

抽象原则是一切系统科学方法都必须遵循的基本原则，它注重把握系统的本质内容，而忽略与系统当前目标无关的内容，它是一种基本的认知过程和思维方式。

（2）分解原则

分解原则是结构化方法中最基本的原则，它是一种先总体，后局部的思想原则。在构造信息系统模型时，它采用自顶向下，分层解决的方法。

（3）模块化原则

模块化是结构化方法最基本的分解原则的具体应用，它主要出现在结构化设计阶段中，其目标是将系统分解成具有特定功能的若干模块，从而完成系统指定的各项功能。

6.2.3 结构化方法的核心问题

模型问题是结构化方法的核心问题。建立模型（简称建模）是为了更好地理解我们要模拟的现实世界。建模通常是从系统的需求分析开始，在结构化方法中，就是使用SA 方法构建系统的环境模型；然后使用 SD 方法，确定系统的行为和功能模型；最后使用 SP 方法，进行系统的设计，并确定用户的现实模型。

1. 环境模型

SA 的主要任务就是要完成系统的需求分析，并构建现实世界的环境模型。在结构化方法中，环境模型包括需求分析、环境图和事件列表等内容。

（1）需求分析

需求分析是系统分析的第一步，它的主要任务是明确用户的各种需求，并对系统要做什么作一个清晰、简洁和无二义性的文档说明。

需求分析阶段的用户一般是高级主管、人事主管和执行官，且基本上每个人都不直接参与新系统的开发。

（2）环境图

环境图是数据流图的一种特殊形式。环境图模拟系统的一个大致边界，并展示系统和外部的接口、数据的输入和输出以及数据的存储。

（3）事件列表

事件列表是发生在外部世界，但系统必须响应的叙述性列表。事件列表是对环境图的一个补充。

2. 行为和功能模型

SD 的主要任务就是要在系统环境模型的基础上建立系统的行为和功能模型，完成系统内部行为的描述。实现系统行为和功能模型的主要工具有：数据字典、数据流图、状态变迁图和实体-联系模型等。

（1）数据字典

数据字典是一个包含所有系统数据元素定义的仓库。数据元素的定义必须是精确的、严格的和明确的。一个实体一般应包括以下几个部分的内容。

① 名字；

② 别名；

③ 用途；

④ 内容描述；

⑤ 备注信息。

（2）数据流图

数据流图是 SA 和 SD 的核心技术，它采用面向处理过程的思想来描述系统，它是一种描述信息流和数据从输入到输出变换的应用图形技术。

（3）状态变迁图

状态变迁图及时地描述了对象的状态，它着重系统的时间依赖行为。状态变迁图源于实时系统的建模，并被广泛应用于商业信息处理领域中。

状态变迁图看起来非常像数据流图，然而，它们之间却存在着本质的不同。数据流图着重于数据流和数据转换的过程，而状态变迁图着重于状态的描述，如激励发生时的开始状态和系统执行响应后的结果状态。状态变迁图的条件和一个过程的输入数据流相对应，同时，还与控制流的流出相对应。

（4）实体一联系模型

实体一联系模型被用来模拟系统数据部件之间的相互关系。实体一联系模型独立于当前的系统状态，并与具体的计算机程序设计语言无关。

3. 实现模型

SP 的主要任务就是要在系统行为和功能模型的基础上建立系统的实现模型。实现该模型的主要工具有：处理器模型、任务模型以及结构图等。

（1）处理器模型

在多处理器系统和网络环境中，还需要将处理器分成不同的组，以便确定操作在哪个处理器上进行。

（2）任务模型

任务模型建立在处理器模型的基础之上，它将所有过程都划分成操作系统的任务。

（3）结构图

结构图是使用图形符号来描述系统的过程和结构的工具。结构图常由数据流图转换而来，它展示了模块的划分、层次和组织结构以及模块间的通信接口，从而有助于设计者和程序开发人员进行系统的设计。

在结构图中，通常有一个主模块在最高层，并由该主模块启动程序并协调所有的模块。低级模块则包含更详细的功能设计。

（4）模块设计

在结构化方法中，SP 阶段的目标就是将系统分解成更容易实现和维护的模块。SP 方法要求每个模块执行单一的功能，而且不同模块间的依赖性要尽可能低。

（5）实现阶段

实现阶段包括系统的编码、测试和安装。这一阶段的产物主要是能够模拟现实世界的软件系统。除此之外，软件文档和帮助用户熟悉系统的客户培训计划也是这一阶段的产物。

6.3 面向对象方法

面向对象（Object-Oriented，简称 OO）方法是以面向对象思想为指导进行系统开发的一类方法的总称。这类方法以对象为中心，以类和继承为构造机制来抽象现实世界，并构建相应的软件系统。

6.3.1 面向对象方法的产生和发展

1. 面向对象程序设计语言的形成

与结构化方法一样，面向对象方法也起源于面向对象程序语言（Object-Oriented Program Language，简称 OOPL）。面向对象程序语言开始于 20 世纪 60 年代后期，第一个 OOPL 是挪威计算中心的 Kristen Nygaard 和 Ole-Johan Dahl 于 1967 年研制的 Simula 语言，该语言引入了许多面向对象的概念，如类和继承性等。

受 Simula 语言的影响，1972 年，Alan Kay 在 Xerox 公司研制成功了 Smalltalk 语言，

并对面向对象的一些概念作了更精确的定义。1980 年，Xerox 公司推出的 Smalltalk-80 语言标志着 OOPL 进入实用化阶段。

20 世纪 80 年代，OOPL 得到了极大地发展，相继出现了一大批实用的面向对象语言，如 Objective C（1986 年）、C++（1986 年）、Self（1987 年）、Eiffel（1987 年）和 Flavors（1986 年）等。

2. 面向对象设计和面向对象分析的形成

20 世纪 80 年代中期，随着 OOPL 的推广使用，面向对象技术很快被应用到系统分析和系统设计之中。20 世纪 90 年代，面向对象分析（Object-Oriented Analysis，简称 OOA）和面向对象设计（Object-Oriented Design，简称 OOD）开始成熟，一些实用的面向对象开发方法和技术相继出现。如 G.Booch 提出的面向对象开发方法学，P.Coad 和 E.Yourdon 提出的 OOA 和 OOD 等等。这些方法的提出，标志着面向对象方法逐步发展成为一类完整的系统化的技术体系。

6.3.2 面向对象方法的基本思想

《大英百科全书》描述了“分类学理论”中有关人类认识现实世界普遍采用的 3 个构造法则：

（1）区分对象及其属性；

（2）区分整体对象及其组成部分；

（3）形成并区分不同对象的类。

按照 P.Coad 和 E.Yourdon 的论述，面向对象思想正是根据以上 3 个常用的构造法而建立起来的。在实际应用中，它采用对象及其属性，整体和部分，类、成员和它们之间的区别等 3 个法则来对系统进行分析和设计，遵循了分类学理论的基本原理，符合认识来源于实践，又服务于实践的辩证唯物主义思想。

在 OO 方法中，对象和类是其最基本的概念。其中，对象是系统运行时的基本单位，是类的具体实例，是一个动态的概念；而类是对具有相同属性和操作（或称方法、服务）的对象进行的抽象描述，是对象的生成模板，是一个静态的概念。

类可以形式化地定义为：

Class=<ID，INH，ATT，OPE，ITF>

其中：

ID——类名；

INH——类的继承性集；

ATT——属性集；

OPE——操作集；

ITF——接口消息集。

类是数理哲学和逻辑哲学的根本问题，类理论是逻辑方法学、数学方法论和哲学方法论的本体论基础。对“类”的进一步认知，如对类的构成原则、基本特征、种类以及类的关系与类的运算等的认知，有助于理解 OO 方法的实质，感兴趣的读者可参阅赵总宽等人编著的《现代逻辑方法论》一书。

1. 面向对象模型及其特性

面向对象模型是一个可见的、可复审的和可管理的层次模型集，一般认

为面向对象模型应包括下面几个特性：

（1）身份、状态、行为

① 身份是某一对象区别于其他对象的属性。所有的对象都有一个可以相互区别的身份。对象与对象之间相互区别是通过它们固有的独立的个体存在，而不是通过它们的属性来区分的，相同的属性不等于相同的身份（例如两个苹果，尽管有相同的形状、颜色或质地，但仍是两个独立的苹果）。

② 状态是指对象所有属性被附上值所具有的一种情形。

③ 行为是指对象在其状态变化和消息传递过程中的作用及反应，状态可以定义为行为的累积结果，而行为则可改变对象的状态。

（2）分类

分类意味着有相同的数据结构（属性和状态）和行为的对象组成一个类，每个类描述一个类的集合。每个对象都是它的类的一个实例，实例的每个属性都有它自己的值，但是和类的其他实例共享相同的属性名和操作。

（3）继承

继承是指在类中基于层次的关系，共享属性和操作。一个类可以被细化为子类，每个子类继承父类的所有属性，并可以增加它独有的属性。

（4）多态

多态是指相同的操作在不同的类上可以有不同行为的特性。

2. 面向对象模型遵循的基本原则

面向对象模型遵循的基本原则有：抽象、封装、模块化以及层次原则等。

（1）抽象

抽象是处理现实世界复杂性的最基本方式，在 OO 方法中，它强调一个对象和其他对象相区别的本质特性。对于一个给定的域，确定合理的抽象集是面向对象建模的关键问题之一。

（2）封装

封装是对抽象元素的划分过程，抽象由结构和行为组成，封装用来分离抽象的原始接口和它的执行。

封装也称为信息隐藏（Information Hiding），它将一个对象的外部特征和内部的执行细节分割开来，并将后者对其他对象隐藏起来。

（3）模块化

模块化是已经被分为一系列聚集的和耦合的模块的系统特性。对于一个给定的问题，确定正确的模块集几乎与确定正确的抽象集一样困难。通常，每个模块应该足够简单，以便能够被完整地理解。

（4）层次

抽象集通常形成一个层次。层次是对抽象的归类和排序。在复杂的现实世界中有两种非常重要的层次，一个是类型层次，另一个是结构性层次。

确定抽象的层次是基于对象的继承，它有助于在对象的继承中，发现抽象间的关系，搞清问题的所在，理解问题的本质。

6.3.3 面向对象方法的核心问题

面向对象方法与结构化方法一样，其核心问题也是模型问题。面向对象模型主要由OOA 模型、OOD 模型组成。其中，OOA 主要属于学科抽象形态方面的内容，OOD 主要属于学科设计形态方面的内容。

1．OOA 模型

OOA 关心的是构建现实世界的模型问题。如何解决现实世界的建模问题呢？根据系统科学的思想，首先需要对复杂的系统进行分解，最常用的分解方法就是分层。

关于 OOA 模型的分层方法有不少，本书采用 P.Coad 和 E.Yourdon 的分层方法。该方法将 OOA 模型划分为 5 个层次，即主题层、对象层、结构层、属性层和服务层。OOA 的主要任务就是要在问题域上构建具有这 5 个层次内容的 OOA 模型。

（1）主题层

主题给出 OOA 模型中各图的概况，为分析员和用户提供了一个相互交流的机制，有助于人们理解复杂系统的模型构成。

（2）对象层

对象是属性及其专用服务的一个封装体，是对问题域中的人、事和物等客观实体进行的抽象描述。对象由类创建，类是对一个或多个对象的一种描述，这些对象能用一组同样的属性和服务来刻画。

（3）结构层

在 OO 方法中，组装结构和分类结构是两种重要的结构类型，它们分别刻画“整体与部分”组织以及“一般与特殊”组织。

组装结构（即整体与部分）遵循了人类思维普遍采用的第 2 个基本法则，即区分整体对象及其组成部分。

分类结构（即一般与特殊）遵循了人类思维普遍采用的第 3 个法则，在 OO 方法中，就是类、成员和它们之间的区别。

（4）属性层

属性是描述对象或分类结构实例的数据单元，类中的每个对象都具有它的属性值，属性值就是一些状态的信息数据。

（5）服务层

一个服务就是收到一条信息后所执行的处理（操作）。服务是对模型化的现实世界的进一步抽象。

2．OOD 模型

OOA 与 OOD 不存在转换的问题。OOD 根据设计的需要，仅对 OOA 在问题域方面建立的 5 个抽象层次进行必要的增补和调整，同时，OOD 还必须对人机交互、任务管理和数据管理 3 个部分的内容进行抽象，最后建立完整的 OOD 模型。该模型的主要内容可以用表 6.1 所示的形式来概括。

表 6.1　　OOD 模型

抽象层次 / 主要内容	主题层	对象层	结构层	属性层	服务层
1．问题域					
2．人机交互					

3．任务管理					
4．数据管理					

在 OOA 模型中，对象强调问题域，用问题域中的意义来表示事物或概念。在 OOD 中，当对象含有问题域中的意义时，对象被称为语义对象。除了分析以外，OOD 不仅强调系统的静态结构，而且还强调系统行为的动态结构。

3. 支持 OOA 和 OOD 模型的实现问题

使用 OOPL 来实现 OOA 和 OOD 模型相对来说比较容易，因为 OOPL 的构造与 OOA 和 OOD 模型的构造是相似的，OOPL 支持对象、运行多态性和继承等概念。使用非 OO 语言则需要特别注意和规定保留程序的 OO 结构。OO 概念可以映射到非 OO 语言结构中，这只是一个表达方式的问题，不是语言能力的问题，因为编程语言最终要转换为机器语言，对 OO 模型而言，使用 OOPL 效果更好一些。

6.4 小 结

在计算机软件领域，很多新的方法与技术都起源于程序设计语言，并向软件生命周期的前期阶段发展。这种发展趋势具有十分重要的意义，它使那些富有生命力的新方法和新技术就此形成自己系统化的技术体系。本章所介绍的结构化方法和面向对象方法，以及以后我们将要介绍的形式化技术等都遵循这一发展规律，它们极大地推动了计算机软件技术的发展。

本章所介绍的结构化方法和面向对象方法都是计算技术中常用的系统开发方法，两种开发方法目前都非常流行，选择哪一种方法要根据分析者的熟练程度和项目的类型而定。

就目前而言，十全十美的开发方法是不存在的，真正实用的系统开发方法往往是多种开发方法的结合。如何综合应用，这就要根据所开发系统的规模、系统的复杂程度、系统开发方法的特点，以及所能使用的计算机软件等诸多因素综合考虑后决定。

系统科学方法在科学技术方法论中占有重要的地位，如何更好地将系统科学的成果应用于计算学科是值得我们思考的问题。下面，我们给出黑箱方法在计算学科中的一些应用，以起抛砖引玉之用。

目前黑箱方法已在软件测试中得到应用。其实，黑箱方法还完全可以应用于计算机病毒以及计算机加密和解密等领域。这类应用的前提基于以下事实：经计算机加密程序处理的数据，必定存放在计算机存储设备的一个确定范围之内；一般所指的计算机病毒分两类情况被调用，一类是通过修改中断地址而被调用；另一种是通过修改操作系统的文件（例如，Io.sys、Command.com 等），并因该类文件的调用而被调用。由于知道目的地或中间环节，因此无论黑箱多复杂，一般都可以解，至少可以让它失效。例如，解 PC-Lock 的方法和解 Boot 区病毒的方法。

思 考 题

1．什么是系统科学？系统科学应遵循哪些原则？

2．如何正确理解系统的整体涌现性？

3．常用的系统科学方法有哪几种？

4．什么是结构化方法？结构化方法应遵循哪些基本原则？

5．结构化方法中如何建立和实现模型？

6．面向对象思想与“分类学理论”中有关人类认识现实世界普遍采用的 3 个构造法则有什么关系？

7．如何理解面向对象方法中的对象和类？

8．面向对象模型有哪些特性？

9．面向对象模型应遵循哪些基本原则？

10．面向对象模型主要由什么组成？

11．结构化方法和面向对象方法的产生和发展规律有何相同之处？

第 7 章　形式化技术*

本书在第 5 章中介绍了一种适用于计算理论研究的“符号化+抽象公理化”的数学方法——形式化方法。而本书所指的形式化技术则是应用于计算机软硬件系统设计的数学建模技术。

本章主要介绍形式化技术的产生和发展、形式化规格技术、形式化验证技术。

7.1　形式化技术概述

1. 形式化技术的产生和发展

自 20 世纪 60 年代末以来，许多学者开展了形式化技术的相关研究。形式化技术最早是从戴克斯特拉和霍尔（C.Hoare）在程序验证方面的工作和斯科特（D.S.Scott），以及其他学者在程序语义方面的工作发展起来的。现在，形式化技术已经成为计算机科学的一个重要分支和研究领域，其作用相当于传统工程设计（如计算流体动力学（CFD））在航空设计中的作用。

一般来讲，形式化技术就是基于严密的、数学上的形式机制的开发方法。一方面，有些开发方法内在地与某种形式机制结合在一起，比如证明数学正确性的证明方法不能独立于代数和逻辑而单独存在。另一方面，有些形式机制和多种方法相关联。如有限状态机是一种形式化机制，但它能够支持多种形式化规格和验证方法。再者，方法（但不一定是形式化方法）和现有形式化机制的结合能够产生新的形式化技术，比如，面向对象方法可以与逻辑语言相结合产生新的形式化规格技术。

近些年来，形式化技术的学术研究及其工业应用得到了长足的发展。研究人员建立了系统设计人员易于理解的规格概念和术语，以及有效应用这些术语和概念的形式化规格技术及语言，建立了功能更加强大和完善的模型验证和定理证明技术，并开发出了与之相应的从研究原型到商品化产品的支撑工具和环境。

尽管当前对大型复杂的系统进行完整的形式化验证还不现实，但把形式化技术应用于大型复杂系统的关键部分确实是一个有效的实用策略。目前，有效的模型检验和定理证明技术已成为硬件验证中传统仿真技术的补充，而软件开发工程师们开始使用更为严格的形式化规格及验证技术，成功地完成了过程控制领域工业规模系统的设计，通信系统中大量的安全可靠通信协议得到了测试和检验。

形式化技术成功的工业应用引起了人们的普遍关注，可以预计，在未来的工业应用系统开发中，形式化技术将会发挥越来越重要的作用。

2. 形式化技术的作用

形式化技术在系统的开发过程中具有十分重要的作用。首先，形式化技术具有改善系统开发质量和提高工程效率的潜能。从表面上看，采用形式化技术会给系统开发设计人员带来不便，增加设计的难度，因为开发人员需要花一定的时间和精力去学习和掌握一些抽象的术语、概念以及相关的支持工具。但事实上，形式化技术的采用的确会降低系统的开发成本、提高开发效率。原因在于，规格的成本仅是系统整体成本开销的一小

部分，这些开销会在后期系统的测试和维护中得到高额的回报。同时，采用形式化技术，可以极大地减少系统设计开发早期阶段的错误，进而提高系统的设计开发效率。

其次，对于具有实时、安全特征的系统，形式化技术是确保系统质量的唯一实际可行的途径。具有实时、安全特征的系统是一类重要的计算机应用系统，涉及许多工业应用领域，如：过程控制系统、空中交通控制系统、病人监护系统、军事防御系统等。此类系统与其他计算机应用系统的明显区别在于：正确的时间有关行为，即系统的正确性不仅取决于逻辑结果，而且取决于产生结果的时间。这些系统的安全、可靠、正确运行显得极其重要。形式化技术克服了非形式化技术的模糊、不一致、不完备等局限，可以对系统进行严格、精确的规格，并对系统性能进行各个不同视角的验证。同时，可以最大限度地保证系统开发过程中的不同阶段、不同层次的一致性和完备性，最大限度地保证所开发系统性能的安全、可靠和正确。

形式化技术在工业系统应用中要较好地发挥作用，还有赖于如下几个方面问题的解决：

（1）易理解性。在实际应用中，形式化的文档常常因其难以理解而受到指责，从而在一定程度上限制了形式化技术的应用。为了使形式化技术易于为一般技术人员甚至没有多少技术背景的人员所理解，形式化描述应该辅以直观的符号，如采用图形符号或增加文字说明。

（2）裁剪性。形式化技术不仅可以应用于较小的说明性实例，还要能够应用于大型复杂系统的研制和开发。为此，形式化技术应具有重用性、复合性、维护性等特点。

（3）工具支持。在工程应用中，形式化技术需要有良好的工具支持。这是因为在形式化技术用于实际系统的开发时，涉及大量的技术细节问题，这些细节如果离开了辅助的自动工具将是难以实现的。原则上，系统开发人员只应负责具有创新性的任务，而一些常规技术细节问题都应该留给专用工具来自动完成。

7.2 形式化规格技术

7.2.1 形式化规格的定义及其分类

规格就是对系统或者对象及其期望的特性或者行为进行的描述。规格所要描述的内容包括：功能特性、行为特性、结构特性、时间特性。功能特性侧重于系统的功能方面，即做什么；行为特性侧重于系统的具体行为演化，即如何做；结构特性侧重于系统的组成，各个组成部分或者子系统间的联系和复合；时间特性则是时间相关的系统特性。

形式化规格通过具有明确数学定义的文法和语义的语言实现以上描述。形式化规格技术可分为：操作类、描述类和双重类。

操作类技术基于状态和迁移，因其本质上可执行，故具有直观和可视的特点；描述类技术基于数学公理和概念，通过逻辑和代数给出系统的状态空间，具有高度抽象、便于通过自动工具进行验证的特点；双重类技术则兼有二者的特点，既能够通过数学公理和概念高度抽象描述系统，又具有状态和迁移的可执行特征。

7.2.2 操作类规格技术

操作类技术通过可执行模型描述系统，即模型本身能够采用静态分析和执行模型得

到验证。尽管操作类技术于 20 世纪 70 年代末就已提出，但是还是在 Zave 将操作类方法引入编程语言 PAISley 之后才引起研究人员的重视。操作类技术侧重于系统行为的特性描述，主要包括：有限状态机及其扩展技术和 Petri 网技术等。

1. 有限状态机

（1）产生背景

有限状态机或者自动机的概念于 20 世纪 50 年代提出，包括 Moore 机和 Mealy 机。由于状态机本质上的可操作性，因而它成为多种操作模型的基础。经典的有限状态机（Moore 机和 Mealy 机），可用来规格系统的行为特性，并具有状态迁移图和状态迁移矩阵两种表述方式。经典状态机在系统描述方面主要存在以下几方面的问题：

① 在多自动机通信系统中，系统的状态为所有状态机的状态的笛卡儿积，因此系统的状态数具有状态组合复杂性问题，且会出现死锁；

② 不适合描述功能和结构特性；

③ 缺乏描述时间特性的机制；

④ 不能描述无限状态系统；

⑤ 仅能够描述顺序系统，无并发描述的能力。

为了克服多自动机通信系统描述中的通信死锁和组合状态复杂性问题，霍尔提出了通信顺序过程 CSP 模型，Milner 建立了通信系统演算 CCS 模型。CSP 的描述范围目前已得到多种扩展，如：计数器模型、迹模型、发散模型、易读模型和失效模型。失效模型可用来描述所规格系统的安全性和活性条件，包括发散和非确定性过程。迹模型可用来分析系统中历史事件行为。因此，CSP 已成为操作类和描述类技术的基础。同时，CSP 模型也在时间特性描述方面进行了扩展，如：CSP-R、赋时 CSP、通信共享资源 CSR 和通信实时状态机 CRSM。

（2）形式化定义

有限状态机 FSM（Finite State Machine）是一种基本的、简单的、重要的形式化技术，它具有广泛的应用，可用于系统生命期中从系统规格到系统设计的所有阶段。直观地理解，有限状态机就是一个具有有限状态的机器。

有限状态机是一个 5 元组 $M=(Q, \Sigma, \delta, q_0, F)$，其中：

① $Q=\{q_0, q_1, \cdots, q_n\}$是有限状态集合。在任一确定的时刻，有限状态机只能处于一个确定的状态 qi；

② $\Sigma=\{\sigma_1, \sigma_2, \cdots, \sigma_m\}$是有限输入字符集合。在任一确定的时刻，有限状态机只能接收一个确定的输入σ_j；

③ δ: $Q\times\Sigma\rightarrow Q$ 是状态转移函数。在某一状态下，给定输入后有限状态机将转入由状态迁移函数决定的一个新的状态；

④ $q0\in Q$ 是初始状态，有限状态机由此状态开始接收输入；

⑤ $F\subseteq Q$ 是终结状态集合，有限状态机在达到终态后不再接收输入。

有限状态机通常用图来表示，图中节点代表状态，有向弧表示迁移关系，输入标注在相应的关系弧旁边。图 7.1 显示了一个简单的有限状态机，该状态机有 4 个状态 q_0、q_1、q_2 和 q_3，输入集合有 3 个元素 a、b 和 c。各个状态之间的转移关系可从图中清楚地看出。

