计算机理论基础与应用丛书

数理逻辑基础与粒计算

闫　林 /著

科学出版社
www.sciencep.com

计算机理论基础与应用丛书

数理逻辑基础与粒计算

闫 林 著

科 学 出 版 社

北 京

内 容 简 介

本书由两部分内容组成。前四章作为第一部分，讨论了数理逻辑的基础知识，其中包括经典命题演算、经典谓词演算和非经典的模态逻辑，讲解的特点是始终贯穿形式推理在自然推理系统与公理系统中相互等价这条主线。第二部分由后两章构成，内容基于逻辑知识之上，是对第一部分内容的应用和扩展，由作者近年的科研成果作为支撑，其中包括采用逻辑方法对粒和粒计算的形式化、粒空间中基于粒计算的粒语义推理、粒计算与逻辑推理相互融合的讨论、粒计算的应用等。

本书面向从事计算机科学、自动控制及相关专业的科研人员和科技工作者，特别是从事人工智能、粒计算研究的读者。本书可作为研究生的教材或阅读材料，也可供本科高年级学生阅读。

图书在版编目(CIP)数据

数理逻辑基础与粒计算/闫林著. —北京：科学出版社，2007
(计算机理论基础与应用丛书)
ISBN 978-7-03-019828-0

Ⅰ.数… Ⅱ.闫… Ⅲ.①数理逻辑②电子计算机-算法理论 Ⅳ.O141
TP301.6

中国版本图书馆 CIP 数据核字(2007)第 134847 号

责任编辑：王志欣 张海娜 杨 然 / 责任校对：包志虹
责任印制：刘士平 / 封面设计：王 浩

科 学 出 版 社 出版
北京东黄城根北街 16 号
邮政编码：100717
http://www.sciencep.com

源海印刷有限责任公司 印刷

科学出版社发行 各地新华书店经销

*

2007 年 8 月第 一 版 开本：B5(720×1000)
2007 年 8 月第一次印刷 印张：17
印数：1—3 000 字数：327 000

定价：36.00 元

(如有印装质量问题，我社负责调换〈新欣〉)

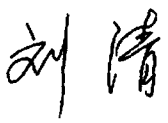
序

当今科技的发展，促进计算机科学研究的深入，这自然伴有新问题的产生，粒计算就是在这种背景下形成的一个新的研究学科。它是一种处理问题的方法学，其思想就是将大、复杂和困难的原始求解问题拆开、分解成小问题的求解，并将这些小问题的求解过程合并成原始整体问题的解。这些操作过程及实现方法很大程度上以数理逻辑作为理论基础，所以数理逻辑和粒计算相互渗透、密切相关。例如，不少学者正在从事 Rough 逻辑、决策逻辑、信息逻辑和粒逻辑的研究，这些都为粒计算提供了研究方法。因此，自粒计算提出发展到今天，虽然只有短短的十几年，但得到国内外许多优秀学者的广泛重视。特别是许多从事逻辑学研究的学者，给予了粒计算特别的关注，因为这种问题的处理方法，正好为 AI 中问题的求解提供了新的思路。

该书以数理逻辑知识为依托，总结了作者近年的科学研究成果，其中包括采用逻辑方法对粒和粒计算的形式化、粒空间中基于粒计算的粒语义推理、粒计算与逻辑推理的相互融合、粒计算的应用等。究竟如何开展粒计算的研究，学者们的理念和做法自然是见仁见智，研究成果的着重点也各具特色，或注重理论上的探讨，或面对实际问题的应用。无论哪种研究，方法存在着共识，那就是理论研究和应用开发都是以逻辑学或代数学为理论工具。

该书针对粒计算的研究就是依托于逻辑方法中两个重要的方面：其一，数理逻辑的形式化方法；其二，数理逻辑所实现的各种推理。把形式化的方法和逻辑推理作为基础用于粒计算的研究形成了本书的理论体系。

由于粒计算是计算机科学近年来发展的新兴学科，新研究方法的出现是大家的热切期待。书中关于粒计算的讨论，内容新颖、概念清楚、推理严谨。以逻辑方法为基础的粒计算形式化概念和形式化推理或许将给粒计算的研究带来新的思想，读者可从中汲取营养、获得启发、开阔思路、取得进展。相信该书的出版将对我国的粒计算以及其他软计算的研究与应用起到积极的促进作用。



南昌大学教授

2007 年 6 月

前　言

随着信息科学和技术的发展，各领域中信息总量不断扩充，信息组合的复杂度进一步增大，造成数据分析、数据推理和数据查询等方面的极度困难。为了寻求解决的方法，