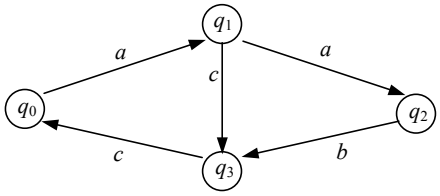


图 7.1　有限状态机

（3）“生产者与消费者”实例

“生产者—消费者”系统中，包含一个生产者和一个消费者。生产者进程产生消息，并把产生的消息写入一个能容纳两个消息的缓存区中。生产者在进行“生产”动作后，状态由 P_1 转变为 P_2；而在“写”动作后，状态由 P_2 恢复为 P_1。消费者进程能读取消息，并把消息从缓存器中取走。消费者在进行“消费”动作后，状态由 C_1 转变为 C_2；而在“读”动作后，状态由 C_2 恢复为 C_1。如果缓存器是满的，那么生产者进程必须等待，直到消费者进程从缓存器中取出一个消息，使缓存器产生一个消息空位。同样，如果缓存器是空的，那么消费者进程就必须等待，直到生产者进程产生一个消息并把所产生的消息写入缓存器中。缓存器在进行“读”动作后，缓存器大小减“1”，而在“写”动作后，缓存器大小加“1”。

利用有限状态机，分别对生产者、消费者和缓存器进行规格，如图 7.2（a）、图 7.2（b）和图 7.2（c）所示。图中清楚地给出了两个进程和一个缓存器的描述，但是作为一个整体系统，我们还需要对系统的整体行为进行规格。在这里可以利用有限状态机的复合运算或者笛卡儿积运算，得到整个系统完整行为的有限状态机规格，如图 7.2（d）所示。

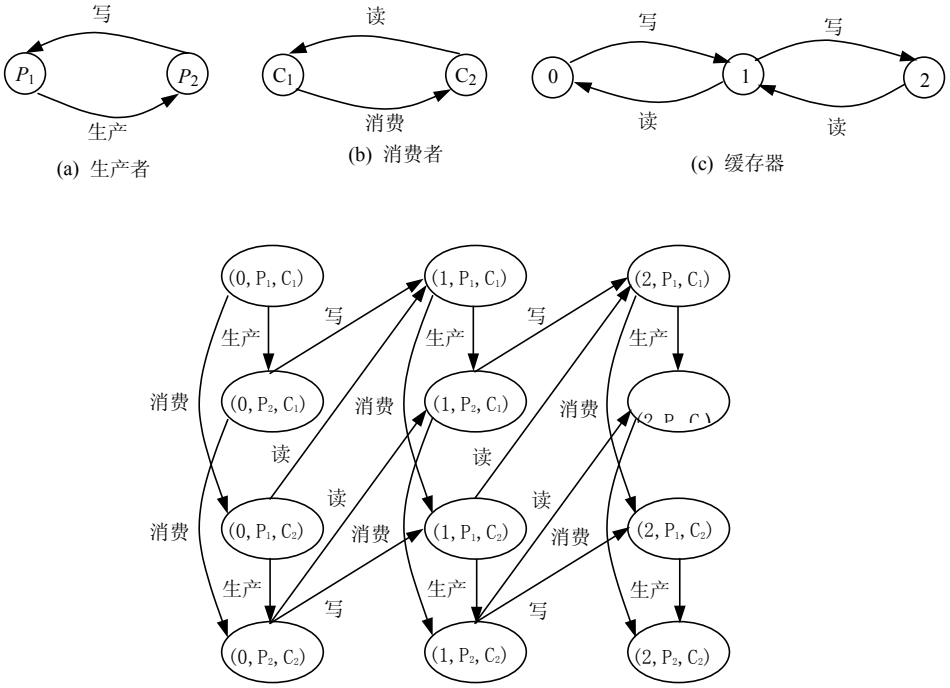


图 7.2　生产者—消费者系统规格

（4）模型工具

SREM 是第一个集成软件规格的操作类工具，支持需求描述语言 RSL 基础上的面向模型和面向特性编程。SREM 中系统分解为子函数，每一基本函数模拟为一个扩展有限状态机。状态图由 R-网给出，R-网表征了系统的演化：起始于初始状态、读取输入、产生输出。同时，R-网具有时间描述机制。

PAISley 是一种用于嵌入式实时系统的操作规格模型，所规格系统被分解为异步通信的过程。尽管通信是异步的，但模型提供了过程同步的几种方法。每一过程等价于一个扩展有限状态机。在系统的状态域中，可以实施并、笛卡儿积和其他操作。

SDL 是一种满足 ISO 和 CCITT 下 FDT 标准的操作语言。在 SDL 中，系统视为可分解成子块或者通信进程的块，因此，可描述系统的结构行为。块之间可异步通信，进程则为扩展有限状态机。在 SDL 中，时间约束通过计时器描述。

Esterel 是一种操作编程语言，支持一些基本指令集，如：loop、if-then-else 和定义时间约束表达式的指令。Esterel 允许定义在时间约束不能满足下执行过程的异常处理。Esterel 的执行是在将规格转换为通信有限状态机下实施的。

Statecharts 是一种描述复杂状态机的可视规格技术和语言。Statecharts 中复杂状态机描述为基于 AND 和 XOR 机制复合的简单状态机，从而有效改善了状态组合复杂性。同时，各个简单自动机间是并发的，并可通过广播方式进行通信。时间特性描述是通过一个特殊的时间函数来定义的。Statecharts 主要用来描述系统的行为特性。在 STATEMATE 环境下，Statecharts 规格语言和其他支撑语言一起实现了系统结构和功能特性的描述。为了改善其时间特性和功能特性描述能力，目前已进行了面向对象、时态约束、辅助变量、继承和封装等多方面的扩展，如：Modecharts、Objectcharts 和 ROOMcharts 等。

2. Petri 网

1962 年，德国学者 Petri（C.A.Petri）在他的博士论文《用自动机通信》（*Communication with Automata*）中首次使用了网状结构模拟通信系统。这种模型后来就被称为 Petri 网。后来研究人员不断拓展和丰富了 Petri 网理论。

（1）形式化定义

Petri 网是一个适合于研究具有并发、竞争、同步等特征行为的形式化技术。类似于有限状态机，Petri 网也是一种使用图形方式对系统进行规格的技术。Petri 网的描述能力强于有限状态机。

Petri 网可以定义为一个 6 元组 $N=(P，T；F，K，W，M_0)$，其中：

① $P=\{P_1，P_2，\cdots，P_m\}$是一个有限库所集，用于表示系统中资源或者条件的状态；

② $T=\{t_1，t_2，\cdots，t_n\}$是一个有限变迁集，用于表示系统中的事件；

③ $F\subseteq(P\times T)\cup(T\times P)$是一个有限的连接库所到变迁或者变迁到库所的有向弧或者关系的集合，表示事件发生的前提或者结果；

④ $K: P \rightarrow Z^+\cup\{\infty\}$ 为库所的容量函数（Z^+位正整数集）；

⑤ W：$F\rightarrow Z^+$为权函数；

⑥ M_0 为初始标识，用于表示系统的初始状态（资源的初始分布）。

上述 Petri 网的定义，在 $K\equiv 1$ 和 $W\equiv 1$ 时可简写为 4 元组 $N=(P，T；F，M_0)$。

托肯位于库所中，所有库所中托肯的分布称为标识，标识用于表示系统的状态，它

会随着变迁的发生而重新分布，从而表征系统的动态行为。

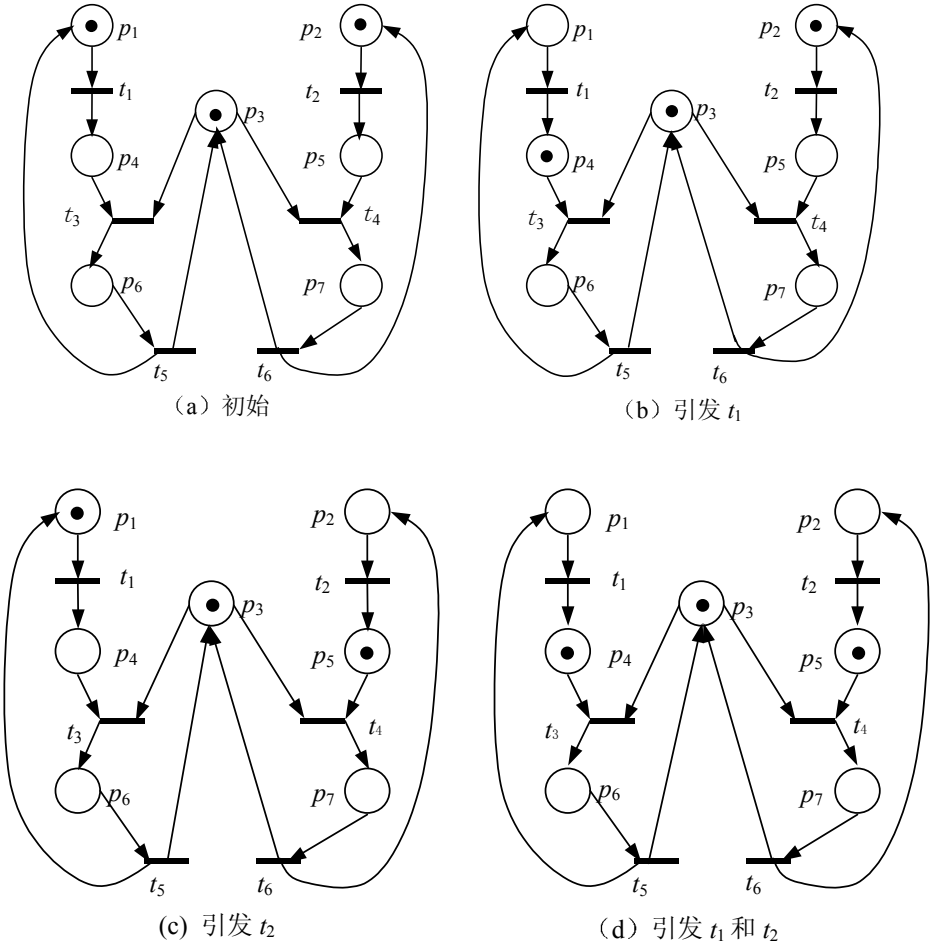
与状态机相比，Petri 网具有更强的描述能力，它克服了状态机不能描述并发的缺陷。事实上，状态机可看作 Petri 网的一类特殊的有向图，其中含有两种节点：库所和变迁。库所节点和变迁节点间用有向弧联系，这种联系可以用变迁矩阵表示。库所节点可以包含有托肯，变迁节点遵循使能规则可以引发，而引发的变迁将按照引发规则改变库所节点中托肯的分布，由此描述系统的动态行为演化。Petri 网可以描述系统中的并发、同步、冲突等特性。普通 Petri 网不能够描述系统的功能和结构特性，也缺乏时间描述机制。为了描述时间特性，通过对 Petri 网中的变迁或者库所附加时间常数因子，产生了赋时库所 Petri 网和赋时变迁 Petri 网。如果时间常数因子为一随机量则为随机 Petri 网。为了改善状态组合的复杂性和描述系统结构特性，建立了有色 Petri 网、高级 Petri 网和递阶 Petri 网。除此之外，Petri 网的其他扩展形式还有时间 Petri 网、通信 Petri 网和面向对象 Petri 网等。

（2）图形表示及实例

在图形表示中，用圆圈表示库所，用短线表示变迁，用有向弧表示关系，用圆点表示托肯。图 7.3（a）为一简单的 Petri 网的示意图。该 Petri 网中，包含库所集合{p_1，p_2，p_3，p_4，p_5，p_6，p_7}、变迁集合{t_1，t_2，t_3，t_4，t_5，t_6}，库所 p_1、p_2 和 p_3 中均包含有 1 个托肯，库所和变迁之间的联系可从图中清楚地看出。

Petri 网中，库所是静态元素，变迁是动态元素。如果存在一条从一个库所指向一个变迁的有向弧，那么我们就说这个库所是该变迁的一个输入库所；如果存在一条从一个变迁指向一个库所的有向弧，那么我们就说这个库所是该变迁的一个输出库所。如果一个变迁的每一个输入库所中都至少包含有一个托肯，那么我们称该变迁是使能的。使能的变迁可以引发，变迁的引发将从该变迁的每一个输入库所中移出一个托肯，并在每一个输出库所中增加一个托肯。

在图 7.3（a）中，变迁 t_1 和 t_2 是使能的。在这种情况下，该 Petri 网的演化，即托肯的变化就可能存在如下 3 种不同方式：引发 t_1，引发 t_2，或者引发 t_1 和 t_2。也就是说，在给定 Petri 网的初始托肯后，其演化过程可能是不同的，具有不确定性。在图 7.3（a）的情况下，引发 t_1 所得到的下一个状态如图 7.3（b）所示；引发 t_2 所得到的下一个状态如图 7.3（c）所示；引发 t_1 和 t_2 所得到的下一个状态如图 7.3（d）所示。在图 7.3（b）的情况下，变迁 t_2 和 t_3 使能。在引发变迁 t_2 后，将同样得到图 7.3（d）所示状态。在图 7.3（c）的情况下，变迁 t_1 和 t_4 使能。在引发变迁 t_1 后，将同样得到图 7.3（d）所示状态。



（a）初始　　（b）引发 t_1

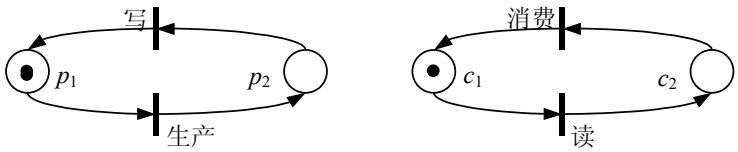
(c) 引发 t_2　　（d）引发 t_1 和 t_2

图 7.3　一个简单 Petri 网

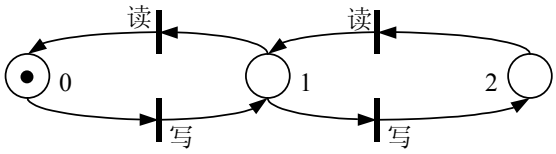
我们也可以将 Petri 网应用于生产者—消费者系统的规格。图 7.4（a）、图 7.4（b）、图 7.4（c）、图 7.4（d）依次为生产者、消费者、缓存器以及整个系统的 Petri 网规格。

7.2.3　描述类规格技术

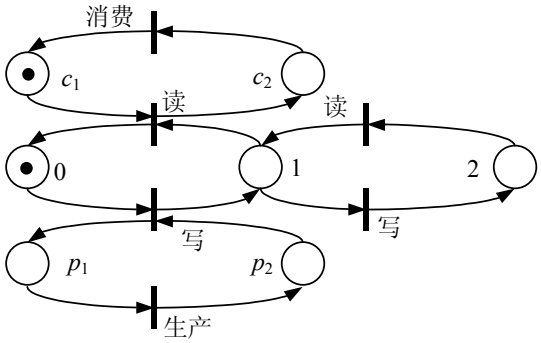
描述类规格技术的数学基础是代数或者逻辑。此类技术具有数学上的严密性和高度抽象性，并通过做什么的方式进行系统性能规格。基于不同的数学基础，描述类规格技术进一步分为：基于代数的描述技术和基于逻辑的描述技术。



（a）生产者　　（b）消费者



（c）缓存器



（d）生产者—消费者系统

图 7.4　生产者—消费者系统

1. 基于代数的描述技术

基于代数的描述技术以抽象数据类型 ADT 概念为基础，可以对系统进行由粗到细的多层次抽象。此类方法中，系统本身视为一个 ADT，规格则在于描述其文法和语义。文法定义给出 ADT 中算子域的描述。语义则通过数学表达式给出这些算子的执行，这些数学表达式的形式为用编程语言书写的一组公理。复杂 ADT 由简单 ADT 定义，重复进行则完成整个系统的规格，这样也就给后期的验证带来方便。验证可以在各个层次的规格上进行，复杂 ADT 的特性可通过简单 ADT 特性的验证而得到验证。

基于代数的描述技术主要有 Z、LOTOS、VDM 和 Larch 等。

（1）Z 语言

自 20 世纪 70 年代晚期，牛津大学的计算实验室和其他地方的程序设计研究组开发出 Z 语言，它基于集合和一阶谓词逻辑。Z 语言中 Z 是指著名的数学家 Zermolo，他的主要成就是集合论。Z 语言中有称作为框架演算的系统分解机制，系统的规格分解为许多小的框架，这些框架则描述系统的静态和动态行为。Z 规格为非形式化文本、定义、公理描述、约束、类型定义和框架的混合，其中的 ADT 操作采用谓词逻辑表述。Z 语言本身并不支持时间约束定义，Z 语言和实时区间逻辑 RTIL 结合则是在此方面的扩展。同时，Z 语言也在面向对象方面进行了扩展，如：OOZE、MooZ、Z++和 Object-Z 等，这些扩展 Z 语言都将信息封装、继承、实例等引入了 Z 框架演算，这样系统的状态空间定义为单个系统对象状态空间的复合。Object-Z 中还集成了时态逻辑，可以进行实时规格。

（2）LOTOS

从 1981~1988 年，ISO 的 FDT（Formal Description Technique）小组的专家们开发了 LOTOS（Language of Temporal Ordering Specifications），主要用于开放分布式系统形式

化规格。现在的国际标准是 IS8807。LOTOS 可通过 ADT 概念定义系统所有结构方面的特性。LOTOS 中的进程概念非常突出，结构解耦基于进程，分布式系统视为含有子进程的进程。进程之间的关系通过代数算子定义，如：顺序和并行执行。进程之间通过消息和门通信，消息可携带有数据和控制。LOTOS 下，系统规格在于定义进程行为，描述进程的通信、执行和同步，进程之间的时态顺序通过门定义。

（3）VDM

自 20 世纪 70 年代早期，IBM 实验室的研究小组提出了 VDM（The Vienna Development Method），其初始目的在于建立一种定义编程语言 PL/I 的形式化方法。目前 VDM 不仅是一种规格语言，而且成为了英国标准。VDM 的数学基础是集合理论和谓词逻辑理论。VDM 下的系统规格在于定义类型、函数和操作。数据类型由异构的 VDM 基本类型复合来定义，VDM 基本类型包括自然数、整数、布尔量等。对于新的类型，其操作可自动获得。函数定义为具有变元的过程，返回结果。函数也可由其前、后条件来规格。操作作用于状态，这些状态基于操作自身的前条件来择取。操作可以读、写外部事件。VDM 模型没有定义系统结构的机制，数据类型根据其他数据类型定义，不能够将系统分解为相互通信的子系统。目前，VDM 也在时间特性处理和面向对象方面得到了扩展。

（4）Larch

Larch 族语言是基于 ADT 的较早建立的规格语言之一，它是 LARCH 项目的主要成果之一。它支持通过代数语言—Larch 共享语言描述公共 Larch 模型，使用该语言可以定义新的 ADT。Larch 共享语言的接口支持由一谓词语言定义，该接口起着支持主语言的作用。例如：Larch/Pascal 通过通常的 Pascal 提供 Larch 型编程。目前已有多种其他 Larch 语言，如：Larch/CLU、Larch/ADA、Larch/C、Larch/Smalltalk、Larch/Modula-3 和 Larch/C++等。

2. 基于逻辑的描述技术

基于逻辑的描述技术通过逻辑规则来规格系统的演化，与操作类技术不同的是规格所描述的系统状态空间是抽象和无限的。这些规则一般是一阶 Horn 子句或者高阶逻辑公式。这类技术不能够对系统的结构特性进行描述。时态逻辑是从命题逻辑基础上演变而来的，并通过引入时态算子实现对断言随时间变化进行的描述和解释。典型的时态算子包括 always、sometimes、henceforth 和 eventually 等。

基于时态逻辑的描述规格技术主要有计算树逻辑 CTL、实时区间逻辑 RTIL、赋时计算树逻辑 TCTL 和赋时命题时态逻辑 TPTL 等。

（1）实时逻辑 RTL

实时逻辑 RTL 是一种描述事件和动作间时态关系的形式语言。RTL 中有 3 种类型的常数：动作、事件和整数。动作可以是简单的或者复合的，复合动作可以是顺序的或者并发的。事件和动作类似于激励和响应，周期事件通过递归谓词规格。整数可以是时间常数或者时间数目。系统的 RTL 规格在于从事件—动作模型推演出一组公理。RTL 中，时间是绝对的，执行语义和调度机制无关。

（2）TRIO

TRIO 是在一阶逻辑基础上引入时态函数 Dist(F,t)、Futr(F,t)和 Past(F,t)的一种语言，事件之间的时态关系则可由这些函数的一阶逻辑公式表述。时间在这里是蕴涵表示的，

所以难以规格绝对时间约束。

3. 双重类规格技术

理想的形式化规格技术，应该是既具有操作类技术的可执行性和可视性，又具有描述类技术的形式可验证性。双重类规格技术则是在此方面的努力，现有的此类规格技术包括：ESM/RTTL、TRIO+和 TROL 等。

（1）ESM/RTTL

ESM/RTTL 是一种集成扩展状态机 ESM 语言和实时时态逻辑 RTTL 的双重规格技术。ESM 是一种扩展 Mealy 机模型，其中引入了变量，以及赋值、发送和接受等操作。ESM 中状态的转移条件为状态变量的一阶表达式，输出为状态变量的赋值。时间机制通过整体时间变量描述，时间是离散的，状态迁移是即时的。RTTL 是一种在时态逻辑算子 until 和 next 基础上，增加了 Eventually 和 Henceforth 算子。通过 RTTL 的一阶逻辑公式可对系统的功能特性进行描述。

（2）TRIO+

TRIO+是 TRIO 的面向对象扩展，将可视和递阶解耦结合到描述性逻辑语言。通过定义类构件及其关系，TRIO+能够描述系统的结构特性。同时，TRIO+是一个可执行模型。

（3）TROL

TROL 是一种面向对象的双重规格语言，其中采用了修正面向对象模型，能够对系统行为、功能和结构特性进行描述。在 TROL 中，系统递阶解耦为对象和子对象，而不能进一步分解的对象定义为扩展状态机。状态机模型支持时间约束描述。TROL 语言的描述特征可用于系统的分析阶段，支持面向对象，包括继承和实例等。

7.2.4 形式化规格技术的应用

形式化规格就是精确地列出系统特性的过程。通过这一过程，可获得系统的进一步深入理解，进而发现设计中的错误、不一致性、模糊性和不完备性。同时，可以对系统进行分析、并推演出系统的一些其他性能。规格也是客户和设计者、设计者和实施者、实施者和测试人员之间的有效通信工具。某种程度上规格可理解为系统的一个原代码文件，只不过具有更高的抽象性。形式化规格技术已经在许多工业应用系统中得到成功的实施。

典型应用系统包括：IBM 商用信息控制系统、英国伦敦空中交通管理系统、空中交通防碰撞系统 TCAS 等。

牛津大学和 Hursley 实验室于 20 世纪 80 年代合作将 Z 用于 IBM 商用信息控制系统。IBM 对整个开发进行测试表明，明显地改善了产品质量、大量减少了错误和早期诊断错误。IBM 估计使总体开发成本降低 9％。这一成果获皇家技术成就奖。

Praxis 公司于 1992 年交付给英国民航局的信息显示系统是伦敦机场新空中交通管理系统的部分。在该系统的需求分析阶段，形式化描述和非形式结构化的需求概念相结合；在系统规格阶段，采用了抽象的 VDM 模型；在设计阶段，抽象 VDM 细化为更为具体的模化规格。项目开发的生产效率和采用非形式化技术相当、甚至更好。同时，软件质量得到了很大的提高，软件的故障率仅为 0.75 每千行代码，大大低于采用非形式化

技术所提供的软件的故障率（约为 2~20 每千行代码）。

加州大学的安全关键系统研究组所开发的空中交通防碰撞系统的形式化需求规格 TCASII，采用了基于 Statecharts 的需求状态机语言 RSML，解决了开发过程中遇到的许多问题。TCASII 项目表明：复杂过程控制系统列写形式化需求规格的可能性以及应用工程师们不经任何专门培训建立易读且易评判的形式化规格的可行性。

除此之外，形式化规格还在如下方面得到应用：

（1）数据库：用于存储病人监护信息的 HP 医用仪器实时数据库系统。

（2）电子仪器：Tektronix 系列谐波发生器、Schlumberger 家用电度计。

（3）硬件：INMOS 浮点处理器、INMOS 中 T9000 系列的虚拟信道处理器。

（4）医疗设备：核磁共振理疗系统。

（5）核反应堆系统：核反应器安全系统、核发电系统的切换装置。

（6）保密系统：NATO 控制指挥和控制系统中的保密策略模型、Multinet 网关系统的数据安全传输、美国国家标准和技术院的令牌访问控制系统。

（7）电信系统：AT&T 的 5ESS 电话交换系统、德国电信的电话业务系统。

（8）运输系统：巴黎地铁的自动火车保护系统、英国铁路信号控制、以色列机载航空电子软件。

（9）通信协议：协议规格、测试集的生成、协议转换。

（10）生产过程控制：安全性分析、故障诊断、控制器综合等。

7.3 形式化验证技术

形式化验证就是基于已建立的形式化规格，对所规格系统的相关特性进行分析和验证，以评判系统是否满足期望的特性。形式化验证并不能完全确保系统的性能正确无误，但是可以最大限度地理解和分析系统，并尽可能地发现其中的不一致性、模糊性、不完备性等错误。形式化验证的主要技术包括模型验证和定理证明。

7.3.1 模型检验

1. 概述

模型检验是一种基于有限模型并检验该模型的期望特性的一种技术。粗略地讲，检验就是状态空间的蛮力搜索，模型的有限性确保了搜索可以终止。

模型检验有两种主要方法。其一是时态模型检验，该方法中规格以时态逻辑形式表述，系统模拟为有限状态迁移系统。有效的搜索过程用来检验给定的有限状态迁移系统是否是规格的一个模型。另一种方法中，规格以自动机方式给出，系统也模拟为一个自动机。系统的自动机模型和规格比较，以确定其行为是否与规格的自动机模型一致。一致性概念已进行了广泛的研究，包括：语言包含、细化有序、观测等价等。

不同于定理证明，模型检验是完全自动且高效的。模型检验可用于系统部分规格，因此可用于未完全规格的系统。

模型检验的主要局限性在于状态组合爆炸问题。有序布尔决策图 OBDD 是表述状态迁移系统的高效率方法，使得较大规模系统的验证成为可能。其他可能改善状态组合

复杂性的途径有利用偏序信息、局部简化、语义最小、消除不必要状态等几种。

目前，模型检验预计可以处理 100~200 个状态变量的系统。模型检验已用来检验状态数目可达 10^{120} 的系统，并且通过采用适当的抽象技术就可以处理本质上无限状态的系统。模型检验的技术挑战在于设计可以处理大型搜索空间的算法和数据结构。

2. 应用

模型检验已在硬件电路、协议的验证、软件系统规格与分析中得到成功的应用。

（1）硬件电路

贝尔实验室对其高级数据链路控制器在 FormalCheck 下进行了功能验证，6 个性能进行了规格，其中 5 个验证无误、另外一个失败，从而进一步发现了一个影响信道流量的 Bug。LOTOS 用于 PowerPC 中的微处理器，对其中 4 个基本功能特性进行了验证。基于 CCS 语言对楼宇抗震分布式主动结构控制系统设计进行了规格，所生成的系统模型有 2.12×1019 数目的状态。经过自动分析发现了影响主动控制效果的计时器设置错误。

（2）协议的验证

基于 SMV 输入语言建立了 IEEE Futurebus+896.1－1991 标准下 cache 一致协议的精确模型，通过 SMV 验证了迁移系统模型满足 cache 一致性规格。从中发现了先前并未找到和潜在的协议设计中的错误。这项工作是第一次从 IEEE 标准中发现错误；Murphi 有限状态验证系统对 IEEE 标准 1596－1992 下 cache 一致协议进行了验证，发现了变量、逻辑等方面的错误；Philips 公司音响设备的控制协议通过 HyTech 得到了完全自动验证，这是一个具有离散和连续特征的混杂系统验证问题。

（3）软件系统规格与分析

形式化模拟和验证用于 AT&T 公司的通信软件开发，总共对 7500 条 SDL 源代码进行了验证，从中发现 112 个错误，约 55％的初始设计需求在逻辑上不一致。

3. 模型检验器

模型检验的工业应用离不开模型检验器的支持。工业应用部门可以选取现有的、也可以根据自身应用特点自行开发。目前，学术和工业界已开发出了大量的模型检验器，根据所检验规格的特点可分为时态逻辑模型检验器、行为一致检验器和复合检验器。

（1）时态逻辑模型检验器

时态逻辑模型检验器中，EMC 和 CESAR 是最早的两个；SMV 中使用了 OBDD；Spin 中采用偏序关系简化来改善状态组合复杂性；Murphi 和 UV 基于 Unity 编程语言；Concurrency Workbench 验证由 mμ演算公式表述特性的 CCS；SVE、FORMAT 和 CV 侧重于硬件验证；HyTech 用于混杂系统；Kronos 用于实时系统。

（2）行为一致检验器

行为一致检验器中，Cospan/Formal Check 基于自动机间的包含；FDR 检验 CSP 程序的细化；Concurrency Workbench 检验 CCP 程序细化。

（3）复合检验器

复合检验器中，HSIS 复合模型检验和语言包含；Step 复合模型检验和演绎方法；VIS 复合模型检验和逻辑综合；PVS 定理证明器中有用于模态 mμ演算的模型检验器；METAFrame 是支持整个软件开发过程模型检验的环境。

7.3.2 定理证明

1. 概述

定理证明是系统及其特性均以某种数学逻辑公式表示的技术。逻辑由一具有公理和推理规则的形式化系统给出。定理证明实质上是从系统公理中寻找特性证明的过程。证明采用公理或者规则，且可能推演出定义和引理。不同于模型检验，定理证明可以处理无限状态空间问题。定理证明系统可粗略地分为自动的和交互的两种类型。自动定理证明系统是通用搜索过程，在解决各种组合问题中比较成功；交互式定理证明系统则更适合于系统的形式化开发和机械形式化。同样地，定理证明的实施也是需要定理证明器的支持。

现有的定理证明器包括：用户导引自动推演工具、证明检验器和复合证明器。用户导引自动推演工具有 ACL2、Eves、LP、Nqthm、Reve 和 RRL，这些工具由引理或者定义序列导引，每一个定理采用已建立的推演、引理驱动重写和简化启发式来自动证明；证明检验器有 Coq、HOL、LEGO、LCF 和 Nuprl；复合证明器 Analytica 中将定理证明和符号代数系统 Mathematica 复合，PVS 和 Step 将决策过程、模型检验和交互式证明复合在一体。

2. 应用

定理证明在硬件和软件设计的安全特性验证中得到了应用。基于符号代数运算的自动定理证明用于证明 Pentium 中 SRT 算法的正确性，检查出了一个由故障商数字选择表引起的错误；PowerPC 和 System/390 中寄存器传输级、门级、晶体管级的硬件设计模拟为布尔状态迁移函数，基于 OBDD 的算法用来检验不同设计级上状态迁移函数的等价性；Nqthm 用于 Motorola 68020 微处理器的规格，该规格进而用来证明不同来源的二进制机器码的正确性；ACL2 用于 AMD5K86 的浮点除微代码的规格和机械证明，ACL2 还用来检验浮点方根微的正确性，发现了其中的 Bug，并对修改后的微代码进行了正确性机械证明；ACL2 用于 Motorola 复数算术处理器 CSP 的完全规格，同时对 CSP 的几个算法进行了验证；PVS 用于航空电子微处理器 AAMP5 的规格和验证，对 209 条 AAMP5 指令中的 108 条进行了规格，验证了 11 个有代表性的微代码。

思 考 题

1．什么是形式化技术？为什么要在系统开发中采用形式化技术？

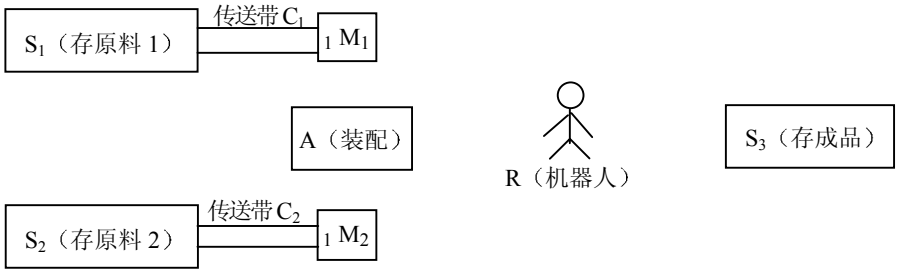
2．什么是规格，形式化规格技术是如何分类的？

3．举例说明如何使用有限自动机和 Petri 网进行系统规格？

4．简述描述类规格技术的分类及各自的特征。

5．什么是形式化验证？简述模型检验和定理证明技术及其应用。

6．如下图所示的制造系统。该系统由两个物料存储区（S1 ── 存物料 1；S2 ── 存物料 2），一个产品存储区（S3），一个装配台（A），两台机床（M1，M2），两条传送带（C1，C2）和一个机器人（R）组成。假设 S1 和 S2 中有无限的原料，成品送到 S3 后马上运走，S3 中不会出现溢出。



系统的工作过程如下：

（1）传送带 C1 和 C2 分别将原料 1 和原料 2 送到机床 M1 和 M2；

（2）机床 M1 和 M2 开始对物料进行加工，并分别得到中间产品工件 1 和工件 2；

（3）机器人把工件 1 和工件 2 送到装配站 A；

（4）装配站 A 开始工作；

（5）装配完成后，机器人将成品送到存储区 S3。

试建立该制造系统的 Petri 网模型。

第 8 章　社会和职业的问题

20 世纪 80 年代以来，随着计算机技术（特别是网络技术）的迅猛发展和广泛应用，由这一新技术带来的诸如网络空间的自由化、网络环境下的知识产权、计算机从业人员的价值观与工作观等社会和职业的问题已极大地影响着计算产业的发展，并引起业界人士的高度重视。CC1991 报告将“社会、道德和职业的问题”列入到计算学科主领域之中，并强调它对计算学科的重要作用和影响。

CC1991 报告要求计算专业的学生不但要了解专业，还要了解社会。例如要求学生要了解计算学科的基本文化、社会、法律和道德方面的固有问题；了解计算学科的历史和现状；理解它的历史意义和作用。另外，作为未来的实际工作者，他们还应当具备其他方面的一些能力，如能够回答和评价有关计算机的社会冲击这类严肃问题，并能预测将已知产品投放到给定环境中去将会造成什么样的冲击；知晓软件和硬件的卖方及用户的权益，并树立以这些权益为基础的道德观念；意识到他们各自承担的责任，以及不负这些责任可能产生的后果等。

CC2001 充分肯定了 CC1991 关于“社会、道德和职业的问题”的论述，并将它改为“社会和职业的问题”，继续强调它对计算学科的重要作用和影响。“社会和职业的问题”主要属于学科设计形态技术价值观方面的内容，广义地讲，它属于一种技术方法。

CC2001 报告将“社会和职业的问题”主领域划分为以下 10 个子领域：

（1）计算的历史；

（2）计算的社会背景；

（3）道德分析的方法和工具；

（4）职业和道德责任；

（5）基于计算机系统的风险与责任；

（6）知识产权；

（7）隐私与公民的自由；

（8）计算机犯罪；

（9）与计算有关的经济问题；

（10）哲学框架。

本章主要介绍其中 1～8 个子领域的内容，第 9 和第 10 两个子领域只列出以下研究主题，供读者参考。

与计算有关的经济问题子领域的研究主题：垄断及其经济上的含义；劳动力的供应和需求对计算产品质量的作用；计算领域的定价策略；访问计算机资源中的差异和由此产生的不同效果等。

哲学框架子领域的研究主题：相对主义、功利主义和道德理论；道德的相对论问题；从历史的角度看科学道德；哲学方法与科学方法等。

8.1 计算的历史

8.1.1 计算机史前史——1946 年以前的世界

在 1946 年美国研制成功第一台高速电子数字计算机 ENIAC 问世之前，计算机器的发展经历了一个漫长的阶段。根据计算机器的特点，可以将其划分为 3 个时代：算盘时代、机械时代和机电时代。

1. 算盘时代

这是计算机器发展史上时间最长的一个阶段。这一阶段出现了表示语言和数字的文字及其书写工具、作为知识和信息载体的纸张和书籍以及专门存储知识和信息的图书馆。这一时期最主要的计算工具是算盘，其特点是：通过手动完成从低位到高位的数字传送（十进位传送），数字由算珠的数量表示，数位则由算珠的位置来确定，执行运算就是按照一定的规则移动算珠的位置。

2. 机械时代

随着齿轮传动技术的产生和发展，计算机器进入了机械时代。这一时期计算装置的特点是：借助于各种机械装置（齿轮、杠杆等）自动传送十进位，而机械装置的动力则来自计算人员的手。如：1641 年，法国人帕斯卡利用齿轮技术制成了第一台加法机，德国人莱布尼茨在此基础上又制造出能进行加、减、乘、除的演算机；1822 年，英国人巴贝奇制成了第一台差分机（Difference Engine），这台机器可以计算平方表及函数数值表。1834 年，巴贝奇又提出了分析机（Analytical Engine）的设想，他是提出用程序控制计算思想的第一人。值得指出的是：分析机中有两个部件（用来存储输入数字和操作结果的“strore”和在其中对数字进行操作的“mill”）与现代计算机相应部件（存储器和中央处理器）的功能十分相似。遗憾的是该机器的开发因经费短缺而失败。

3. 机电时代

计算机器的发展在电动机械时代的特点是：使用电力做动力，但计算机构本身还是机械式的。1886 年，赫尔曼·霍勒瑞斯（Herman Hollerith）制成了第一台机电式穿孔卡系统——造表机，成为第一个成功地把电和机械计算结合起来制造电动计算机器的人。这台造表机最初用于人口普查卡片的自动分类和计算卡片的数目。该机器获得了极大的成功，于是，1896 年，霍勒瑞斯创立了造表公司 TMC（Tabulating Machines Company），这就是 IBM 公司的前身。电动计算机器的另一代表是由美国人霍华德·艾肯（Howard Aiken）提出、IBM 公司生产的自动序列控制演算器（ASCC），即 Mark I，它结合了霍勒瑞斯的“穿孔卡”技术和巴贝奇的通用可编程机器的思想。1944 年，Mark I 正式在哈佛大学投入运行。IBM 公司从此走向开发与生产计算机之路。

从 20 世纪 30 年代起，科学家认识到电动机械部件可以由简单的真空管来代替。在这种思想的引导下，世界上第一台电子数字计算机在爱荷华州大学（Iowa State University）产生了。1941 年，德国人朱斯（Konrad Zuse）制造了第一台使用二进制数的全自动可编程计算机。此外，朱斯还开发了世界上第一个程序设计语言——Plankalkul，该语言被当作现代算法程序设计语言和逻辑程序设计的鼻祖。此后，在包括图灵、冯·诺依曼在内的数学家研究成果的影响下，在美国军方的资助下，在包括约翰·莫奇利（John

William Mauchly）和约翰·埃克特（John Presper Echert）在内的物理学家和电气工程师的直接组织参与下，1946 年世界上第一台高速、通用计算机 ENIAC 在宾西法尼亚大学研制成功。从此，电子计算机进入了一个快速发展的新阶段。

8.1.2 计算机硬件的历史

现代计算机的历史可以追溯到 1943 年英国研制的巨人计算机和同年美国哈佛大学研制的 Mark I。今天，计算机已经历了四代，并得到了迅猛地发展。

1. 第一代计算机（1946 年~1957 年）

第一代计算机利用真空管制造电子元件，利用穿孔卡作为主要的存储介质，体积庞大，重量惊人，耗电量也很大。UNIVAC－I 是第一代计算机的代表，它是继 ENIAC 之后由莫奇利和埃克特再度合作设计的。

2. 第二代计算机（1958 年~1964 年）

在计算的历史上，1947 年晶体管的发明是一个重要的事件。使用晶体管的计算机被称作第二代计算机。和真空管计算机相比，晶体管计算机无论是耗电量还是产生的热能都大大降低，而可靠性和计算能力则大为提高。第二代计算机利用磁芯制造内存，利用磁鼓和磁盘取代穿孔卡作为主要的外部存储设备。此时，出现了高级程序设计语言，如 FORTRAN 和 COBOL。

3. 第三代计算机（1965 年~1971 年）

这一代计算机的特征是使用集成电路代替晶体管；使用硅半导体制造存储器；广泛使用微程序技术简化处理机设计；操作系统开始出现。系列化、通用化和标准化是这一时期计算机设计的基本思想。

4. 第四代计算机（1972 年至今）

主要特征是采用了大规模（LSI）和超大规模（VLSI）集成电路；使用集成度更高的半导体元件做主存储器。在此期间，微处理器产生并高速发展，个人微型计算机市场迅速扩大。第四代计算机在体系结构方面的发展引人注目，发展了并行处理机、分布式处理机和多处理机等计算机系统。同时巨型、大型、中型和小型机也取得了稳步的进展。计算机发展呈现出网络化和智能化的趋势。

随着第四代计算机向智能化方向发展，最终将导致新一代计算机的出现。新一代计算机的研制是各国计算机界研究的热点，如知识信息处理系统（KIPS）、神经网络计算机、生物计算机等。知识信息处理系统是从外部功能方面模拟人脑的思维方式，使计算机具有人的某些智能，如学习和推理的能力。神经网络计算机则从内部结构上模拟人脑神经系统，其特点是具有大规模的分布并行处理、自适应和高度容错的能力。生物计算机是使用以人工合成的蛋白质分子为主要材料制成的生物芯片的计算机。生物计算机具有生物体的某些机能，如自我调节和再生能力等。

8.1.3 计算机软件的历史

软件是由计算机程序和程序设计的概念发展演化而来的，是程序和程序设计发展到

规模化和商品化后所逐渐形成的概念。软件是程序以及程序实现和维护程序时所必须的文档的总称。

1. 第一位程序员

19 世纪初，在法国人约瑟夫·雅各（Joseph Marie Jaquard）设计的织布机里已经具有了初步的程序设计的思想。他设计的织机能够通过“读取”穿孔卡上的信息完成预先确定的任务，可以用于复杂图案的编织。早期利用计算机器解决问题的一般过程是：

（1）针对特定的问题制造解决该问题的机器；

（2）设计所需的指令并把完成该指令的代码序列传送到卡片或机械辅助部件上；

（3）使计算机器运转，执行预定的操作。

英国著名诗人拜伦（Byron）的女儿，数学家爱达·奥古斯塔·拉夫拉斯伯爵夫人（Ada Augusta Lovelace）在帮助巴贝奇研究分析机时，指出分析机可以像织布机一样进行编程，并发现进行程序设计和编程的基本要素，被认为是有史以来的第一位程序员，而著名的计算机语言 Ada 就是以她的名字命名的。

2. 布尔逻辑与程序设计

在计算机的发展史上，二值逻辑和布尔代数的使用是一个重要的突破。其理论基础是由英国数学家布尔奠定的。1847 年，布尔在《逻辑的数学分析》（*The Mathematical Analysis of Logic*）中分析了数学和逻辑之间的关系，并阐述了逻辑归于数学的思想。这在数学发展史上是一个了不起的成就，也是思维的一大进步，并为现代计算机提供了重要的理论准备。遗憾的是，布尔的理论直到 100 年之后才被用于计算。在此期间，程序设计随硬件的发展，其形式也不断发展。在基于继电器的计算机器时代，所谓“程序设计”实际上就是设置继电器开关以及根据要求使用电线把所需的逻辑单元相连，重新设计程序就意味着重新连线。所以通常的情况是：“设置程序”花了许多天时间，而计算本身则几分钟就可以完成。此后，随着真空管计算机和晶体管计算机的出现，程序设计的形式有不同程度的改变，但革命性的变革则是 1948 年香农重新发现了二值演算之后发生的。二值逻辑代数被引入程序设计过程，程序的表现形式就是存储在不同信息载体上的“0”和“1”的序列，这些载体包括纸带、穿孔卡、氢延迟线以及后来的磁鼓、磁盘和光盘。此后，计算机程序设计进入了一个崭新的发展阶段。就程序设计语言来讲，经历了机器语言、汇编语言、高级语言、非过程语言等 4 个阶段，第 5 代自然语言的研究也已经成为学术研究的热点。

3. 计算机软件产业的发展

计算机软件的发展与计算机软件产业化的进程息息相关。在电子计算机诞生之初，计算机程序是作为解决特定问题的工具和信息分析工具而存在的，并不是一个独立的产业。计算机软件产业化是在 20 世纪 50 年代，随着计算机在商业应用中的迅猛增长而发生的。这种增长直接导致了社会对程序设计人员需求的增长，于是一部分具有计算机程序设计经验的人分离出来专门从事程序设计工作，并创建了他们自己的程序设计服务公司，根据用户的订单提供相应的程序设计服务。这样就产生了第一批软件公司，如 1955 年由 Elmer Kubie 和 John W. Sheldon 创建的计算机使用公司（CUC）和 1959 年创建的应用数据研究（ADR）公司等。进入 20 世纪 60 和 70 年代，计算机的应用范围持续快速增长，使计算机软件产业无论是软件公司的数量还是产业的规模都有了更大的发展。

同时与软件业相关的各种制度也逐步建立。1968 年 Martin Goetz 获得了世界上第一个软件专利；1969 年春，ADR 公司就 IBM 垄断软件产业提出了诉讼，促使 IBM 在 1969 年 6 月 30 日宣布结束一些软件和硬件的捆绑销售，为软件产品单独定价。这一时期成立的软件公司有美国计算机公司（CCA）、Information Builder 公司和 Oracle 公司等。

8.1.4 计算机网络的历史

计算机网络是指将若干台计算机用通信线路按照一定规范连接起来，以实现资源共享和信息交换为目的的系统。

1. 计算机网络发展的 4 个阶段

（1）第一代网络：面向终端的远程联机系统。其特点是：整个系统里只有一台主机，远程终端没有独立的处理能力，它通过通信线路点到点的直接方式或通过专用通信处理机或集中器的间接方式和主机相连，从而构成网络。在前一种连接方式下主机和终端通信的任务由主机来完成；而在后一种方式下该任务则由通信处理机和集中器承担。这种网络主要用于数据处理，远程终端负责数据采集，主机则对采集到的数据进行加工处理，常用于航空自动售票系统、商场的销售管理系统等。

（2）第二代网络：以通信子网为中心的计算机通信网。其特点是：系统中有多台主机（可以带有各自的终端），这些主机之间通过通信线路相互连接。通信子网是网络中纯粹通信的部分，其功能是负责把消息从一台主机传到另一台主机，消息传递采用分组交换技术。这种网络出现在 20 世纪 60 年代后期。1969 年由美国国防部高级研究计划局建立的阿帕网（ARPANET）就是其典型代表。

（3）第三代网络：遵循国际标准化网络体系结构的计算机网络。其特点是：按照分层的方法设计计算机网络系统。1974 年美国 IBM 公司研制的系统网络体系结构（SNA）就是其早期代表。网络体系结构的出现方便了具有相同体系结构的网络用户之间的互连。但同时其局限性也是显然的。20 世纪 70 年代后期，为了解决不同网络体系结构用户之间难以相互连接的问题，国际标准化组织（ISO）提出了一个试图使各种计算机都能够互连的标准框架，即开放系统互连基本参考模型（OSI）。该模型包括 7 层：物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。模型中给出了每一层应该完成的功能。20 世纪 80 年代建立的计算机网络多属第三代计算机网络。

（4）第四代网络：宽带综合业务数字网。其特点是：传输数据的多样化和高的传输速度。宽带网络不但能够用于传统数据的传输，而且还可以胜任声音、图像、动画等多媒体数据的传输，数据传输速率可以达到几十到几百 Mbit/s，甚至达到几十 Gbit/s。第四代网络将可以提供视频点播、电视现场直播、全动画多媒体电子邮件、CD 级音乐等网上服务。作为因特网的发源地，美国在第四代计算机网络的筹划和建设上走在了世界的前列。1993 年 9 月美国提出了国家信息基础设施（NII）行动计划（NII 又被译为信息高速公路），该文件提出高速信息网是美国国家信息基础结构的 5 个部分之一，也就是这里所说的宽带综合业务数字网。现在世界各国都竞相研究和制订建设本国“信息高速公路”的计划，以适应世界经济和信息产业的飞速发展。

2. 因特网（Internet）的由来

因特网是由许多计算机网络连成的网络，也即网络的网络。它的产生主要分 3 个过程。

（1）阿帕网的诞生：1969 年，第一个计算机网络——阿帕网诞生，这种计算机网络跨越的地理范围较大，如一个省、一个国家甚至全球，被称为广域网。

（2）以太网的出现：1973 年，鲍勃·梅特卡夫（Bob Metcalfe）在施乐（Xerox）公司发明了以太网（Ethernet）。这种计算机网络所跨越的地域较小，如几个办公室、一栋大楼。今天的以太网已成为局域网的代名词。局域网的传输速率高出阿帕网几千倍，成为中小型单位网络建设较理想的选择。

（3）因特网的产生：1973 年，美国斯坦福研究院的文特·瑟夫（Virt Cerf）提出了关于计算机网络的一个重要概念——网关（Gateway），这对最终形成 TCP/IP（传输控制协议/网际协议）起了决定性的作用，因此他被人们誉为“因特网之父”。1974 年 5 月，文特·瑟夫和鲍勃·卡恩（Bob Kahn）正式发表了传输控制协议（TCP），即后来的 TCP/IP 两个协议（1978 年将 TCP 中的处理分组路由选择部分分割出来，单独形成一个 IP 协议）。1977 年，文特·瑟夫和鲍勃·卡恩成功地实现了阿帕网、无线分组交换网络和卫星分组交换网三网互连。虽说因特网源于阿帕网，但是真正促成因特网形成的则是美国国家科学基金会（NSF）。1986 年，主干网使用 TCP/IP 协议的 NSF 网络建成。1986 年~1991 年，并入 NSF 网的网络数从 100 个增到 3000 个。1989 年，NSF 网络正式改称为因特网。

8.2 计算的社会背景

8.2.1 计算的社会内涵

高科技是一柄双刃剑，计算机也不例外。计算机的广泛使用为社会带来了巨大的经济利益，同时也对人类社会生活的各个方面产生了深远的影响。不少社会学家和计算机科学家正在密切关注着计算机时代所特有的社会问题，如计算机化对人们工作和生活方式、生活质量的影响，计算机时代软件专利和版权、商业机密的保护，公民的权利和电脑空间的自由，计算的职业道德和电脑犯罪等。实际上，如何正确地看待这些影响和这些新的社会问题并制订相应的策略已经引起了越来越多计算职业人员和公众的重视。

8.2.2 网络的社会内涵

由计算机和通信线路构成的计算机网络正在使我们所在的这个世界经历一场巨大的变革，这种变革不但在人们的日常工作和生活中体现出来，而且深刻地反映在社会经济、文化等各个方面。比如：计算机网络信息的膨胀正在逐步瓦解信息集中控制的现状；与传统的通信方式相比，计算机通信更有利于不同性别、种族、文化和语言的人们之间的交流，更有助于减少交流中的偏见和误解；“网络社会”这一“虚拟的真实（Virtue Reality）” 社会有着自己独特的文化和道德，同时也存在其特有的矛盾和偏见。今天，网络技术飞速发展的事实已经使不同国籍的人们不得不对网络技术对社会政治、经济、文化、军事、国防等领域的影响及其社会意义进行认真的考虑。

网络作为资源共享的手段是史无前例的。以因特网为例，经过几十年的飞速发展，今天因特网已经成为规模空前的信息宝库。许多信息发达国家的人们已经习惯于从因特网上了解他们感兴趣的信息。如今，网络建设的发展已经成为衡量一个社会信息化程度的重要标准。

网络的迅猛发展创造了一个新的空间：电脑空间（Cyberspace）。电脑空间长期以

来处于无序状态，如因特网上至今流传着“三无”的说法（无国界、无法律、技术无法管理）。自 20 世纪 90 年代以来，随着计算机犯罪（如网上诈骗、发布恶意计算机程序等）和网络侵权事件的增多，人们逐渐认识到，为了让网络长远地造福于社会，就必须规范对网络的访问和使用。这就为各国政府、学术界和法律界提出了挑战，现在各国面临的一个难题就是如何制订和完善网络法规。具体地说，就是如何在电脑空间里保护公民的隐私，如何规范网络言论，如何保护电子知识产权，如何保障网络安全等等。

此外，网络对社会的另一个重要影响就是促使世界各国在面临网络新技术为社会带来的共同挑战时，重新认识开展国际合作的重要性。

8.2.3 因特网的增长、控制和使用

因特网的规模到底有多大，现在已经没有人能够讲得清楚了。下面给出的数据可以从一个侧面反映因特网的增长。1969 年，因特网的前身阿帕网诞生的时候，只有 4 台主机，1984 年这个数字增加到 1024，据报道，1994 年和 1995 年又分别达到 3 864 000 台和 6 642 000 台。根据最新的统计数据，2002 年前 3 个月和 2001 年同期相比，因特网主机的数量分别增长了 51.8%、56.9%和 58.0%，截止 2002 年 4 月 24 日主机数达到了 190 624 000 台。

尽管使用因特网是不受控制的，但由此造成的负面效应却不容忽视。因特网上的资料和信息并不是对所有人都适合的，这一点已经成为人们的共识。为了保证网络资源的合理使用，世界上许多国家和机构都制订了相应的政策和法规。

以美国为例，自 20 世纪 80 年代以来，美国政府相继颁布了《计算机反欺诈和滥用法》、《全球电子商务框架》和《数字千年版权法》等多项法律法规，初步建立了互联网法制的整体框架。如颁布于 1986 年的《计算机反欺诈和滥用法》，主要目的是惩处计算机欺诈和与计算机有关的犯罪行为，被视为惩治计算机黑客犯罪的里程碑；1997 年 7 月 1 日发布的《全球电子商务框架》报告，阐述了美国政府在建立全球电子商务基础结构上的原则立场，是划时代的政策性文件；1998 年 10 月 28 日由美国总统克林顿签署的《数字千年版权法》，对网络上的软件、音乐、文字作品的著作权给予了新的保护。

澳大利亚堪培拉大学在其制订的《网络使用和用户的责任与义务》中规定：该校的网络，包括因特网和 E－mail，必须并只能用于与该校有关的事务；用户必须以一种礼貌的、负责任的方式进行网上通信；用户必须遵守国家立法和学校制订与网络相关的规章、制度和政策，还规定学校有权利也有义务监督本校网络的使用与访问，以保证其与国家立法和学校的法规、制度和政策相符合。

在我国，网络立法已经受到有关方面的高度重视，近年来我国出台了多部有关网络使用规范、网络安全和网络知识产权保护的规定，如 1997 年 5 月 20 日修正的《中华人民共和国计算机信息网络国际联网管理暂行规定》，2000 年发布的《互联网信息服务管理办法》、《中文域名注册管理办法（试行）》、《教育网站和网校暂行管理办法》、《计算机病毒防治管理办法》、《关于音像制品网上经营活动有关问题的通知》、《计算机信息系统国际联网保密管理规定》和《全国人大常委会关于维护互联网安全的决定》等。这些管理规定的制订标志着我国网络法规的起步。

从技术上对用户使用因特网实施控制可以用两种方法来实现：一种是使用代理服务器的技术。代理服务器位于网络防火墙上，当代理服务器收到用户请求的时候，就检查其请求的 Web 页地址是否在受控列表中，如果不在就向因特网发送该请求，否则拒绝

请求，这是一种根据地址进行访问控制的方法，微软开发的 I－Gear 使用的就是这种方法；还有一种基于信息内容的控制技术，即从技术角度控制和过滤违法与有害信息。它主要是对每一个网页的内容进行分类，并根据内容特性加上标签，同时由计算机软件对网页的标签进行监测，以限制对特定内容网页的检索。如互联网内容选择平台（Platform for Internet Content Selection，PICS）就是这一类的技术。1996 年， Microsoft、Netscape、SurfWatch、CyberPatrol 和其他一些软件厂商宣布已经开发出了自己的 PICS 兼容产品。同年，AOL、AT&T WorldNet、CompuServe 和 Prodigy 开始提供免费的 PICS 兼容软件。

8.2.4 有关性别的问题

计算领域中的性别问题已经引起了国内外许多学者的注意。从世界范围来看，从事计算机科学的研究及从事 IT 等行业的女性所占的比例显然大大低于男性，这不仅仅是男女之间生理的差异带来的问题，更主要包括文化、经济等深层的社会环境造成的影响。在我国，虽然当前女性的就业率位居世界榜首，但与发达国家相比，我国女性从事以体力劳动为主的产业的比重较高，而从事信息和服务部门的比重甚低。当然，这种因性别问题造成的就业差别将随着数字化时代的到来而逐步淡化。20 世纪 80 年代以来，随着计算机信息与网络技术的迅速发展及广泛应用，给女性的职业选择带来了新的契机，同时也为女性平等、独立地步入国际社会创造了良好的条件。21 世纪，女性可以通过计算机网络从事网上编辑、美术设计、广告设计、会计、教师等多种职业。新的观念和新的工作模式使女性也开始加入 SHHO（Small Office and Home Office）的行列。与此同时，互联网改变了现实社会中人与人之间的关系，它突破了现实生活中地域、人的社会地位、职业以及性别等的差异，意味着个体间的真正平等。在这一变化中，女性可通过互联网以个人身份加入国际社会，扩大视野，创造更多、更自由的发展空间。

网络时代的另一性别问题是女性涉及网络的人数远远低于男性。据一些网络调查表明，在我国，女性互联网用户大大低于男性，除去女性在家庭中的地位、受教育程度以及男女之间在兴趣培养方面的差异等因素外，更主要的原因之一是网络空间的复杂性、易变性，使女性在网络中常常容易被骚扰、被欺骗，以及在网络上遇到的色情问题等等。使女性远离网络的另一原因还包括女性的网络素养问题，即对信息的判断能力及创造和传播能力。

以上有关性别的问题只是计算时代诸多性别问题中的一个侧面，如何面对和消除性别问题带来的负面影响依然是目前各国学者研究和争论的热门话题。

8.3 道德分析的方法

道德分析的主要内容包括对道德争论的分析、确定及其评价，如何进行道德选择及道德评价，如何理解软件设计的社会背景，识别假设和价值观念等等。本节着重讲述道德分析的一个重要方法——道德选择及道德评价。

8.3.1 道德选择

道德选择，就是在处理与道德相关的事务时，以道德原则（Ethical Principles）为根据，以与道德原则一致为标准对可能的道德观点进行选择的过程。进行道德选择是一件

困难而复杂的事情。这种选择往往伴随着来自经济的、职业的和社会的压力，有时这些压力会对我们所信守的道德原则或道德目标提出挑战，掩盖或混淆某些道德问题。道德选择的复杂性还在于，在许多情况下同时存在多种不同的价值观和不同的利益选择，我们必须为这些相互竞争的价值观和利益进行取舍。除此之外，有时我们赖以进行道德选择的重要事实是我们不知道、无法知道或不清楚的。既然道德选择可能会在使一些人受益的同时损害其他一些人的利益，所以我们就必须对此进行权衡，充分考虑各种道德选择可能出现的后果。

8.3.2 道德评价

道德评价是道德选择的关键。道德评价必须遵循一定的道德原则。1984 年，Kitchener 提出了下面 5 条为公众和许多社会组织接受的道德原则：

（1）自治（Autonomy）原则；

（2）公正（Justice）原则；

（3）行善（Beneficence）原则：指尽量预防和制止对他人造成的危害，并主动地做对他人有益的事；

（4）勿从恶（Nonmaleficence）原则：强调不要对他人造成伤害，并避免可能对他人造成伤害的行为；

（5）忠诚（Fidelity）原则：指诚实对人，信守诺言。

8.3.3 道德选择中其他相关因素及道德选择过程

在确定了道德评价的标准之后，为了使道德选择顺利进行，进行道德选择的人还必须具备其他一些必要的条件。比如选择者对各种选择的道德意义必须具有一定的敏感性、理解力和对道德问题的奉献精神，必须能够对复杂、不明朗或不完整的事实作出比较恰当的评价，要避免在执行道德选择时因为方式不当而对一个职业造成伤害。

道德选择一般包括以下步骤：

（1）确定所面临的问题：尽量搜集更多的信息以帮助自己对当前问题有一个清晰的认识，包括问题的性质、已有的事实、前提和假设等；

（2）利用现有的道德准则，检查该问题的适用性，如果适用则采取行动进行解决；否则如果问题比较复杂，解决方案尚不明确，则继续下面的步骤；

（3）从不同的角度认识所面临的难题的性质，包括确定特定情况下适用的道德原则，并对相互之间可能发生冲突的道德原则进行权衡；

（4）形成解决问题的候选方案；

（5）对候选方案进行评价，考虑所有候选方案的潜在道德后果，作出最为有利的选择；

（6）实施所选方案；

（7）对实施的结果进行检查和评价。

8.4 职业和道德责任

8.4.1 职业化的本质

职业化的英文单词是 Professionalism，该单词也常被译为“职业特性”、“职业作风”、“职业主义”或“专业精神”等。那么职业化的本质是什么呢？

英国 De Montfort 大学 IT 管理和研究中心的 Mohamed Sheriff 认为“职业化”应该视为从业人员、职业团体及其服务对象——公众之间的三方关系准则。该准则是从事某一职业，并得以生存和发展的必要条件。实际上，该准则隐含地为从业人员、职业团体（由雇主作为代表）和公众（或社会）拟订了一个三方协议，协议中规定的各方的需求、期望和责任就构成了职业化的基本内涵。如从业人员希望职业团体能够抵制来自社会的不合理要求，能够对职业目标、指导方针和技能要求不断进行检查、评价和更新，从而保持该职业的吸引力。反过来，职业团体也对从业人员提出了要求，要求从业人员具有与职业理想相称的价值观念，具有足够的、完成规定服务所要求的知识和技能。类似地，社会对职业团体以及职业团体对社会都具有一定的期望和需求。任何领域提供的任何一项专业服务都应该达到三方的满意，至少能够使三方彼此接受对方。

“职业化”是一个适用于所有职业的一个总的原则性协议，但具体到某一个行业时，还应考虑其自身特殊的要求，如在广播行业里，公众要求广播公司和广播人员公正地报道新闻事件，广播公司则对广播人员的语言有特别的要求。

8.4.2 软件工程师的道德准则及行为规范

任何一个职业都要求其从业人员遵守一定的职业和道德规范，同时承担起维护这些规范的责任。虽然这些职业和道德规范没有法律法规所具有的强制性，但遵守这些规范对行业的健康发展是至关重要的。计算职业也不例外。在计算机日益成为各个领域及各项社会事务中的中心角色的今天，那些直接或间接从事软件设计和软件开发的人员，有着既可从善也可从恶的极大机会，同时还可影响着周围其他从事该职业的人的行为。为能保证使其尽量发挥有益的作用，这就必须要求软件工程师致力于使软件工程成为一个有益的和受人尊敬的职业。为此，1998 年，IEEE－CS 和 ACM 联合特别工作组在对多个计算学科和工程学科规范进行广泛研究的基础上，制订了软件工程师职业化的一个关键规范：资格认证。在经过广泛的讨论和严格的审核之后，IEEE－CS 和 ACM 采纳了特别工作组提出的《软件工程资格和专业规范》。该规范不代表立法，它只是向实践者指明社会期望他们达到的标准，以及同行们的共同追求和相互的期望。该规范要求软件工程师应该坚持以下 8 项道德规范。

原则 1：公众。从职业角色来说，软件工程师应当始终关注公众的利益，按照与公众的安全、健康和幸福相一致的方式发挥作用。

原则 2：客户和雇主。软件工程师应当有一个认知，什么是其客户和雇主的最大利益。他们应该总是以职业的方式担当他们的客户或雇主的忠实代理人和委托人。

原则 3：产品。软件工程师应当尽可能地确保他们开发的软件对于公众、雇主、客户以及用户是有用的，在质量上是可接受的，在时间上要按期完成并且费用合理，同时没有错误。

原则 4：判断。软件工程师应当完全坚持自己独立自主的专业判断并维护其判断的声誉。

原则 5：管理。软件工程的管理者和领导应当通过规范的方法赞成和促进软件管理的发展与维护，并鼓励他们所领导的人员履行个人和集体的义务。

原则 6：职业。软件工程师应该提高他们职业的正直性和声誉，并与公众的兴趣保持一致。

原则 7：同事。软件工程师应该公平合理地对待他们的同事，并应该采取积极的步骤支持社团的活动。

原则 8：自身。软件工程师应当在他们的整个职业生涯中，积极参与有关职业规范的学习，努力提高从事自己的职业所应该具有的能力，以推进职业规范的发展。

软件工程资格认证将会指导从业者遵循有关协会期望他们所要符合的规范和他们所要奋斗的目标，或两者之一。更重要的是，认证将使公众意识到责任心对职业的重要性。

另外，在软件开发的过程中，软件工程师及工程管理人员不可避免地会在某些与工程相关的事务上产生冲突。为了减少和妥善地处理这些冲突，软件工程师和工程管理人员就应该以某种符合道德的方式行事。1996 年 11 月，IEEE 道德规范委员会指定并批准了《工程师基于道德基础提出异议的指导方针草案》，草案提出了 9 条指导方针：

（1）确立清晰的技术基础：尽量弄清事实，充分理解技术上的不同观点，而且一旦证实对方的观点是正确的，就要毫不犹豫地接受。

（2）使自己的观点具有较高的职业水准，尽量使其客观和不带有个人感情色彩，避免涉及无关的事务和感情冲动。

（3）及早发现问题，尽量在最低层的管理部门解决问题。

（4）在因为某事务而决定单干之前，要确保该事务足够重要，值得为此冒险。

（5）利用组织的争端裁决机制解决问题。

（6）保留记录，收集文件：当认识到自己处境严峻的时候，应着手制作日志，记录自己采取的每一项措施及其时间，并备份重要文件，防止突发事件。

（7）辞职：当在组织内无法化解冲突的时候，要考虑自己是去还是留。选择辞职既有好处也有缺点，作出决定之前要慎重考虑。

（8）匿名：工程师在认识到组织内部存在严重危害，而且公开提请组织的注意可能会招致有关人员超出其限度的强烈反应时，对该问题的反映可以考虑采用匿名报告的形式。

（9）外部介入：组织内部化解冲突的努力失败后，如果工程人员决定让外界人员或机构介入该事件，那么不管他是否决定辞职，都必须认真考虑让谁介入。可能的选择有：执法机关、政府官员、立法人员或公共利益组织等。

8.4.3 检举政策

检举（Whistle－Blowing），通常指雇员或组织成员对欺诈或辱骂虐待等不正当行为进行公开曝光的行为。世界上许多政府部门和组织都制订了与检举相关的制度和政策，以鼓励公众和雇员对部门和组织内不合理、不合法、不道德以及与公众利益相背离的行为和做法进行检举，同时保护检举人的利益不受侵害。如中国的《中华人民共和国监察法》、英国剑桥大学的《公共利益检举法案 1998》等等。在我国，根据《中华人民共和

国著作权法》、《计算机软件保护条例》以及新刑法中关于“侵犯知识产权罪”和“扰乱市场秩序罪”的有关规定，一个公司如果生产、销售或者复制使用非法软件，属于非常明确的违法行为，任何人都可以进行检举。

社会学和法学领域的研究人员一直在探索如何利用检举作为一种控制组织的不正当行为，提高管理水平、维护有效经济所必须的透明性的手段。如美国 IBM 公司在人事管理上制定了一套严格的消除不满的检举制度，这项制度不仅局限于一个国家的 IBM 公司内部，也适用于各分公司。2001 年底位列全美 500 强第 7 位的美国安然（Enron）公司破产事件震惊了全世界。安然公司高层管理人员虚报成绩，欺骗员工、股东和社会以及利用政府为自己攫取利益的行为，在使人们看到美国公司制度缺陷的同时，也突出地显示社会监督的重要性。检举作为社会监督的一种有力手段也引起了公众和相关研究人员的关注。2002 年 4 月，世界上第一次关于检举的国际大会在美国印地安娜大学召开。来自美国、欧洲、亚洲和澳洲的社会科学家和法学学者对检举行为、检举程序模型、检举的文化影响、检举与道德的关系以及其他一些与检举有关的问题进行了探讨。这表明与检举有关的问题越来越受到学术界的重视。

8.4.4 计算中的“可接受使用”政策

“可接受使用”政策通常是指计算机或网络资源提供者制订的共享资源使用规则，该规则明确资源提供者和用户各自的责任和义务，指出什么样的行为是可接受的，什么样的行为是不可接受的。接受“可接受使用”政策中规定的条款往往是用户获得共享资源使用权的前提条件。比如我们在申请免费电子信箱的时候常常看到这样的一句话：您只有无条件接受以下所有服务条款，才能继续申请……如果我们选择了“接受”，申请继续，否则申请终止。“可接受使用”政策是资源服务提供者为维持其服务、保证其服务用于所期望的目的、保护大多数用户及自身的利益不受损害而制订的。一般来讲，所制订的政策必须与有关的国家法律或组织规章制度相一致。

8.5 基于计算机系统的风险和责任

8.5.1 历史上软件风险的例子

计算机系统一般由硬件和软件两部分构成，二者的可靠性构成了整个系统的可靠性。相应地，系统的风险也就由硬件风险和软件风险构成。根据 CC2001 教学计划的要求，本书以历史上著名的“Therac-25”事件为例，着重讲述系统设计中存在的软件风险及其影响。Therac-25 事件就是历史上软件风险的著名案例。

Therac-25 是加拿大原子能公司（AECL）和一家法国公司 CGR 联合开发的一种医疗设备（医疗加速器），它产生的高能光束或电子流能够杀死人体毒瘤而不会伤害毒瘤附近健康的人体组织。该设备于 1982 年正式投入生产和使用。在 1985 年 6 月到 1987 年 1 月不到两年的时间里，因该设备引发了 6 起由于电子流或 X-光束的过量使用造成的医疗事故，造成了 4 人死亡，2 人重伤的严重后果。事故的原因要从 Therac-25 的设计说起。

Therac-25 与其前两代产品 Therac-6 和 Therac-20 相比，最大的不同之处在于：软件部分在系统中作用的差异。在前两代产品中，软件仅仅为操作硬件提供了某种方便，硬

件部分具有独立的监控电子流扫描的保护电路；而在 Therac-25 中，软件部分是系统控制机制必要的组成部分，保证系统安全运转的功能更多地依赖于软件。Therac-25 系统有 X-模式和 E-模式两种工作模式。在 X-模式（用“X”表示）下机器产生 25M 电子伏的 X-光束，在 E-模式(用“E”表示)下则产生各种能量级别的电子流；由于前者的能量非常高，所以必须经过一个厚厚的钨防护罩之后才能够与病人发生病变的人体组织相接触。模式的选择由操作员从终端上输入的数据决定。

据调查，1985 年到 1987 年间发生的 6 起事故是操作员的失误和软件缺陷共同造成的。在 1985 年第一起事故中，操作员在终端上输入错误的控制数据“X”后随即对此进行了纠正。但就在纠正输入数据的操作结束时，系统发出错误信息，操作员不得不重新启动计算机。然而就在这段时间里，躺在手术台上接受治疗的病人一直接受着过量的 X-光束的照射，结果造成其肩膀灼伤。三周之后，同样的事情又发生了。当操作员重新启动计算机的时候，支撑钨防护罩的机械手已经缩回，而 X-光束却没有被切断，结果病人被置于超出所需剂量 125 倍的强光束的辐射之下而最终致死。

事后的调查表明，Therac-25 系统中使用的软件有一部分直接来自为前两代产品开发的软件，整个软件系统并没有经过充分的测试。而 1983 年 5 月 AECL 所做的 Therac-25 安全分析报告中，有关系统安全分析只考虑了系统硬件（不包括计算机）的因素，并没有把计算机故障所造成的安全隐患考虑在内。Therac-25 作为医疗加速器设备历史上最为严重的辐射事故之一，给人们以深刻的启示：软件设计的不当很可能对系统的安全性造成巨大隐患。在开发应用系统，尤其是安全至上的应用系统时，必须充分地考虑系统设计中的软件风险。

8.5.2 软件的正确性、可靠性和安全性

软件的正确性、可靠性和安全性是影响软件质量的三个重要因素。根据 McCall 等人提出的软件质量度量模型，正确性是指程序满足其规格说明和完成用户任务目标的程度。正确性的评价准则包括：可跟踪性、完整性和一致性；可靠性是指程序在要求的精度下，能够完成其规定功能的期望程度。可靠性的评价准则包括：容错性、准确性、一致性、模块性和简洁性；安全性则是对软件的完备性进行评价的准则之一，指控制或保护程序和数据机制的有效性，比如对于合理的输入，系统会给出正确的结果，而对于不合理的输入程序则予以拒绝等。在 1985 年 ISO 建议的软件质量度量模型中，正确性和安全性是软件质量需求评价准则（SQRC）中的两条准则，前者包括软件质量设计评价准则（SQDC）中的可跟踪性、完备性和一致性，后者则包括其中的存取控制和存取审查。而 McCall 模型中的可靠性因素则体现在可容性准则和可维护性准则中。

8.5.3 软件测试

软件测试（Software Testing）是发现软件缺陷、保证软件质量的主要手段。根据 1983 年制订的美国国家标准和 IEEE 标准的定义：软件测试，就是以手工或自动方式，通过对软件是否满足特定的需求进行验证或识别软件的实际运行结果与期望值之间的不同，而对系统或系统部件进行评价的过程。

1. 软件测试的目标

（1）测试是一个程序的执行过程，其目标是发现错误；

（2）一个好的测试用例能够发现至今尚未察觉的错误；

（3）一个成功的测试则是发现至今尚未察觉的错误的测试。

但遗憾的是，现在并没有一种测试方法能够找出软件中存在的所有错误。对于大型软件系统，更加难以保证它是没有任何错误的。

对于软件测试的局限性，戴克斯特拉有句名言：测试只能够证明软件是有错的，但不能证明软件是没有错误的（Testing reveals the presence, but not the absence of bugs）。

2. 软件测试的原则

（1）程序员或程序设计机构不应测试由其自己设计的程序；

（2）测试用例设计中，不仅要有确定的输入数据，而且要有确定预期输出的详尽数据；

（3）测试用例的设计不仅要有合理的输入数据，还要有不合理的输入数据；

（4）除了检查程序是否做完了它应做的事之外，还要检查它是否做了不应做的事；

（5）保留全部测试用例，并作为软件的组成部分之一；

（6）程序中存在错误的概率与在该段程序中已发现的错误数成比例。

8.5.4 软件重用中隐藏的问题

软件重用是指在新的环境中，一些软件部件、概念和技术等被再次使用的能力，它是提高软件生产率、降低软件开发成本的有效手段。软件重用技术就是要把软件设计人员从反复设计相互雷同程序的重复劳动中解放出来。软件重用可以在不同的级别上实施，重用的单位可以是软件规格说明、软件模块和软件代码等。

与此同时，软件重用还对软件开发的各个阶段提出了新的要求和新的问题。比如在基于部件的软件开发中，为了保证软件重用部件能够成功地运用在新的应用环境中，重用部件开发者必须考虑到以下几个问题：重用部件在新的特定的环境中能否合理地发挥作用？根据是什么？对重用部件的测试是否充分考虑了可能出现的各种不同情形？设计的时候是否考虑了各种可能环境下部件的有效性、可靠性、健壮性以及可维护性？

20 世纪 90 年代后，随着面向对象软件设计方法的广泛应用，软件重用的应用前景越来越广阔，但软件重用中还存在不少理论和技术问题，尚需进一步的研究。

8.5.5 风险评定与风险管理

所谓风险就是潜在的问题，已知的确定会发生的问题不是风险。风险源于实际工作环境的不确定性。不同行业中风险的具体类别也不相同。如在软件工程中，可能的风险包括：技术风险（如目前还比较薄弱的技术领域）、来自用户的风险（如用户对项目执行情况的确认、用户新需求对原来需求的影响）、关键开发人员离职的风险、开发队伍管理不善的风险（如资源配置和协调落后）等。

风险管理（Risk Management）一词最初是由美国的肖伯纳博士于 1930 年提出的，至今还没有一个统一的概念。Karl E. Wiegers 在 *Know Your Enemy: Software Risk Management* 一文中给出的解释是：风险管理就是使用适当的工具和方法把风险限制在可以接受的限度内。台湾的袁宗慰把风险管理定义为：在对风险的不确定性及可能性等因素进行考察、预测、收集、分析的基础上，制定出包括识别风险、衡量风险、积极管理风险、有效处置风险及妥善处理风险所致损失等一整套系统而科学的管理方法。尽管

定义的细节不尽相同，但风险管理的目的却是一致的，即以一定的风险处理成本达到对风险的有效控制和处理。

一般来说，风险管理包括以下几个阶段：

（1）风险评定。风险评定是风险管理的出发点，同时又是风险管理的核心，它包括以下三方面的内容：

① 风险识别（Risk Identification），就是要确定风险的存在情况，对所面临的以及潜在的风险加以判断、归类整理并对风险的性质进行鉴定的过程。

② 风险分析（Risk Analysis），就是基于目前掌握的信息对风险发生时可能造成的损害及损害程度进行评价的过程。

③ 风险优先级评定（Risk Prioritization），其任务就是对可能存在的风险设置优先级。对发生的可能性比较大，并对组织的整体利益有较大影响的风险要设置较高的优先级。

风险评定的主要方法有：失败模型和效果分析法、危险和可操作性能（HAZOP）评定法、历史分析法、认为错误分析法、概率风险评定法和树分析法。

（2）选择处理风险的方法。即对各种处理风险的方法进行优化组合，把风险成本降到最低。

（3）风险管理效果评价。就是分析、比较已实施的风险管理技术和方法的结果与预期目标的契合程度，以此来评判管理方案的科学性、适应性和收益性。同时，当项目风险发生新变化，如目标平台改变，项目成员变动，用户需求变动的时候，风险的优先级以及风险处理的方法也要相应改变。所以风险管理效果评价的另一个内容就是对风险评定及管理方法进行定期检查、修正，以保证风险管理方法适应变化了的新情况。

需要指出的是，尽管风险管理的过程大致相同，但行业不同风险管理的原理和具体要求也不相同，如医疗、保险、审计等行业都有其各自独特的风险管理问题，需要引起职业人员的注意。

8.6 知识产权

8.6.1 什么是知识产权

目前，在世界范围内尚没有一个统一的从知识产权的内涵出发的知识产权定义。我国不少学者根据自己的理解给出了他们对知识产权的定义，如高等学校法学统编教材《知识产权法教程》所下的定义为：知识产权指的是人们可以就其智力创造的成果依法享有的专有权利。

按照 1967 年 7 月 14 日在斯德哥尔摩签订的《关于成立世界知识产权组织公约》第二条的规定，知识产权应当包括以下权利：

（1）关于文学、艺术和科学作品的权利；

（2）关于表演艺术家的演出、录音和广播的权利；

（3）关于人们努力在一切领域的发明的权利；

（4）关于科学发现的权利；

（5）关于工业品式样的权利；

（6）关于商标、服务商标、厂商名称和标记的权利；

（7）关于制止不正当竞争的权利；

（8）在工业、科学、文学或艺术领域里一切其他来自知识活动的权利。

世界各国大都有自己的知识产权保护法律体系。在美国，与出版商和多媒体开发商关系密切的法律主要有四部：《版权法》、《专利法》、《商标法》和《商业秘密法》。版权在我国称为著作权。

信息时代的知识产权问题要复杂得多，法律条文之外的讨论、争议和争论为知识产权问题增加了丰富的内容，同时这些讨论、争议和争论的存在又是完善现有知识产权保护法律体系的必要前提。Michael C. McFarland 在《知识产权、信息和公共利益》一文中列举了信息时代与电子数据相关的事务上可能存在的知识产权冲突，尤其是网络环境下的知识产权问题。美国立法中与网络知识产权保护相关的法律包括《数字千年版权法》、《反域名抢注法》等。其中，《反域名抢注法》保护公司商标不受到恶意注册域名行为的损害。

当前，有关知识产权的道义基础、因特网专利、数字时代的版权、信息的公平使用、联机社区法规、现有知识产权法律在数字化时代的使用等问题已引起人们的关注，表明了人们对信息时代知识产权问题的重视。

8.6.2 我国有关知识产权保护的现状

我国在知识产权方面的立法始于 20 世纪 70 年代末，经过 20 多年的发展现在已经形成了比较完善的知识产权保护法律体系，它主要包括《中华人民共和国著作权法》、《中华人民共和国专利法》、《中华人民共和国商标法》、《出版管理条例》、《电子出版物管理规定》和《计算机软件保护条例》等。在网络管理法规方面还制订了《中文域名注册管理办法（试行）》、《网站名称注册管理暂行办法实施细则》和《关于音像制品网上经营活动有关问题的通知》等管理规范。

另外，我国还积极参加相关国际组织的活动，非常重视加强与世界各国在知识产权领域的交往与合作。1980 年 6 月 3 日，中国正式成为世界知识产权组织成员国。从 1984 年起，中国又相继加入了《保护工业产权巴黎公约》、《关于集成电路知识产权保护条约》、《商标国际注册马德里协定》、《保护文学和艺术作品伯尔尼公约》、《世界版权公约》、《保护录音制品制作者防止未经许可复制其录音制品公约》与《专利合作条约》等诸多公约。这样，不但使中国在知识产权保护方面进一步和国际接轨，同时也提高了中国现行知识产权保护的水平。

商业秘密作为一种无形资产是知识产权的一部分，具有重大价值，需要法律的保护。美国专门制订了《商业秘密法》，加强对商业秘密的保护。我国现有的商业秘密的法律保护条款主要分散在《反不正当竞争法》、《刑法》、和《公司法》等法规中。然而，现行的商业秘密法律在与国际管理接轨中尚存在保护力度不够、法律规则不完善等问题，需要进一步加强和完善。

8.6.3 软件专利

按照知识产权保护法规，软件设计人员对其智力成果（所开发的软件）享有相应的专有权利。但是不同的国家对软件保护的形式是不一样的。软件专利是其中的一种。

美国和欧盟也有各自的软件专利保护方法。历史上，美国专利和商标事务所（P.T.O）一度不愿意给计算机软件授予专利。20 世纪 70 年代的做法是避免给包含或与计算机计

算有关的发明授予专利，理由是他们认为专利只能授予处理方法（Process）、机器（Machine）或制造厂商制造的零件（Articles of Manufacture）等，而不能授予科学事实或描述科学真理的数学表达式。P.T.O 认为计算机程序以及包含或与计算机程序相关的发明仅仅是数学算法，而不是机器或处理过程。20 世纪 80 年代，在美国最高法院的干涉下，P.T.O 被迫改变了立场，开始区别对待纯粹的数学算法和只是包含了数学算法的发明。后者可以授予专利。20 世纪 90 年代，法院又进一步认定：如果一项发明使用了计算机操纵数字，而这些数字又表示真实世界中的某个值，那么该发明就是一种处理方法，因而是可以授予专利的。法院的这些决定反映在为专利审查员编纂的指导方针里，该指导方针用于确定一项与软件相关的发明是否能够授予专利。

现在，在美国，软件不仅能被授予专利，并且相关的申请条件也较为宽松，从而使该国软件获得专利的数量大大增加，据报道 2001 年一年获得软件专利的数量就超过了 1 万项。相比之下欧洲各国对软件申请专利的要求就比较严格。

目前，在我国，软件只能申请发明专利，申请条件较严，因此，一般软件通常用著作权法来保护。软件开发者依照《中华人民共和国著作权法》和《计算机软件保护条例》对其设计的软件享有著作权。著作权包括如下人身权和财产权：

（1）发表权，即决定作品是否公之于众的权利；

（2）署名权，即表明作者身份，在作品上署名的权利；

（3）修改权，即修改或者授权他人修改作品的权利；

（4）保护作品完整权，即保护作品不受歪曲、篡改的权利；

（5）使用权和获得报酬权。

我国在加入 WTO 之后，对知识产权的保护将会越来越重视。但软件的知识产权保护问题却较为复杂，它和传统出版物的版权保护既相似又有不同，需要各个方面的专家进行深入的研究，以拿出适合我国软件发展的对策。

8.6.4 有关知识产权的国际问题

随着国际贸易和国际商业往来的日益发展，知识产权保护已经成为一个全球性的问题。各国除了制订自己国家的知识产权法律之外，还建立了世界范围内的知识产权保护组织，并逐步建立和完善了有关国际知识产权保护的公约和协议。1990 年 11 月，在关税与贸易总协定（乌拉圭回合）多边贸易谈判中，达成了《与贸易有关的知识产权协议》草案，它标志着保护知识产权的新的国际标准的形成。

8.7 隐私和公民自由

8.7.1 隐私保护的道德和法律基础

隐私，又称私人生活秘密或私生活秘密。隐私权，即公民享有的个人生活不被干扰的权利和个人资料的支配控制权。具体到计算机网络与电子商务中的隐私权，可从权利形态来分：隐私不被窥视的权利、不被侵入的权利、不被干扰的权利、不被非法收集利用的权利；也可从权利内容上分：个人特质的隐私权（姓名、身份、肖像，声音等）、个人资料的隐私权、个人行为的隐私权、通信内容的隐私权和匿名的隐私权等。

在西方，人们对权利十分敏感，不尊重甚至侵犯他人的权利被认为是最可耻的。如

在西方国家，人们不能随便问他人的年龄、工资等这一类触及隐私权的敏感问题。随着我国改革开放和经济的飞速发展，人们也开始逐渐对个人隐私有了保护意识。人们希望属于自己生活秘密的信息由自己来控制，从而避免对自己不利或自己不愿意公布于众的信息被其他个人、组织获取、传播或利用。因此，尊重他人隐私是尊重他人的一个重要方面，隐私保护实际上体现了对个人的尊重。

在保护隐私安全方面，目前世界上可供利用和借鉴的政策法规有：《世界知识产权组织版权条约》（1996 年）、美国《知识产权与国家信息基础设施白皮书》（1995 年）、美国《个人隐私权和国家信息基础设施白皮书》（1995 年）、欧盟《欧盟隐私保护指令》（1998 年）、加拿大的《隐私权法》（1983 年）等。

从总体上说，我国目前还没有专门针对个人隐私保护的法律。在已有的法律法规中，涉及到隐私保护的有以下规定：

《中华人民共和国宪法》第 38 条、第 39 条和第 40 条分别规定：中华人民共和国公民的人格尊严不受侵犯。禁止用任何方式对公民进行非法侮辱、诽谤和诬告陷害，中华人民共和国的公民住宅不受侵犯。禁止非法搜查或者非法侵入公民的住宅，中华人民共和国的通信自由和通信秘密受法律的保护，除因国家安全或者追究刑事犯罪的需要，公安机关或者检察机关依照法律规定的程序对通信进行检查外，任何组织或者个人不得以任何理由侵犯公民的通信自由和通信秘密。

《民法通则》第 100 条和第 101 条规定：公民享有肖像权，未经本人同意，不得以获利为目的使用公民的肖像，公民、法人享有名誉权，公民的人格尊严受到法律保护，禁止用侮辱、诽谤等方式损害公民、法人的名誉。

在宪法原则的指导下，我国刑法、民事诉讼法、刑事诉讼法和其他一些行政法律法规分别对公民的隐私权保护作出了具体的规定，如刑事诉讼法第 112 条规定：人民法院审理第一审案件应当公开进行。但是有关国家秘密或者个人隐私的案件不公开审理。

目前，我国出台的有关法律法规也涉及到计算机网络和电子商务等中的隐私权保护，如《计算机信息网络国际联网安全保护管理办法》第 7 条规定：用户的通信自由和通信秘密受法律保护。任何单位和个人不得违反法律规定，利用国际联网侵犯用户的通信自由和通信秘密。《计算机信息网络国际联网管理暂行规定实施办法》第 18 条规定：用户应当服从接入单位的管理，遵守用户守则；不得擅自进入未经许可的计算机学校，篡改他人信息；不得在网络上散发恶意信息，冒用他人名义发出信息，侵犯他人隐私；不得制造传播计算机病毒及从事其他侵犯网络和他人合法权益的活动。

8.7.2 隐私保护的技术

在电子信息时代，网络对个人隐私权已形成了一种威胁，电脑系统随时都可以将人们的一举一动记录、收集、整理成一个个人资料库，使我们仿佛置身于一个透明的空间，毫无隐私可言。隐私保护，已成为关系到现代社会公民在法律约束下的人身自由及人身安全的重要问题。

人们认识到，仅靠法律并不能达到对个人隐私完全有效的保护，而发展隐私保护技术就是一条颇受人们关注的隐私保护策略。发展隐私保护技术的直接目的就是为了使个人在特定环境下（如因特网和大型共享数据库系统中），从技术上对其私人信息拥有有效的控制。

现在，有许多保护隐私的技术可供因特网用户使用。这些技术大致可以分为两类：

一类是建立私人信息保护机制的技术，如 Cookies 管理、提供匿名服务、防火墙和数据加密技术等。Cookies 管理技术允许用户管理 web 站点放置在其硬盘上的 Cookies。Cookies 是由 Web 页服务器置于用户硬盘上的一个非常小的文本文件。它的用途是当用户再次访问某 Web 页的时候，通知服务器用户再次回到了该 Web 页。Cookies 中的信息包括用户登录或注册数据、“购物”选择信息、用户偏好等。Cookies 管理技术使用户可以选择禁用或有条件使用 Cookie，以避免其私人信息泄露。提供匿名服务的技术通过代理或其他方式为用户提供了匿名访问和使用因特网的能力，使用户在访问和使用因特网的时候隐藏其身份和属于个人隐私的信息。不过，不同的技术和隐私保护工具对私人信息保护的强度是不同的，而在防范别有用心的人蓄意获取他人隐私的行为上尤其不同。

另一类保护隐私的技术虽然不直接提供保护私人信息的能力，但能够增加 Web 站点隐私政策的透明性，加强用户在隐私政策上与站点的交流，从而有利于隐私保护，如 P3P 标准。2000 年 6 月 21 日， 主要的因特网公司第一次对新一代的 Web 浏览软件进行了演示。该软件是基于 P3P（Privacy Preferences Project）平台开发的，它允许用户对其私人信息拥有更强的控制能力。P3P 是万维网协会（the World Wide Web Consortium ，简称 W3C）制订的一套软件制作指导方针。在能够使用 P3P 的 Web 站点上, 用户访问该站点之前，浏览器首先把该站点的隐私政策翻译成机器可识别的形式，然后把它传递给用户，这样用户就可以基于该信息决定是否进入该网站。

8.7.3 电脑空间的言论自由

电脑空间（CyberSpace）是随着计算机信息网络的兴起而出现的一种人类交流信息、知识、情感的生存环境。它具有三个特征：

（1）信息传播方式数码化（或非物体化）；

（2）信息范围和速度的时空压缩化；

（3）获取信息全面化。

电脑空间的言论自由是一个引起全球关注的问题，同时也是一个在各国政府、产业界、学术界和法律界引起众多争议的问题。

1996 年 2 月，为了限制、阻止网上色情内容对青少年的影响和危害，美国总统克林顿签署了《通信规范法》（Communications Decency Act of 1996，CDA）。但该法生效几分钟后，美国公民自由联盟（ACLU）以该法侵害美国宪法第一修正案赋予公民的言论自由权利为由，对美国政府提出起诉。1997 年 6 月 26 日，美国最高法院终审裁定《通信规范法》违背了美国宪法，并宣布即刻废止。2000 年 6 月，一项以限制未成年人访问国际互联网上的成人内容的《儿童在线保护法》（美国国会 1998 年 1 月通过）也遭遇同样的结局。政府规范网络言论的努力又一次归于失败。

国际上，规范电脑空间言论自由的问题引起了广泛的关注。1997 年召开的经济合作和发展组织（OECD）会议对就因特网内容和在线服务达成国际协议的可能性进行了讨论。同年 9 月，在巴黎召开的第 29 届联合国教科文组织（UNESCO）代表大会上，与会的各国代表讨论了建立电脑空间法律框架的可能性。会议提交的报告把电脑空间的言论自由与信息的自由流通和信息权并列列为该组织需要考虑的重要的、复杂的、相互依赖的法律和道德问题之一。1998 年 9 月 1 日，澳大利亚广播局（Australian Broadcasting Authority）主管 Gareth Grainger 在联合国教科文组织国际代表大会上发表的演讲中说道：

很清楚，电脑空间管理的一些基本原则在国际范围内正在得到广泛的认可。这种认可在北美、欧洲和亚太地区已经得到明确的表达。接着他列举了 13 条重要的电脑空间管理原则。其中一些涉及到对网络有害内容的控制问题，如第 6 条原则认为国家权力机关有权宣布某些在线服务的内容是非法的；而第 8 条原则列举了与提供和使用在线服务相关的一些要求，比如要求在线服务提供者高度重视对未成年人和青年的保护，不使他们接触可能对其造成危害的内容。

在我国，无论是政府、立法界或学术界都同样面临着如何规范网络言论自由的问题。面对这样一个全球性难题，我国社科院新闻所研究员张西民提出三点意见：

（1）网络自律先行，法律慎行；

（2）法律的制订和调整要考虑技术的可能和可行性，以实现法律的现实性；

（3）把经济社会发展作为制订这方面法律的参照。

电脑空间言论自由的规范问题最终解决要靠全球各国的共同协作，可谓任重而道远。

8.7.4 相关的国际问题和文化之间的问题

与隐私和公民自由相关的国际问题可以从两方面来理解。一方面，世界各国在政治、经济、历史和文化等方面的差异往往造成了隐私和公民自由在不同的国家有不同的理解。比如在美国，个人收入被视为当然隐私，而在其他国家则未必如此；初到一个文化差异比较大的国家时人们常常会遭遇所谓的“文化冲击”（Cultural Shock）；同样的个人行为在一个国家被视为个人自由，而在另一个国家则可能被明令禁止等等。在进行国际交往的时候，这些差异往往是潜在冲突的根源。另一方面，当这个世界跨入信息时代的时候，各国在许多领域都同时面临着新技术带来的挑战，其中一些只有通过国际协作才能够解决，比如因特网上的隐私保护和言论自由问题。

第 29 届联合国教科文组织（UNESCO）代表大会提交的报告非常重视电脑空间的道德和法律问题，并非常希望通过国际合作协调各国和各国际组织为解决该问题所做的努力。该组织希望在下面几个主要领域建立解决其相关问题的国际框架：主权、裁判权和国际合作，民主和公平，言论自由和正直，隐私和加密，知识产权保护、数据库保护、信息访问和公平使用，安全、道德和暴力——尤其是对未成年人的保护，电子商务和跨国界的数据流交换，劳工法律事务。

8.8 计算机犯罪

8.8.1 计算机犯罪及相关立法

计算机犯罪的概念是 20 世纪五六十年代在美国等信息科学技术比较发达的国家首先提出的。国内外对计算机犯罪的定义都不尽相同。美国司法部从法律和计算机技术的角度将计算机犯罪定义为：因计算机技术和知识起了基本作用而产生的非法行为。欧洲经济合作与发展组织的定义是：在自动数据处理过程中，任何非法的、违反职业道德的、未经批准的行为都是计算机犯罪行为。

一般来说，计算机犯罪可以分为两大类：使用了计算机和网络新技术的传统犯罪和计算机与网络环境下的新型犯罪。前者例如网络诈骗和勒索、侵犯知识产权、网络间谍、

泄露国家秘密以及从事反动或色情等非法活动等，后者比如未经授权非法使用计算机、破坏计算机信息系统、发布恶意计算机程序等。

和传统的犯罪相比，计算机犯罪更加容易，往往只需要一台连到网络上的计算机就可以实施。计算机犯罪在信息技术发达的国家里发案率非常高，造成的损失也非常严重。据估计，美国每年因计算机犯罪造成的损失高达几十亿美元。

2000 年 12 月，由麦克唐纳国际咨询公司进行的一项调查结果表明：世界上大多数国家对有关计算机犯罪的立法仍然比较薄弱。大多数国家的刑法并未针对计算机犯罪制定具体的惩罚条款。麦克唐纳公司的总裁布鲁斯一麦克唐纳表示：很多国家的刑法均未涵盖与计算机相关的犯罪，因此企业和个人不得不依靠自身的防范系统与计算机黑客展开对抗。在被调查的 52 个国家中，仅有 9 个国家对其刑法进行了修改，以涵盖与计算机相关的犯罪。调查中发现，美国已制定有针对 9 种计算机犯罪的法律，惟一没有被列入严惩之列的就是网上伪造活动。而日本也制定了针对 9 种计算机犯罪的法律，惟一尚未涵盖的就是网上病毒传播。但是，调查人员指出，总体而言，很多国家即使制定有相关法律也在尺度方面非常薄弱，不足以遏制计算机犯罪。

我国刑法认定的几类计算机犯罪包括：

（1）违反国家规定，侵入国家事务、国防建设、尖端科学技术领域的计算机信息系统的行为；

（2）违反国家规定，对计算机信息系统功能进行删除、修改、增加、干扰造成计算机信息系统不能正常运行，后果严重的行为；

（3）违反国家规定，对计算机信息系统中存储、处理或者传输的数据和应用程序进行删除、修改、增加的操作，后果严重的；

（4）故意制作、传播计算机病毒等破坏性程序，影响计算机系统正常运行，后果严重的行为。这几种行为基本上包括了国内外出现的各种主要的计算机犯罪。

8.8.2 黑客（Cracking/Hacking）

虽然 Cracking 和 Hacking 都被翻译成“黑客行为”，但二者是有区别的。Cracking 是指闯入计算机系统的行为。Cracker 指在未经授权的情况下闯入计算机系统并以使用、备份、修改或破坏系统数据/信息为目的的人，媒体又称之为“坏客”或“解密高手”。Hacking 的原意是指勇于探索、勇于创新、追求精湛完美的技艺的工作作风。Hacker 最初是指在美国大学计算团体中，以创造性地克服其所感兴趣领域，即程序设计或电气工程领域的局限和不足为乐的人。根据有关资料，现在常常在 4 种意义上使用该词：

（1）指熟知一系列程序设计接口，不用花太多的精力就能够编写出新奇有用软件的人。

（2）指试图非法闯入或恶意破坏程序、系统或网络安全的人。软件开发群体中很多人希望媒体在该意义下使用 Cracker 而不是 Hacker。在这种意义下的黑客常常被称为“黑帽子黑客”。

（3）指试图闯入系统或者网络以便帮助系统所有者能够认识其系统或网络安全缺陷的人。这种人常常被称为“白帽子黑客”，他们中的许多人都受雇于计算机安全公司，他们的行为是完全合法的。其中不少人是从“黑帽子黑客”转化而来。

（4）指通过其所掌握的知识或用反复实验的方法修改软件从而改变软件功能的人，他们对软件所做的改变通常是有益的。由此可见，Cracker 意味着恶意破坏和犯罪，而

Hacker 常常意味着能力，而确切的意义则要根据具体的情况而定。

8.8.3 恶意计算机程序和拒绝服务攻击

1. 恶意计算机程序

恶意计算机程序主要有 4 种：病毒（Virus）、蠕虫（Worm）、特洛伊木马（Trojan Horse）和逻辑炸弹（Logic Bomb）。

（1）病毒：是一种感染可执行文件的程序，感染后的文件以不同于原先的方式运行，从而造成不可预料的后果，如删除硬盘文件或破坏用户数据等。计算机病毒具有“寄生”于可执行文件进行传播的能力，且只有在感染了可执行文件之后才会“发作”。世界上第一个计算机病毒是 1983 年出于学术研究的目的而编写的，而计算机第一次真正遭受病毒的恶意攻击则是在 1987 年。当时来自巴基斯坦拉合尔一台计算机上的病毒感染了美国 Delaware 大学的计算机，至少造成该校一名研究生论文被毁。从此之后，计算机病毒就以每月 100 个以上新病毒的速度不断增长。目前世界上的计算机病毒达 4 万种，并且增长速度更是达到了每月近 300 种。

（2）蠕虫：是能够自我复制的计算机程序。虽然它并不感染其他文件，但其危害是显而易见的，比如把蠕虫放在网络上会使网络流量大大增加，造成网络拥塞；而在计算机上则会占用大量存储空间。

（3）特洛伊木马：是一种能够散布病毒、蠕虫或其他恶意程序的计算机程序。公元前 1200 年，在特洛伊战争中，古希腊人利用把士兵隐藏在木马腹中的战术攻克了特洛伊城堡。这种战术用于计算机犯罪中，就是一种以软件程序为基础进行欺骗和破坏的犯罪手段。特洛伊木马程序和计算机病毒不同，它不依附于任何载体而独立存在。如 AIDS 事件就是一个典型的特洛伊木马程序，它声称是爱滋病数据库，当运行时它实际上毁坏了硬盘。

（4）逻辑炸弹：是一旦某个事件发生就会“引爆”的计算机程序，可能的后果有使“被炸”程序停止运行、使计算机崩溃、散布病毒、删除文件等。如 1996 年上海某公司寻呼台主控计算机系统被损坏一案就是一个比较典型的逻辑炸弹的案例。该公司某工程师因对单位不满，遂产生报复心理，离职时在计算机系统中设置了逻辑炸弹。这一破坏性程序在当年 6 月 29 日这一特定的时间激活，导致系统瘫痪，硬盘分区表被破坏，系统管理执行文件和用户资料数据全部丢失，使公司遭受很大的损失。

2. 拒绝服务攻击

拒绝服务（Denial-of-Service，简称 DoS）攻击是一种常见的网络攻击方式，其基本特征是：攻击者通过某种手段，如发送虚假数据或恶意程序等剥夺网络用户享有的正常服务。DoS 攻击常见的形式有：缓冲区溢出攻击，如通过发送大量的垃圾邮件耗尽邮件服务器资源，使合法的邮件用户不能得到应有的服务；扰乱正常 TCP/IP 通信的 SYN 攻击和 Teardrop 攻击；向目标主机发送哄骗 Ping 命令的 Smurf 攻击等。近年来又出现了一种攻击力更强、危害更大的“分散拒绝服务”攻击（DDoS），这种攻击采用了分布协作的方式，从而更容易在攻击者的控制下对目标进行大规模的侵犯。2000 年 2 月 7 日，Yahoo、Amazon 和 eBay 等著名站点遭受的攻击就是 DDoS 攻击，攻击期间，服务器基本停止了对合法用户的服务。现在，对 DDoS 攻击的防范已经引起计算机安全人员的高度重视。

8.8.4 防止计算机犯罪的策略

一般来说，防范计算机犯罪有以下几种策略：

（1）加强教育，提高计算机安全意识，预防计算机犯罪。一方面，社会和计算机应用部门要提高对计算机安全和计算机犯罪的认识，从而加强管理，减少犯罪分子的可乘之机；另一方面，从一些计算机犯罪的案例中看到，不少人，特别是青少年常常出于好奇和逞强而在无意中触犯了法律。对这部分人进行计算机犯罪教育，提高其对行为后果的认识，预防犯罪的发生。

（2）健全惩治计算机犯罪的法律体系。健全的法律体系一方面使处罚计算机犯罪有法可依，另一方面能够对各种计算机犯罪分子起到一定的威慑作用。

（3）发展先进的计算机安全技术，保障信息安全。比如使用防火墙、身份认证、数据加密、数字签名和安全监控技术、防范电磁辐射泄密等。

（4）实施严格的安全管理。计算机应用部门要建立适当的信息安全管理办法，确立计算机安全使用规则，明确用户和管理人员职责；加强部门内部管理，建立审计和跟踪体系。

思 考 题

1．CC1991 报告关于“社会、道德和职业的问题”的主要论述是什么？

2．从硬件来看，计算机的发展经历了哪些阶段？

3．计算机网络的发展经历了哪几个阶段？

4．因特网是怎样产生的？

5．计算机网络有何社会内涵？

6．为什么在一些国家要限制对因特网的访问？如何从技术上实现对用户使用因特网的控制？

7．结合国内外情况，分析当前计算领域中存在的性别问题。

8．什么是道德选择？它包括哪些步骤？

9．职业化的本质是什么？

10．软件工程师应具备哪些基本的道德规范？

11．如何解决软件开发过程中出现的冲突？

12．什么是计算中的“可接受使用”政策？

13．以“Therac-25 事件”为例，简述系统设计中存在的软件风险及其影响。

14．什么是软件测试？软件测试的目标是什么？软件测试的原则是什么？

15．阐述软件的正确性、可靠性和安全性之间的不同。

16．软件重用中包括哪些隐藏的问题？

17．什么是风险管理？在风险管理中如何进行风险评定？

18．什么是知识产权？它包括哪些权利？

19．简要分析隐私权在国内外的法律基础。

20．因特网上的隐私保护技术有哪几种？

21．我国刑法认定的计算机犯罪有哪几类？

22．什么是黑客行为？

23．恶意计算机程序包括哪几种？

24．什么是拒绝服务攻击？

25．如何防止计算机犯罪？

第 9 章　计算教育哲学

计算教育哲学源于教育哲学的思想，是应用哲学的基本原理探讨计算教育理论问题的一门学问，是系统化的世界观和方法论，是人们从事计算教育实践的重要思想武器。

本章主要介绍计算教育哲学的性质、研究对象及其 3 个基本任务。

9.1　概　述

1. 教育哲学思想的引入

教育哲学是运用哲学的基本原理探讨教育基本问题的一门学问，是系统化的世界观和方法论，是教育科学中的一个重要分支，是人们从事教育实践的重要思想武器。我们将它与计算学科进行有机的结合，并用它的思想来科学地阐述计算学科时，就产生了一个新的研究领域——计算教育哲学。

2. 计算教育哲学的性质和研究的对象

计算教育哲学是对计算教育科学的高度概括和总结，它给计算教育科学研究以理论上的指导，而计算教育科学的发展又为计算教育哲学提供了丰富的内容。计算教育哲学和计算教育科学的关系，既不是对立的，也不是可以互相代替的，它是计算教育科学中一门概括性、理论性更高的基础科学。

计算教育哲学研究的对象，是计算教育中的根本理论问题，而不是具体的枝节问题。它将从哲学的高度对计算教育科学进行研究和探讨，从中找出一般的规律，作为计算教育科学的理论和实践的指导。

3. 计算教育哲学的 3 个基本任务

（1）对计算教育科学中的一些根本问题，从哲学的高度，即从方法论的高度给以理论上的阐明；

（2）对计算教育史上和当前计算教育实际中有争议的问题，作出科学的分析和评论；

（3）根据计算技术发展的趋势和要求，对计算教育中提出的新课题作出回答，对未来的计算教育作出科学的预测。

9.2 计算教育哲学的第一个基本任务

计算教育哲学的第一个基本任务：从哲学的高度，即从方法论的高度对计算教育中的一些根本问题加以理论上的阐明。

计算教育哲学的第一个基本任务，实质上就是要建立起计算学科自己的方法论。从研究的内容来看，计算学科与计算教育所研究的一些根本问题在本质上是一致的。计算教育研究的根本内容无非也就是计算学科中的科学问题，抽象、理论和设计三个过程，核心概念、数学方法、技术方法、系统科学方法、形式化技术，以及社会和职业的问题等方面的内容。因此，学科方法论在对学科的一些根本问题进行阐明的同时，也对学科教育中的一些根本问题给以了理论上的阐明。在前面几章中，计算机科学与技术方法论已对计算学科中的一些根本问题进行了科学的阐明，即从方法论的角度对学科教育中的一些根本问题给以了理论上的阐明。因此，计算教育哲学的第一个基本任务已经完成。

9.3 计算教育哲学的第二个基本任务

计算教育哲学的第二个基本任务：对计算教育史上和当前计算教育实际中有争议的问题，作出科学的分析和评论。

在计算教育史上，有争议的问题主要有：计算能否作为一门学科？计算学科的本质是什么？计算学科的根本问题是什么？计算学科是否将持续兴旺下去或者是否会在我们的下一代衰落下去？计算学科是“工科”还是“理科”？程序设计在计算学科中的地位如何？计算学科目前的核心课程能准确反映这个领域吗？这些核心课程是否能培养计算方面的能力？在计算课程中如何做到理论与实践相结合等等。

9.3.1 如何定义一门学科

《计算作为一门学科》报告从定义一个学科的要求、学科的简短定义，以及支撑一个学科所需要的足够的抽象、理论和设计的内容等方面，详细地阐述了计算作为一门学科的事实。

前面的章节中，我们已介绍了计算学科的定义及其各主领域 3 个形态的核心内容。下面我们以《计算作为一门学科》报告中提出的定义一门学科的 5 项要求作为补充内容，以便使大家了解定义一门学科的方式：

（1）定义应该能为本领域以外的人所理解；

（2）定义应该以本领域以内的人为着力点；

（3）定义必须是明确的；

（4）必须阐明本学科的数学、逻辑和工程的历史渊源；

（5）必须指明本学科的根本问题和已有的重要成果；

从《计算作为一门学科》报告中来看，专家们的确就是根据这五点要求，给出了计算学科的定义。

9.3.2 计算学科的本质、根本问题以及学科的未来

1. 计算学科的本质

计算学科的本质，也就是计算的本质。图灵用形式化方法成功地表述了计算这一过程的本质。直观地说，所谓计算就是计算者（人或机器）对一条两端可无限延长的纸带上的一串 0 和 1 执行指令，一步一步地改变纸带上的 0 或 1，经过有限步骤，最后得到一个满足预先规定的符号串的变换过程。

2. 计算学科的根本问题

计算学科的根本问题，即什么能被（有效地）自动进行。

3. 计算学科的未来

计算学科能否持续兴旺下去？或者是否会在我们的下一代衰落下去？

一个学科的兴旺取决于它存不存在（或能否涌现）大量的科学问题，如果该学科的科学问题已基本解决，则这个学科就面临衰落的可能，否则，它将因为不断涌现的科学问题而继续兴旺下去。

人类的发展历史就是一个不断提出问题和解决问题的历史，本书在第 2 章中列举了计算学科各主领域需要解决的大量基本问题和一些富有挑战性的科学问题，只要这些富有挑战性的科学问题没有得到彻底解决，那么这个学科还将持续兴旺地发展下去。

9.3.3 计算学科是“工科”还是“理科”

计算学科是“工科”还是“理科”的问题是一个长期以来一直困扰计算机界的问题。这个问题在《计算作为一门学科》报告中得到阐明。

《计算作为一门学科》报告给出了一个计算学科的二维定义矩阵，使得学科各主领域中有关抽象、理论和设计 3 个形态的核心内容完整地呈现出来，该二维定义矩阵是对学科的一个高度概括和总结。3 个学科形态的内容以及学科的根本问题都清楚地表明：计算机科学和计算机工程在本质上没有区别，学科中的抽象、理论和设计要解决的都是计算中的“能行性”和“有效性”的问题。相对而言，计算机科学注重理论和抽象，计算机工程注重抽象和设计，计算机科学和工程则居中。因此，不能简单地将计算学科归属于“理科”还是“工科”，在统一认识之后，ACM 和 IEEE-CS 任务组将计算机科学、计算机工程、计算机科学和工程、计算机信息学以及其他类似名称的专业及其研究范畴统称为计算学科。

9.3.4 程序设计在计算学科中的地位

《计算作为一门学科》报告对程序设计的作用进行了以下深入的分析。

计算学科所包括的范围要远比程序设计大得多。例如：硬件设计、系统结构、操作系统结构、应用系统的数据库结构设计以及模型的验证等内容覆盖了计算学科的整个范围，但是这些内容并不是程序设计。计算机界长期以来一直认为程序设计语言是进入计算学科其他领域的优秀工具，甚至还有人认为计算科学的导论课程就是程序设计，计算科学等于程序设计等等。这些认识过分地强调了程序设计的重要性，从而阻碍了我们对计算学科的深入认识，削弱了我们宣传和展现计算学科的深度和广度的力量，并使喜欢迎接挑战的最优秀的学生离这个学科而去。这类观点还否定了计算科学是理论与实践密切

的、有机的、协调一致的产物，并将使我们误入歧途。

同时，《计算作为一门学科》报告进一步指出了程序设计在计算学科的正确地位：程序设计是计算学科课程中固定练习的一部分，是每一个计算学科专业的学生应具备的能力，是计算学科核心科目的一部分。并且，程序设计语言还是获得计算机重要特性的有力工具。

9.3.5 计算学科目前的核心课程能否培养学生计算方面的能力

就培养能力而言，目前，一些大学的核心计算课程的设置是不合适的。主要存在以下几方面的问题：

（1）面向计算学科方法论的思维能力和面向计算学科的数学思维能力是培养学生能力的重要内容，而目前多数高校计算课程中尚未将此作为其有机的组成部分。

（2）计算领域的历史内容常常不被强调，以致许多毕业生忽视计算学科的历史，重复原来的错误。

（3）许多计算专业的学生毕业后进入商业领域，而他们学习的课程并没有注重培养这方面的能力。这种能力究竟应该由计算机系来培养，还是由商业系来培养是一个长期争论的老问题。

（4）计算领域典型的实践活动包括设置和实验，为大型协作课题做贡献，以及和其他学科的交流等等，以便让他们能有效地运用计算学科的抽象和理论知识。但是，目前，大多数课程忽视了对实验室操作、集体项目和交叉学科的研究。

9.3.6 在计算课程中如何做到理论与实践相结合

这个问题实际上就是如何做到将课堂上讲授的原理与实验室培养的技能紧密结合的问题。ACM 和 IEEE-CS 的专家们非常重视这个问题，他们从实验室工作的 3 个目的出发，介绍了解决这个问题的一般方法：

（1）提供具体经验。实验室必须提供将课堂上讲授的原理运用于实际软件和硬件的设计、实现和测试的具体经验，以培养学生关于实际计算的感性认识，帮助学生理解抽象概念。

（2）强调程序设计。必须强调学生对实验室技术、硬件能力、软件工具的正确理解和运用。实验室主机上要求备有许多的软件工具以及实验和方案的适当文档，并教会学生如何正确地使用这些工具及文档。

（3）介绍试验方法。包括对试验的使用和设计、软件和硬件监控器、结果的统计分析，以及研究结果的适当陈述，使学生们懂得如何将粗心的观察和细心的试验区别开来。

实验课题应与课堂讲授的材料相协调。个人实验课题一般探讨硬件与软件的结合。根据不同的情况，实验作业可以强调简化软件开发过程的技术与工具；或强调分析和测量已有软件或比较已知的算法；还有的则可以强调基于课堂上所学原理的程序开发。

9.3.7 关于创新

创新是民族的灵魂。《中华人民共和国高等教育法》明确规定：高等教育的任务是培养具有创新精神和实践能力的高级专门人才。

创新，就是创造性地提出问题和创造性地解决问题。具体地说，它是指个体根据一定的目的和任务，利用已知的一切条件，产生出新颖、有价值的成果的认知和行为活动。

它主要包括两个重要特征：新颖性、价值性。

根据新颖性在时间和地域范围上的层次性可将创新划分为 3 个层次。

（1）低级层次：只对创造者个人来说是前所未有的。如人们在日常生活、工作中提出的一些新问题及新建议等等。

（2）中间层次：具有地区、行业的新颖性，具有一般的社会价值，能产生一定的经济效益和社会效益。如一个新的旅游项目的开发、新医疗设备的生产等等。

（3）最高层次：具有原创性，具有巨大的历史价值，甚至可以改变整个社会的理念、改变科学和技术的面貌。如创建一个科学理论体系、提出一种新的划时代的思想等。

德国国家信息技术研究中心（GMD）主席 Dennis Tsichritzis 认为，创新至少具有以下 4 种过程：

（1）提出新思想。极具生命力的新思想可以改变一个理论，然后改变实践这一新理论的活动。出版科学论文的目的就是通过同行的审查来证实其新奇性和原创性。

（2）产生新实践。

（3）产生新产品。

（4）开拓新业务。

前 3 个过程是研究中心的主旨，计算学科与大多数学科一样，将第一个过程置于最有价值的地位。

9.3.8 关于能力的培养

教育的目的就是要培养学生从事某一领域的工作能力。工作能力，也就是有效的活动能力，它是评价一个人在本领域从事独立的实践活动水平的标准，这个评价标准是基于本领域的历史的。

培养能力的教育过程有 5 个步骤：

（1）引起学习该领域的动机；

（2）充分展示该领域能做什么；

（3）揭示该领域的特色；

（4）追溯这些特色的历史根源；

（5）实践这些特色。

关于计算领域的工作者应该具备什么能力？《计算作为一门学科》报告指出有两类能力：

（1）面向计算学科的思维能力：发现本领域新的特性的能力。这些特性将导致新的活动方式和新的工具的产生。面向计算学科的思维能力应包含两层意思：其一是面向计算学科方法论的思维能力；其二是面向计算学科的数学思维能力。在面向计算学科的思维方式这类问题上，戴克斯特拉教授告诫我们，在计算机科学的教学过程中不要用拟人化的术语，而要用数学的形式化方法。

（2）使用工具的能力：使用本领域的工具有效地进行其他领域实践活动的能力。

报告建议：把面向计算学科的思维能力作为大学计算专业课程设计的主要目的。同时，计算专业工作者必须充分熟悉工具，以便与其他学科的人们有效地合作，进行那些学科的设计活动。

9.4 计算教育哲学的第三个基本任务

计算教育哲学的第三个基本任务：根据计算技术发展的趋势和要求，对计算教育中提出的新课题作出回答，对未来的计算教育作出科学的预测。

计算学科的迅猛发展，对计算教育的内容、思想方法和手段均产生了深刻的影响。下面，我们以 CC2001 的工作为基础，从计算技术的变化和文化的改变两方面，对计算教育当前面对的一些新问题作出回答，并给出制定新的教学计划的原则，同时还对未来计算教育的发展趋势作一些简要的分析和展望。

9.4.1 技术的变化

从计算学科 3 个形态的内容来看，影响计算领域变化的不是来自其抽象和理论两个形态的东西，而主要是来自其设计形态的内容，即来自于技术的进步，这就是我们只讨论计算技术的改变对计算学科教育影响的原因。

1965 年，Intel 公司的创始人摩尔（Gordon Moore） 提出了著名的摩尔定律，他预测微处理器处理速度每 18 个月要增加一倍，该定律至今仍然适用。结果，我们已经看到，可以获得的计算能力呈指数增长，这使得在短短的几年前尚不能解决的问题有可能得到解决。该学科的其他变化，比如万维网出现后网络的迅猛增长更富戏剧性，这表明该变化也是革命性的。渐进的和革命性的变化都使计算领域所要求的知识体系和教育过程受到影响。过去十年的技术进步使许多课程的主题内容变得更加重要，比如下列各项：

（1）WWW 及其应用；

（2）网络技术，尤其是基于 TCP/IP 的技术；

（3）图形学和多媒体；

（4）嵌入式系统；

（5）关系数据库；

（6）面向对象的程序设计；

（7）先进的应用程序接口（APIs）的应用；

（8）人机交互；

（9）软件安全；

（10）安全与加密。

鉴于以上主题重要性的日益突出，把它们收入大学本科的课程之中是理所应当的。但当前的现状是，大多数学校的教学计划，在不减少如汇编语言程序设计、形式语义学和数值分析等传统课程学时的情况下，很难增加新的主题。

9.4.2 文化的改变

计算教育也受到文化变更和变更赖以发生的社会背景的影响。例如，以下变化都对教育过程的性质产生了影响。

（1）新技术带来了教育方法的改变

推动计算学科最新发展的技术变革与教育文化都有直接的联系。例如，计算机网络的发展使分布在大范围内的机构和学校可以共享课程资料，从而使远程教育得以实现和发展。同时，新技术也影响了教育学的性质，计算学科的讲授方式较过去有了很大变化。

计算教程的设计必须把这些引起变化的技术考虑在内。

（2）计算的发展影响了教育的变革

在过去的十多年里，计算领域已得到了大大的拓展，例如，20 世纪 90 年代初，即使在像美国这样发达的国家，上因特网的家庭也为数不多，而如今，上网已经是一件很普通的事情了。计算领域的拓宽明显地影响着教育的变革，这其中包括计算学科的学生对计算的了解程度及其应用能力的提高，以及接触与不接触计算的人们之间技术水平的差距。

（3）经济影响

由于对高技术产业的极度狂热，从而对教育及其可用资源产生了巨大的影响。对计算专家的巨大需求和能够得到丰硕经济回报的前景吸引了许多学生涉足该领域，其中包括一些对计算专业几乎没有一点内在的兴趣的学生。而产业的大量需求，使得高等学校的吸引力大大降低，造成人才的大量流失，反过来，又极大地影响着计算专业人才的培养。

（4）计算作为一门学科已不再是问题

在计算发展的初期，许多机构还在为“计算”的地位而抗争。毕竟，那时候它还是一个新的科目，没有支持其他多数学术领域的历史基础。在某种程度上，这个问题贯穿了 CC1991 创作过程的始末。该报告与《计算作为一门学科》报告密切相关，并为“计算”的重要地位而进行的抗争获得了胜利。现在，计算学科已经成为许多大学最大最活跃的学科之一，同时再也没有必要为把计算教育是否列入学科而进行争论了。现在的问题是找到一种方法来满足这种需求。

（5）计算学科的拓展

随着计算学科的成长及其合法地位的确定，计算学科的研究范围得到了扩展。早年，计算主要聚焦在计算机科学上，其基础是数学和电气工程。而近些年来，计算学科发展成为一个更大更具包容性的领域，开始涉及到越来越多的其他领域。CC2001 任务组认为，理解计算学科的拓展对计算教育的影响，是我们工作的重要组成部分。

9.4.3 制定教学计划的原则

基于对过去课程报告的分析和前面所列学科的变化，CC2001 工作组为制定教学计划确定了以下原则：

（1）教学计划必须对新的分支学科有一定的敏感度，并且确保计算基础课程能为更广大范围的人们服务。

（2）教学计划仍要强调基础课程及实验工作的重要性，做到理论和实践有机地结合在一起。CC2001 明确赞成 CC1991 所阐述的观点：掌握这个学科不仅包括理解基本的主题，而且要理解这些概念在解决现实世界问题时的适用性。

（3）建立一种允许该教程的各组成部分循环更新的方法，以适应计算学科的快速发展。考虑到我们学科变化的速度，每隔 10 年更新教程的做法已经行不通了，应做到随时对相应的课程进行检查、更新。

（4）必须超越知识单元，并能有效地指导单个课程的设计。尽管 CC1991 使用的知识单元结构能提供一个有用的知识框架，但大多数的学校需要的是更详细的指导。明确描述一系列定义完好的模型将更便于各学校共享教育学的策略和工具，同时也为提供这些课程教材和其他资料的出版者提供了一个教学计划框架。

（5）确定一个相对较小并要求所有计算学科学生掌握的核心知识单元的集合，以更

大的弹性适应计算学科的变革。CC2001 任务组指出，核心知识单元包括计算机科学、计算机工程和其他类似命名的实质性主题的内容，这些内容将是人们公认的对本科生的基本要求。然而，核心本身并不能构成完整的大学本科教程，它必须还要由另外的课程加以补充，可以根据不同的学校、不同的研究领域或者不同的学生而设置不同的课程。

（6）必须为所要求的核心以外的课程提供指导方针。除了指定该学科的基本核心以外，教学计划还必须为作为更前沿领域技术选修课的高级课程提供指导方针。

（7）教学计划在范围上必须是国际性的。CC2001 的目标对象并不局限于美国一个国家，相反，它必须对全世界的计算专业人员都是有帮助的。

（8）教学计划的编写必须有产业方面人士的参与。多数计算学科本科毕业生将在产业界工作，为了使毕业生对他们即将面临的工作岗位有一个充分的适应和准备过程，有必要在新教程的设计、发展和执行中引进从业人员的参与。

（9）教学计划有必要在强调理论的同时，强调职业实践。为了保证毕业生能够顺利地适应新的工作环境，作为计算教育的一部分，必须使学生接受实践的锻炼。此外，这些实践应包括超出特定计算技能之外的广阔的活动范围，如管理、道德规范和价值观念，书面和口头的交流以及作为项目组成员的工作协调能力等。

（10）CC2001 教学计划必须满足的其他要求：

① 要充分概括计算学科的内容，以满足具有不同的重点和目标的计算教学计划的需求；

② 要有足够的灵活性，以便适时调节以适应计算学科未来的发展；

③ 必须得到美国计算科学鉴定委员会和美国工程和技术鉴定委员会（The Accreditation Board for Engineering and Technology）以及其他国家类似组织的认可。

9.4.4 未来计算教育的发展

本书从整个学科综述性导引课程的构建、教程的继续完善，以及计算教育理论体系等几个重要问题入手，对未来计算教育的发展作简要分析。

1. 关于整个学科综述性导引课程的构建问题

在计算教育史上，有关整个学科综述性导引课程（导论）的构建问题是一个长期以来引起激烈争论的主题。CC2001 报告对该问题非常重视，并鼓励各种组织团体（如学会、协会或研究会）以及教师个人从事这方面的研究。报告指出，整个学科综述性导引课程的构建有助于推动学科的发展。

在整个学科综述性导引课程的构建问题上，人们容易将《计算机操作初步》（也称《计算机文化基础》）与《计算机导论》（或《计算机科学与技术方法论》）混为一谈。其实，这是两门性质不同的课程。

《计算机操作初步》要解决的是人们对计算机功能的工具性认识，它的目的在于培养人们操作计算机的初步能力，而《计算机导论》要解决的是人们对计算本质的认识问题。就“计算作为一门学科”而言，我们不能局限于仅仅把“计算”看成一种工具，而更应该让学生们理解和掌握计算学科的基本原理、根本问题，以及解决问题的新的思维模式。

根据《计算作为一门学科》报告任务组的要求，整个学科综述性导引课程应采用严密的方式将学生引入计算学科各个富有挑战性的领域。根据本书的分析，这种满足严密

性和挑战性要求的整个学科的综述性导引课程的构建问题已演变为《计算机科学与技术方法论》课程的构建问题。本书基本完成了《计算机科学与技术方法论》课程的构建任务，从而为科学的认知计算学科提供了一个强有力的工具。

2. 关于教程继续完善的问题

CC2001 报告强调：计算机科学核心本身并不能构成一个完整的教程。为了使教程得到完善，计算学科教学计划还须增加计算职业所需的一定技能和背景知识，以及高级课程知识单元的内容。

CC2001 报告给出了对计算专业学生进行计算教育的一般要求，它认为一个成功的计算专业大学毕业生除了需要掌握计算机科学知识体系中包括的技术之外，还需要具备一定的数学素养、科学的研究和思维方法、有效的交流技能以及作为项目组成员富有成效地开展工作的能力。为了使教程有一定的深度，报告概述了一系列高级课程的内容，并讨论了高级课程的设置问题。最后，针对各类高校继续完善教学计划的需要，报告还给出和分析了几个教程模型。

3. 关于计算教育理论体系的构建问题

尽管计算机技术正突飞猛进地发展，但从该学科诞生至今，也不过几十年的时间，而在高等学校全面开展计算机教育的时间则更为短暂。与其他学科相比，计算教育显得有些幼稚和不成熟，还存在一些长期争论的系列问题，这些问题长期地困扰着计算教育界，并有可能使我们计算学科的教育研究陷入某种困境。

以哲学为指导思想进行计算教育的科学研究，是计算教育走向成熟的必由之路，也是解决计算教育根本性问题的必由之路。

相对于教育哲学，计算教育哲学来源于计算教育科学，是计算学科、教育学和哲学的交叉学科，它是在计算教育科学不断发展的基础上逐步分离出来的一门分支，是计算教育科学中一门概括性、理论性更高的基础科学。今天，这个领域的雏形已经建立。

综上所述，计算教育哲学是我们研究计算教育的指导思想，因此，在现有基础上深入地进行计算教育哲学的研究，不仅将使计算学科长期以来存在的争论问题从哲学的高度得到解决和阐述，使计算教育的改革取得实质性的突破，同时也将对未来计算教育（甚至整个理工科教育）的科学研究产生积极而深远的影响。

思 考 题

1．什么是计算教育哲学？
2．计算教育哲学研究的对象是什么？
3．计算教育哲学的三个基本任务是什么？
4．如何定义一门学科？
5．计算学科的本质是什么？什么是计算学科的根本问题？
6．计算学科是“理科”，还是“工科”？
7．简述程序设计在计算学科中的地位？
8．当前大学计算学科核心课程的设置存在哪些主要问题？
9．如何做到计算课程中的理论与实践相结合？

10.《中华人民共和国高等教育法》中规定的高等教育的任务是什么？

11．什么是创新？“创新”包括哪些过程？

12．根据新颖性在时间和地域范围上的层次性可将创新划分为哪三个层次？

13．出版科学论文的目的是什么？

14．培养能力的教育包括哪几个过程？

15．计算领域工作者应具备什么能力？

16．面向计算学科的思维能力包含哪两层意思？

17．制定计算教学计划应遵循哪些原则？

18．试从整个学科综述性导引课程的构建、教程的继续完善，以及计算教育理论体系等问题入手，对未来计算教育的发展作简要分析。

附录　计算机科学知识体

CC2001 报告用计算机科学知识体的概念定义了计算科学的知识领域，为计算学科教学计划核心课程的详细设计奠定了基础。

计算机科学知识体不仅对各主领域进行了概括，对没有列入核心单元的科学计算主领域进行了说明，而且还给出了每个知识单元的学习目标。计算机科学知识体的详细内容来源于众多学者的工作，是高校制定计算学科教学计划的一份必备资料，同时，对每一位从事计算教育工作的教师来说，又是一份极其重要的参考文献。下面，我们用表格的形式给出计算机科学知识体的框架，供读者参考。

计算机科学知识体划分为下面 3 个层次：

第一个层次——领域。它代表一个特定的学科主领域，每个主领域由两个字母的缩写表示，比如 OS 代表操作系统，PL 代表程序设计语言。

第二个层次——单元。它对主领域作了进一步地划分。每个单元都用一个领域名加一个数字后缀表示，如 OS3 是关于并发的单元。

第三个层次——主题。它是单元的组成部分，是知识体中最小的组成单位。

注：在附表-1 中，打波浪线的为核心单元。

附表-1　　计算机科学知识体

主领域	单元	主题
1.（DS）离散结构（共 43 个核心小时）	DS1. 函数,关系,集合（6 小时）	函数 （满射、入射、逆、复合）；关系 （自反、对称、传递、等价关系）；集合 （文氏图、补集、笛卡尔积、幂集）；鸽洞原理；基数和可数性
	DS2. 基本逻辑（10 小时）	命题逻辑；逻辑联结词；真值表；范式（合取与析取范式）；永真性；谓词逻辑；全称量词和存在量词；假言推理和否定后件推理（modus tallens）；谓词逻辑的局限性
	DS3. 证明技术（12 小时）	蕴含，逆，补，逆否, 否定，矛盾；形式证明的结构；直接证法；反例证法；通过逆否命题证明；归谬证法；数学归纳；完全归纳；递归数学定义；良序
	DS4. 计算基础（5 小时）	计算参数（和积规则、包含排斥原理、算术和几何级数、斐波那契（Fibonacci）数列）；鸽洞原理；排列和组合（基本定义、Pascal indentity、二项式定理）；求解递推关系式（常见实例、马斯特（Master）定理）
	DS5. 图和树（4 小时）	树；无向图；有向图；生成树；遍历策略
	DS6. 离散概率（6 小时）	有限概率空间、概率的度量、事件；条件概率，独立性、贝叶斯定律；整型随机变量、期望
2. （PF）程序设计基础（共 38 个核心小时）	PF1.基本程序设计结构（9 小时）	高级语言的基本语法和语义；变量、类型、表达式和赋值；简单 I/O；条件和循环控制结构；函数和参数传递；结构化分解
	PF2.算法和问题求解（6 小时）	问题求解策略；算法在问题求解过程中的作用；算法的实现策略；调试策略；算法的概念和性质

	PF3. 基本的数据结构（14 小时）	原语类型；数组；记录；字符串和字符串处理；数据在内存中的表示；静态、栈和堆分配；运行时间存储管理；指针和引用；链接结构；栈、队列、哈希表的实现策略；图和树的实现策略；选择正确数据结构的策略
	PF4. 递归（5 小时）	递归的概念；递归数学函数；简单的递归过程；分而治之策略；递归回溯；递归的实现
	PF5. 事件驱动的程序设计（4 小时）	事件处理方法；事件传播；异常处理
3. （AL） 算法和复杂性（共 31 个核心小时）	AL1. 基本算法的分析（4 小时）	复杂度的上界和平均值近似分析；区分最好，平均和最坏情况下行为的不同；大 “O”、小 “o”、ω 、和θ 表示法；标准复杂类；性能的实验测量；算法的时间和空间折衷；使用循环关系分析递归算法
	AL2.算法策略（6 小时）	蛮力算法；贪婪算法；分而治之；回溯；分支界限；启发式；模式匹配和字符串/文本算法；数值逼近算法
	AL3. 基本的计算算法（12 小时）	简单的数值处理算法；顺序和二分查找算法；二次排序算法（选择、插入）；O（N log N）排序算法（快速排序、堆分类排序、归并排序）；哈希表（包括避免冲突的策略）；二分查找树；图的表示（邻接表、邻接矩阵）；深度和广度优先遍历；最短路径算法（Dijkstra 和 Floyd 算法）；传递闭包（Floyd 算法）；最小生成树（Prim 算法和 Kruskal 算法）；拓扑排序
	AL4. 分布式算法（3 小时）	一致与选举；终结检测；容错；稳定性
	AL5. 基本可计算性理论（6 小时）	有限状态机；上下文无关语法；易处理和不易处理的问题；不可计算函数；停机问题；不可计算性的意义
	AL6. P 和 NP 复杂类	P 和 NP 类的定义；NP-完备性 （库克定理）；标准的 NP-完备问题；归约技术
	AL7.自动机理论	确定有限自动机（DFA）；非确定有限自动机（NFA）；DFA 和 NFA 的等价；正则表达式；正则表达式的缩胀定理；下推自动机（PDA）；PDA 和上下文无关文法的关系；上下文无关文法的性质；图灵机；非确定图灵机；集合和语言；乔姆斯基（Chomsky）层次结构；丘奇——图灵论题
	AL8. 高级算法分析	缓冲分析；在线（on-line）和离线（off-line）算法；随机算法；动态程序设计；组合优化
	AL9.密码算法	密码学的历史概述；密钥密码学和密钥交换问题；公钥密码学；数字签名；安全协议；应用（零知识证明、认证, 等）
	AL10.几何算法	线段（性质、交点）；凸包查找算法
	AL11. 并行算法	PRAM 模型；互斥和并行读写；指针跳转；布伦特（Brent）定理和运行效率
4. （AR）体系结构和组织（共 36 个核心小时）	AR1. 数字逻辑和数字系统（6 小时）	计算机体系结构的概况和历史；基本构件（逻辑门、触发器、计数器、寄存器、可编程逻辑阵列）；逻辑表达式、化简、与或式（Sum of Product Forms）；寄存器传输符号表示；物理事项的考虑 （门延迟、扇入、扇出）
	AR2. 数据的机器表示（3 小时）	位、字节和字；数值数据表示和数字基础；定点和浮点系统；有符号和双补码表示；非数值数据的表示（字符代码、图形数据）；记录和数组的表示

	AR3.汇编级机器组织（9 小时）	冯•诺依曼机器的基本组织；控制单元、指令的获取、解码和执行；指令集和类型（数据操纵、控制、I/O）；汇编/机器语言程序设计；指令格式；寻址方式；子程序调用和返回机制；I/O 和中断
	AR4.存储系统组织和体系结构（5 小时）	存储系统及其技术；编码、数据压缩和数据完整性；存储器的层次；主存储器的组织和操作；延迟、周期时间、带宽和交叉存取；高速缓存（地址映射、块的大小、替换和存储策略）；虚拟存储（页表、TLB）；故障处理和可靠性
	AR5.接口和通信（3 小时）	I/O 基础（握手、缓冲、程序化 I/O，中断驱动的 I/O）；中断结构（向量结构和优先级结构）；外部存储、物理的组织和驱动器；总线（总线协议、仲裁、直接存储器存取（DMA））；网络导引；多媒体支持；RAID 体系结构
	AR6.功能的组织（7 小时）	简单数据通路的实现；控制单元（硬布线实现和微程序实现）；指令流水线；指令级并行（ILP）导引
	AR7.多道处理和预备体系结构（3 小时）	SIMD、MIMD、VLIW、EPIC 导引；脉动体系结构；互联网络（超级立方体结构、混洗互联、平面网格结构、纵横结构）；共享存储器系统；高速缓存的一致性；存储器模型和存储器的一致性
	AR8.性能提高	超级标量体系结构；分支预测；预取；预测执行（Speculative Execution）；多线程；可伸缩性（Scalability）
	AR9.网络和分布式系统的体系结构	局域网和广域网导引；层次协议设计、ISO/OSI、IEEE 802；体系结构问题对分布式算法的影响；网络计算；分布式多媒体
5．（OS）操作系统（共 18 个核心小时）	OS1.操作系统概述（2 小时）	操作系统的作用和目的；操作系统发展的历史；典型操作系统的功能；支持客户——服务器模型的机制、握手设备；设计问题（效率、健壮性、灵活性、可移植性、安全性、兼容性）；安全、组网、多媒体、Windows 的影响
	OS2.操作系统原理（2 小时）	构造方法（整体式的、分层的、模块化的、微内核模型）；抽象、进程和资源；应用程序接口（API）的概念；应用的需要和硬件与软件技术的演化；设备组织；中断：方法和实现；用户/系统态的概念以及核心态的保护和转入
	OS3.并发（6 小时）	状态与状态图；结构（就绪表、进程控制块等）；分派和上下文交换；中断的作用；并发执行（优点和缺点）；“互斥”问题和一些解决方案；死锁（原因、条件和预防）；模型和机制（信号量、监控程序、条件变量、会合点）；生产者—消费者问题和同步；多处理器问题（旋转锁、重入）
	OS4.调度与分派（3 小时）	抢占非抢占调度；调度器和（调度）策略；进程和线程；时限和实时问题
	OS5.存储器管理（5 小时）	物理存储器和存储管理硬件的回顾；覆盖、交换和分区；分页和段；换入和换出策略；工作区和系统颠簸；高速缓存
	OS6.设备管理	并行和串行设备的特点；设备差异的抽象；缓冲策略；直接存储器存取；故障恢复
	OS7. 安全和保护	系统安全概述；策略和机制的分离；安全方法和设备；保护、存取和认证；保护模型；存储保护；加密；恢复管理
	OS8.文件系统	文件（数据、元数据、操作、组织、缓冲、顺序、非顺序）；目录（内容和结构）；文件系统（分区、安装/卸载、虚拟文件系统）；标准实现技术；存储映像文件；专用文件系统；命名、搜索、访问、备份

	OS9.实时和嵌入系统	进程和任务调度；实时环境中的存储/磁盘管理要求；失败、风险和恢复；实时系统中特别需要关注的问题
	OS10.容错	基本概念（可靠和可用的系统）；空间和时间冗余；实现容错的方法；可靠系统的例子
	OS11. 系统性能评价	为什么系统性能需要评价，评价什么；高速缓存、分页、调度、存储管理和安全等策略；评价模型（确定模型、分析模型、仿真模型或特定于具体实现的模型）

续表

主领域	单元	主题
	OS12.脚本	脚本和脚本语言的作用；基本的系统命令；建立脚本、参数传递；执行脚本；脚本对程序设计的影响
6.（NC）网络计算 （共15核心小时）	NC1. 网络计算导引 （2小时）	网络产生的背景、历史以及Internet；网络体系结构；网络计算的范围（网络和协议、网络和多媒体系统、分布式计算、移动和无线计算）
	NC2. 通信和组网（7小时）	网络标准和标准体系；ISO 7层参考模型及其在TCP/IP中的实例化；电路交换和包交换；流和数据包；物理层的网络概念（理论基础、传输介质、标准）；数据链路层概念（成帧、出错控制、流量控制、协议）；网络互联和路由（路由算法、网络互联、拥塞控制）；传输层服务（建立连接、性能问题）
	NC3. 网络安全（3小时）	密码学基础；密钥算法；公开密钥算法；认证协议；数字签名；实例
	NC4. 客户－服务器计算的一个实例：Web（3小时）	Web技术（服务器端程序、常见的网关接口（CGI）程序、客户端脚本、（Java）小程序的概念）；Web服务器的特点（处理请求、文件管理、通常的服务器体系结构的功能）；客户机的角色；客户－服务器关系的本质；Web协议；Web站点创建和Web管理的支持工具；开发Internet信息服务器；发布信息和应用
	NC5. 建立Web应用	应用层协议；Web工程原则；数据库驱动的Web网站；远程调用（RPC）；轻量（light-weighted）分布式对象；中间件的作用；支持工具；分布式对象系统的安全问题；企业中基于Web的应用
	NC6. 网络管理	网络管理问题概述；密码的使用和访问控制机制；域名和名字服务；因特网服务提供者（ISP）的相关事项；安全问题和防火墙；服务质量问题（性能、错误修复）
	NC7. 压缩和解压	模拟和数字表示；编码与解码算法；无损耗和有损耗压缩；数据压缩（Huffman和 Ziv-Lempel 算法）；音频压缩和解压；图像压缩和解压；视频压缩和解压；性能问题（同步、压缩比、用于实时应用时的适用性）
	NC8. 多媒体数据技术	声音和音频、图像和图形、动画和视频；多媒体标准（音频、音乐、图形、图像、电话、视频、电视）；容量规划和性能问题；输入和输出设备（扫描仪、数码相机、触摸式屏幕、声控设备）；MIDI键盘、合成器；存储标准（磁光盘、CD-ROM、DVD）；多媒体服务器和文件系统；支持多媒体开发的工具
	NC9. 无线和移动计算	无线标准的历史、演化和兼容性概述；无线和移动计算的特殊问题；无线局域网和卫星网；无线本地环路；移动因特网协议；扩展客户－服务器模型以适用于移动计算；移动数据访问：服务器数据传播和客户高速缓存管理；支持无线和移动计算的软件包；中间件和支持工具的作用；性能问题；正在出现的技术

续表

主领域	单元	主题
7．（PL）程序设计语言（共21个核心小时）	PL1. 程序设计语言概述（2 小时）	程序设计语言的历史；程序设计模式概述（过程式语言、面向对象语言、函数语言、说明性、非算法语言、脚本语言）；规模对程序设计方法的影响
	PL2.虚拟机（1小时）	虚拟机的概念；虚拟机的层次；中间语言；在不同的机器上运行代码引发的安全问题
	PL3.语言翻译导引（2 小时）	解释器和编译器的比较；语言翻译阶段（词法分析、语法分析、代码生成和优化）；翻译中依赖和不依赖于机器的方面
	PL4.声明和类型（3 小时）	作为具有一系列操作的数值集合的类型的概念；声明模型（绑定、可见性、范围和生命期）；类型检查概述；无用存储单元的收集
	PL5.抽象机制（3小时）	作为抽象机制的过程、函数和迭代器；参数化机制（引用和值）；活动记录和存储管理；类型参数和参数化类型；程序设计语言中的模块
	PL6.面向对象的程序设计（10 小时）	面向对象的设计；封装和信息隐藏；行为和实现的分离；类和子类；继承（重载、动态分派）；多态（子类型多态和继承）；类的层次；集合类和迭带协议；对象的内部表示和方法表
	PL7.函数式程序设计	函数式语言产生的动因及其概述；表、自然数、树以及其他递归定义数据上的递归；闭包和函数作为数据的使用（无限集、流）
	PL8.语言翻译系统	正则表达式在扫描器中的应用；语法分析 （具体和抽象语法、抽象语法树）；上下文无关文法（CFG）在表驱动和递归下降分析法中的应用；符号表管理；攀树式代码生成；和特定体系结构相关的操作：指令选择和寄存器分配；优化技术；支持翻译过程的工具的使用，以及使用工具的好处；程序库和独立编译；建立语法制导工具
	PL9.类型系统	数据类型（基本类型，积、余积类型，代数类型，递归类型，向量（函数）类型，参数化类型）；类型检查模型；用户自定义类型的语义模型（类型缩写、抽象数据类型、类型等价）；参数多态；子类型多态；类型检查算法
	PL10. 程序设计语言语义学	非形式语义学；形式语义学概述；指称语义学；公理语义学；操作语义学
	PL11. 程序设计语言的设计	语言设计的一般原则；设计目标；类型体系；数据结构模型；控制结构模型；抽象机制
8．（HC）人机交互（共8个核心小时）	HC1.人机交互基础（6 小时）	动机（为什么要关心人）、人机接口环境（工具、网络超媒体、通信）；以人为中心的开发和评价；人的表现模型（理解、运动和认知）；人的表现模型（文化、通信和组织）；适应人的多样性；好的设计和优秀设计者的原则；工程折衷；可用性测试导引

续表

主领域	单元	主题
	HC2. 建立简单的图形用户接口（2 小时）	图形用户接口（GUI）的原理；GUI 工具箱
	HC3. 以人为中心的软件评价	确立评价目标；没有用户参与的评价（走查、KLM、方针和标准）；用户评价（可用性测试、协商、考察、实验）

	HC4. 以人为中心软件开发	方法、特点和过程概述；功能和可用性（任务分析、协商、调查；交互和表示方式规格）；原型技术和工具；文件情节串联图版（继承和动态分派、原型语言和 GUI 生成器）
	HC5. 图形用户接口设计	选择交互风格和交互技术；普通窗口小配件的人机接口；屏幕设计的人机接口（布局、颜色、字体、标签）；处理人为失败；简单屏幕设计以外的内容（可视化、表达、比喻）；多模式交互（图形、声音和触觉）；3D 交互和虚拟现实
	HC6. 图形用户接口程序设计	UIMS、对话独立性和分析的层次；窗口小配件类；事件管理和用户交互；几何管理；GUI 生成器和 UI 程序设计环境；跨平台设计
	HC7. 多媒体系统的人机接口（HCI）	信息的分类和结构（层次、超媒体）；信息检索和人的表现（网络搜索、数据库查询语言的可用性、图形学、声音）；多媒体信息系统的 HCI 设计；语言识别和自然语言处理；信息设备和移动计算
	HC8. 协同和通信的人机接口	支持专门任务的群件（文档准备，多个玩家的游戏）；异步组通信（电子邮件、公告牌）；同步组通信（聊天室、召开会议）；联机社区（MUD 与 MOO）；软件特征和智能代理
9.（GV）图形学和可视化计算（共 3 核心小时）	GV1. 图形学的基本技术（2 小时）	图形学软件的层次；使用图形应用程序接口（API）；简单的颜色模型（RGB、HSB、CMYK）；齐次坐标系；仿射转换（比例放缩、旋转、平移）；取景变换；裁剪
	GV2. 图形系统（1 小时）	光栅和向量图形系统；视频显示设备；物理和逻辑输入设备；图形系统开发者面临的问题
	GV3.图形通信	颜色心理动力学和颜色的相互作用；为视力缺陷修正颜色；不同颜色在文化上的意义；对服务特定观众的图像使用有效的伪色彩调色板；为有效理解而构造视图；为有效视频而进行的图像修正和硬拷贝；在颜色或者其他可视数据的关键信息中使用图例 ；在表示上下文和背景信息的图像中使用正文；图形操作的可视化用户反馈
	GV4.几何学的模型	3D 对象的多边形表示；参数化多项式曲线和面；构造立体几何（CSG）表示；曲线和面的绝对表示；空间分割技术；过程模型；可变形模型；细分表面；多解模拟；重构

续表

主领域	单元	主题
	GV5. 基本绘制（rendering）	线产生算法（Bresenham）；字体的产生（轮廓和位图）；光源和材料性质；环境、散射和镜面反射；Phong 反射模型；多边形表面绘制、 平直消隐、Gouraud 消隐和 Phong 消隐；文本影射、凸起文本、环境图；光线跟踪介绍；图像合成、采样技术和图形保真
	GV6.高级绘制	迁移方程；光线跟踪算法；光子跟踪；全局照明计算的辐射度、表格因素；全局照明的有效方法；全局照明的蒙特卡洛（Monte Carlo）方法；基于图像的绘制、全景浏览、视觉功能建模；复杂自然现象的描绘；非逼真绘制
	GV7.高级技术	颜色的量化；2D 原语扫描变换、正向差分；曲面的棋盘形布局；隐藏表面的消除方法；Z-缓冲和框架缓冲、颜色通道（暗通道）；高级几何建模技术
	GV8. 计算机动画	关键帧动画；摄影动画；脚本系统；分级结构动画（正向和逆向运动）；动作捕捉；过程动画；变形

	GV9.可视化	基本的可视化查看和询问功能；矢量域、张量和流数据的可视化；标量域或高度域可视化（步进立方体法等值面（Isosurface））；直接实体数据绘制（射线造型法、转换函数、分段、硬件）；信息可视化（工程和并行协同方法）
	GV10.虚拟现实	立体显示；强制反馈仿真、触觉设备；浏览跟踪；冲突检测；可见性计算；有严格时间要求的绘制、多层细节（LOD）；基于图像的虚拟现实系统；分布式虚拟现实、计算机网络协作；交互式建模；用户接口问题；在医学、仿真和训练中的应用
	GV11. 计算机视觉	图像获取；数字图像及其性质；图像预处理；分段（阈值处理、基于边界和区域的分段）；轮廓表示和对象识别；运动分析；实例研究（对象识别、对象跟踪）
10．（IS） 智能系统（共10核心小时）	IS1.智能系统的基本问题（1小时）	人工智能的历史；哲学问题（图灵（Turing）检验、西尔勒（Searle）的“中文屋子”思维实验、人工智能中的道德问题）；基本定义；哲学的问题；为世界建模；启发式方法的作用
	IS2.搜索和约束满足（5小时）	问题空间；蛮力搜索（广度优先、深度优先、迭代深入的深度优先）；最佳优先搜索（普通最佳优先、Dijkstra算法、A*、A*的可采纳性）；二人游戏（极大极小搜索、α～β剪枝法）；约束满足（回溯和局部搜索方法）
	IS3.知识表示和推理（4小时）	命题和谓词逻辑回顾；归结和定理证明；非单调推理；概率推理；贝叶斯（Bayes）定理
	IS4.高级搜索	基因算法；模拟退火算法；局部搜索算法

续表

主领域	单元	主题
	IS5.高级知识表示和推理	结构化表示（框架和对象、描述逻辑、继承系统）；非单调的推理（非经典逻辑、缺省推理、信念修正、优先逻辑（Preference logics）、知识源的综合、冲突信念聚合、行动和变化推理（Reasoning on Act and Change）、情形演算、事件演算、分支问题）；时态和空间推理；不确定性（概率推理、贝叶斯网络、模糊装置和概率论、决策理论）；诊断知识表示、定性表示
	IS6.代理	代理的定义；成功的应用和基于代理的系统现状；代理体系结构（简单的交互代理、交互规划器、层次体系结构、实例结构和应用）；代理理论（承诺、意图、决策理论代理、马尔可夫决策过程（MDP））；软件代理、个人助手和信息访问（协同代理、信息收集代理）；可信赖代理（合成字符、在代理中模拟情感）；学习代理；多代理系统（协同代理、代理组、代理模型、多代理学习）；机器人代理导引；移动代理
	IS7. 自然语言处理	确定和随机文法、语法分析算法、基于语料库的方法、信息检索、语言翻译、语音识别
	IS8.机器学习和神经网络	机器学习的定义和实例；监督式学习；学习决策树；学习神经网络；学习信念网络；最近邻居算法；学习理论；过份适合的问题；非监督学问；加强式学习

	IS9.人工智能规划系统	规划系统的定义和实例；作为搜索的规划；基于算符的规划；命题规划；扩展规划系统（基于事实的、学习和概率系统）；静态世界规划系统；规划与执行；规划和机器人学
	IS10.机器人学	概述（机器人系统现状、规划和交互控制、控制中的不确定性、检测、（现实）世界模型）；配置空间；规划；检测；机器人程序设计；导航和控制
11.（IM）信息系统（共10个核心小时）	IM1.信息模型与信息系统（3小时）	信息系统的历史和发展动力；信息存储和检索（IS&R）；信息管理应用；信息获取和表示；分析和索引；搜索、恢复、连接、导航；信息隐私、完备、安全和表示可缩放性、效率和有效性
	IM2.数据库系统（3小时）	数据库系统的历史和起因；数据库系统组成；数据库管理系统（DBMS）功能；数据库结构和数据独立性；数据库查询语言的使用
	IM3.数据建模（4核心）	数据建模；概念模型（包括实体－关系模型和UML）；面向对象模型；关系数据模型
	IM4.关系数据库	把概念模式映象成关系模式；实体和参照完整性；关系代数和关系演算
	IM5.数据库查询语言	数据库语言简介；SQL （数据定义、查询公式、更新子语言、约束、完整性）；查询优化；仿效实例查询（QBE）和第四代环境；过程语言中的嵌入式非过程查询；对象查询语言导引

续表

主领域	单元	主题
	IM6.关系数据库设计	数据库设计；函数依赖；范式 （1NF、2NF、3NF、BCNF）；多值依赖（4NF）；连接依赖（PJNF、5NF）；表示理论
	IM7.事务处理	事务；失败与恢复；并发控制
	IM8.分布式数据库	分布式数据存储；分布式查询处理；分布式事务模型；并发控制；同构和异构方案；客户－服务器
	IM9.物理数据库设计	存储与文件结构；索引文件；哈希文件；签名文件；B-树；密集索引文件；变长记录文件；数据库效率和协调
	IM10.数据挖掘	数据挖掘的有用性；关联和顺序模式；数据分簇；购物篮分析（Market Basket Analysis）；数据清洗；数据可视化
	IM11.信息存储与检索	字符、字符串、编码、文本；文件、电子出版、标记和标记语言；特里（Tries）、逆序文件、帕特（PAT）树、签名文件、索引；语形学分析、词干、词组、停止列表；检索词频率分布、不确定性、模糊性、加权；矢量空间、概率、逻辑和高级模型；信息需求、关联、评价、效能；主题词表、实体、分类、元数据；文献信息、文献计量学、引用；路由选择和（社区）过滤；搜索和搜索策略、信息搜寻行为、用户建模、反馈；信息汇总与可视化；引用、关键词、分类方案和其他术语的综合；协议和系统（包括Z39.50、OPAC、WWW引擎、研究中的系统）
	IM12.超文本和超媒体	超文本模型（早期的历史、Web、Dexter、Amsterdam、HyTime）；链接服务、引擎和（分布式）超文本结构；结点、尺寸、单元、位置、范围；浏览、导航、视图、放缩；自动链接产生；表示、转换、同步；写作、阅读和注释；协议和系统（包括Web、HTTP）

	IM13.多媒体信息与多媒体系统	设备、设备驱动程序、控制信号和协议、DSP（数字信号处理）；应用、媒介编辑器、写作系统和写作；流/结构、捕捉/表示/转换、空间/区域、压缩与编码；基于内容的分析，索引、音频、图像和视频搜索；表示、绘制、同步、多模式集成和接口；实时发送、服务质量、音频/视频会议、视频点播
	IM13.多媒体信息与多媒体系统	设备、设备驱动程序、控制信号和协议、DSP（数字信号处理）；应用、媒介编辑器、写作系统和写作；流/结构、捕捉/表示/转换、空间/区域、压缩与编码；基于内容的分析，索引、音频、图像和视频搜索；表示、绘制、同步、多模式集成和接口；实时发送、服务质量、音频/视频会议、视频点播
	IM14. 数字图书馆	数字化、存储和交换；数字对象、合成物和包；元数据、分类、作者提交；命名、储存处、文件夹；空间（概念空间、地理空间、2/3D、VR）；体系结构（代理、总线、互操作性）；服务（搜索、连接、浏览等）；知识产权管理、隐私、保护（水印）；归档与保存、完整性

续表

主领域	单元	主题
12.（SP）社会和专业的问题（共16个核心小时）	SP1.计算的历史（1小时）	史前史——1946年以前的世界；计算机硬件、软件和网络的历史；计算先驱
	SP2.计算的社会背景（3小时）	介绍计算的社会内涵；网络的社会内涵；互联网的增长、控制和访问；与性有关的问题；国际事务
	SP3.分析的方法和工具（2小时）	道德争论及其评价；确定和评价道德选择；理解设计的社会背景；确定假设和价值观
	SP4.职业和道德责任（3小时）	我们依赖的社会价值和法律；职业化的本质；各种形式的专业文凭及其优劣；公共政策中职业人员的作用；保持对后果的清醒认识；道德上的异义和检举；道德、行为和实践规范（IEEE、ACM、SE、AITPD等）；处理骚扰和歧视；实践中计算的“可接受使用”政策
	SP5.以计算机为基础的系统的风险和责任（2小时）	软件风险的案例（如：Therac-25）；软件复杂性的意义；风险评估与管理
	SP6.知识产权（3小时）	知识产权的基础；版权、专利和商业秘密；软件版权的侵犯；软件专利；知识产权的国际问题
	SP7.隐私和公民自由（2小时）	隐私保护的道德和法律基础；大型数据库系统的隐私的意义；隐私保护的技术策略；电脑空间中的言论自由；国际意义和在不同文化中的意义
	SP8.计算机犯罪	计算机犯罪的历史和实例；“黑客行为”及其作用；病毒、蠕虫和特洛伊木马；犯罪预防的策略
	SP9.计算中的经济事务	垄断和它的经济上的含义；熟练劳动力的供求对计算产品质量的影响；计算领域的定价策略；访问计算机资源的差异及由此产生的不同效果
	SP10.哲学框架	哲学框架，特别是功利主义和道德理论；道德的相对论问题；从历史的角度看科学道德；科学方法和哲学方法的不同
13.（SE）软件工程（共31个核心小时）	SE1. 软件设计（8小时）	基本设计概念和原则；软件体系结构；结构化设计；面向对象的分析和设计；构件级设计；重用设计
	SE2.使用 API（5小时）	API程序设计；类浏览器和相关工具；根据实例进行程序设计；API环境中的调试；基于构件的计算导引

	SE3.软件工具和环境（3 小时）	程序设计环境；需求分析和设计建模工具；测试工具；配置管理工具；工具集成机制
	SE4.软件过程（2 小时）	软件生命周期和过程模型；过程评估模型；软件过程指标
	SE5. 软件需求与规格（4 小时）	需求的抽取；需求分析和建模技术；功能和非功能的需求；原型；形式化规格技术的基本的概念
	SE6. 软件的验证（3 小时）	验证规划；测试的基本原则，包括测试计划的产生和测试用例的产生；黑盒和白盒测试技术；单元、综合、验证和系统测试；面向对象的测试；检查

续表

主领域	单元	主题
	SE7. 软件演化（3 小时）	软件维护；可维护软件的特点；再生工程（Reengineering）；遗产系统（Legacy systems）；软件重用
	SE8.软件项目管理（3 小时）	队伍管理（队伍的组织和决策、软件队伍的作用和责任、角色确定和分配、项目跟踪、队伍问题的解决）；工程调度；软件测量和评价技术；风险分析；软件质量保证；软件配置管理；项目管理工具
	SE9. 基于构件的计算	基本问题（构件的定义和本质、构件和接口、构件的好处）；基本技术（构件设计和组织、与客户－服务器模型和模式的关系、对象的使用和对象生命周期服务、对象代理的使用、编组）；应用（包括移动构件的使用）；基于构件系统的体系结构；面向构件设计；事件处理：检测、通知和反应；中间件（中间件中的面向对象范例、对象请求代理、事务处理监视器、工作流程（Workflow）系统、当前的最新工具）
	SE10. 形式化方法	形式化方法的概念；形式化规格语言；可执行和非可执行规格；前后断言；形式化验证
	SE11.软件可靠性	软件可靠性模型；冗余和容错；缺陷分类；概率分析方法
	SE12. 专用系统开发	实时系统；客户－服务器系统；分布式系统；并行系统；基于网络的系统；高度集成的系统
14．（CN）计算科学和数值计算方法 （无核心学时）	CN1.数值分析	浮点算术；误差、稳定性和收敛；泰勒级数；根的循环求解（牛顿方法）；曲线拟合；函数逼近；数值微分和积分（Simpson 规则）；显式和隐式方法；微分方程（欧拉方法）；线性代数；有限差分
	CN2.运筹学	线性程序设计（整数程序设计、单纯法（Simplex method））；概率模型；排队理论（Petri 网、马尔可夫模型和马尔可夫链）；优化；网络分析和路由算法；预测和评估（决策分析、预告、 风险管理、经济计量学、微观经济学、敏感度分析）；动态程序设计；试验性应用；软件工具
	CN3.建模与仿真	随机数（伪随机数的产生和测试、蒙特卡洛（Monte Carlo）方法、分布函数导引）；退火模型（离散事件模拟、连续模拟）；仿真模型的验证和确认（输入分析、输出分析）；排队理论模型试验性应用
	CN4.高性能计算	高性能计算导引（计算科学的历史和重要性、应用领域概述、所要求技巧的回顾）；高性能计算（处理器体系结构、高性能的存储器系统、输入/输出设备、流水线（pipelining）、并行语言和体系结构）；科学可视化（结构表示、数据格式、可视化工具和工具包）；部分问题（海洋和大气模型、地震波模型、N-体系统（Barnes-Hut 算法）、化学反应、阶段转移、流体）

参考文献

1 董荣胜，古天龙，蔡国永，谢春光．计算机科学与技术方法论．计算机科学，2002，29（1）

2 董荣胜．计算教育哲学初探．计算机科学，2000，27（1）

3 董荣胜．向学术界推荐一个认知计算学科的工具——计算机科学与技术方法论（大会报告）．上海：新世纪计算机教育及 CC2001 研讨会，2001.7

4 董荣胜．计算教育哲学．计算机科学，1999，26（10，增刊）

5 董荣胜，古天龙．计算教育的三个重大问题．计算机科学，2002，29（7，增刊）

6 董荣胜．关于计算机科学技术的方法论（大会发言）．上海：全国高校计算机学科教改研讨会，1998.10

7 董荣胜．对象—关系数据库的科学技术方法论（会议交流论文）．上海：全国高校计算机学科教改研讨会，1998.10

8 董荣胜．《数据库系统概论》课程改革的实践与探讨．1996 计算机教育学术年会论文集．北京：电子工业出版社，1996

9 董荣胜，古天龙．计算机科学与技术方法论——一个认知计算学科的工具（讲义）．桂林电子工业学院，2001.11

10 Tianlong Gu，Parisa A Bahri．A survey of Petri net applications in batch processes． Computers in Industry，2002，47（1）

11 古天龙．形式化技术及其工业应用：现状与展望．桂林电子工业学院学报， 2000，20（4）

12 冯著明．美国《1991 计算教程》特点的综合分析．桂林电子工业学院学报， 1993，13（增刊）

13 Denning P J，et al．Computing as a discipline．Communications of the ACM，1989，32（1）

14 Turner A J，et al．A Summary of the ACM/IEEE-CS Joint Curriculum Task Force Report：Computing curricula 1991．Communications of the ACM，1991，34（6）

15 http://www.computer.org/education/cc2001:Computing curricula 2001

16 Carl K Chang．Curricula 2001：Bringing the Future to the Classroom． Computer，1999，32（9）

17 Denning P J．A debate on teaching computing science．Communications of the ACM，1989，32（12）

18 Dijkstra E W．On the cruelty of really teaching computing science．Communications of the ACM，1989，32（12）

19 Scherlis W L，et al．Colleagues respond to dijkstra’s comments．Communications of the ACM，1989，32（12）

20 Dijkstra E W. Dijkstra's reply to comments. Communications of the ACM, 1989, 32（12）
21 Searle J R. Minds, brains, and programs. Behavioral and Brain Sciences, 1980, Vol. 3
22 Turing A M. Computing machinery and intelligence. Mind, 1950, Vol. LIX
23 Computing Sciences Accreditation Board Inc. Annual Report, Stamford, Conn, 1998
24 Adleman L M. Molecular Computation of Solutions to Combinatorial Problems. Science, 1994
25 http://www.c3.hu/scca/butterfly/Kunzel/synopsis.html:The Birth of the MACHINE:Raymundus Lullus and His Invention
26 Dijkstra E W. Hierarchical Ordering of Sequential Processes. Acta Informatica pages, 1971
27 Edsger W Dijkstra. Go to statement considered harmful. Communications of the ACM, 1968, 11（3）
28 BÖhm, Corrado, and Jacopini Guiseppe. Flow diagrams, Turing machines and languages with only two formation rules. Communications of the ACM, 1966, 9（5）
29 Knuth D E. Structured programming with goto statements. ACM Computing Surveys, 1974, 6（4）
30 http://www.cs.berkeley.edu/~huebsch/cs267/chess.html:Utilizing Parallel Computingto Play a Better Game of Chess
31 http://www.mark-weeks.com/chess/97dk$$.htm:IBM Challenge Rematch 1997 Deep Blue – Kasparov Highlights
32 Durndell, Alan, Glisson, et al. Gender and Computing:Persisting Differences. Educational Research, 1995, 37（3）
33 Chungang. Theoretical Models of Whistleblowing:An Individual Perspective. Journal of Social Sciences, 1998
34 Kitchener K. Institution, critical evaluation and ethical principles: Foundation of ethical decisions in counseling psychology. The Counseling Psychologist, 1984, 12（3）
35 Banker R D, Kauffman R J, and Zweig D. Repository evaluation of software reuse. IEEE Transactions on Software Engineering, 1993, 19（4）
36 Ploman E W, and Hamilton L C. Copyright: Intellectual Property in the Information Age. London: Routledge & Kegan Paul, 1980
37 Hofmeyr S A, Somayaji A, and Forrest S. Intrusion Detection using Sequences of System Calls. Journal of Computer Security, 1998
38 Gotterbarn D, et al. Computer society and ACM Approve Software Engineering Code of Ethics. Computer, 1999. 32（10）
39 Gotterbarn D, et al. Software Engineering Code of Ethics, Version 3.0, Computer, 1997, 20（11）: 88-92
40 国家教委社会科学研究与艺术教育司. 自然辩证法概论. 修订版. 北京：高等教育

出版社，1991
41 赵致琢．关于计算机科学与技术认知问题的研究简报．计算机研究与发展，2001，38（1）
42 赵致琢．计算科学导论．第二版．北京：科学出版社，2000
43 解恩泽，刘永振，谢树智．科学问题集．湖南：湖南科学技术出版社，1999
44 朱新民．科学争论集．湖南：湖南科学技术出版社，1999
45 王浩．数理逻辑通俗讲话．北京：科学出版社，1983
46 吴允曾选集．北京：北京科学技术出版社，1991
47 莫绍揆．递归论．北京：科学出版社，1987
48 张效祥．计算机科学技术百科全书．北京：清华大学出版社，1998
49 吴鹤龄，崔林．ACM 图灵奖—计算机发展史的缩影．北京：高等教育出版社，2000
50 李文林．数学珍宝—历史文献精选．北京：科学出版社，1998
51 徐利治．现代数学手册：计算机数学卷．武汉：华中科技大学出版社，2001
52 徐利治．数学方法论选讲．第三版．武汉：华中理工大学出版社，2000
53 朱宝荣．现代心理学方法论研究．上海：华东师范大学出版社，1999
54 张楚廷．数学文化．北京：高等教育出版社，2000
55 方延明．数学文化导论．南京：南京大学出版社，1999
56 林尧瑞，郭木河．人类智慧与人工智能．北京：清华大学出版社，2001
57 胡正国，蔡经球．程序设计方法学．修订本．西安：西北工业大学出版社，1997
58 缪淮扣．计算机的灵魂——程序．北京：清华大学出版社，2001
59 萨师煊，王珊．数据库系统概论．第二版．北京：高等教育出版社，1991
60 李学干，苏东庄．计算机系统结构．西安：西安电子科技大学出版社，1994
61 陆汝钤．计算机语言的形式语义．北京：科学出版社，1992
62 侯敏．计算机语言学与汉语自动分析．北京：北京广播学院出版社，1999
63 赵总宽，陈慕泽，杨武金．现代逻辑方法论．北京：中国人民大学出版社，1998
64 朱慧真．80x86 汇编语言教程．北京：清华大学出版社，1995
65 D E 克努特．计算机程序设计技巧：第一卷，基本算法．管纪文，苏远霖译．北京：国防工业出版社，1980
66 T M 沃克．计算机科学基础．周士观，刘国香，刘棠等译．北京：科学出版社，1983
67 谭浩强．C 语言程序设计．第二版．北京：清华大学出版社，1999
68 王仲春，李元中，顾莉蕾等．数学思维与数学方法论．北京：高等教育出版社，1989
69 严蔚敏，吴伟民．数据结构．北京：清华大学出版社，1987
70 江涛．数据结构．北京：中央广播电视大学出版社，1993
71 Kenneth H Rosen．离散数学及其应用． 袁崇义，屈婉玲，王捍贫等译．北京：机械工业出版社，2002
72 Richard Johnsonbaugh．离散数学．王孝喜，邵秀丽，朱思俞译．北京：电子工业出版社，1999
73 Micheal Sipser．计算理论导引．张立昂，王捍贫，黄雄译．北京：机械工业出版社，2000
74 亚历山大洛夫．数学——它的内容、方法和意义．孙小礼等译．北京：科学出版社，

2001
75 李铁木．数学与哲学．北京：科学出版社，1999
76 邹海明，余祥宣．计算机算法基础．武汉：华中理工大学出版社，1985
77 F S 贝克曼．程序设计的数学基础．曹德和，吴延佳译．北京：科学出版社，1991
78 王元元．计算机科学中的现代逻辑学．北京：科学出版社，2001
79 许国志．系统科学．上海：上海科技教育出版社，2000
80 许国志．系统科学与工程研究．第二版．上海：上海科技教育出版社，2000
81 刘大椿．科学技术哲学导论．北京：中国人民大学出版社，2000
82 谢希仁．计算机网络．第二版．大连：大连理工大学出版社，1996
83 Peter Coad，Edward Yourdon．面向对象的分析．邵维忠，廖钢城，李力译．北京：北京大学出版社，1992
84 Peter Coad，Edward Yourdon．面向对象的设计．邵维忠，廖钢城，苏渭珍译．北京：北京大学出版社，1994
85 杨文龙，姚淑珍，吴芸．软件工程．北京：电子工业出版社，1999
86 周之英．现代软件工程：基本方法篇．北京：科学出版社，2000
87 宁远方，成栋．管理信息系统．北京：中国人民大学出版社，1999
88 姜同强．计算机信息系统开发——理论、方法与实践．北京：科学出版社，1999
89 袁崇义．Petri 网原理．北京：电子工业出版社，1998
90 陈有祺．形式语言与自动机．天津:南开大学出版社，1999
91 И А 阿波京，Л Е 梅斯特洛夫．计算机发展史．张修译．上海：上海科学技术出版社，1984
92 王智联．计算机导论．北京：北京航空航天大学出版社，1996
93 杨力平．计算机犯罪与防范．北京：电子工业出版社，2002
94 聂元铭，丘平．网络信息安全技术．北京：科学出版社，2001
95 俞嘉惠．网络啊，网络．北京：清华大学出版社，2001
96 黄济．教育哲学．北京：北京师范大学出版社，1985
97 王坤庆．现代教育哲学．武汉：华中师范大学出版社，2000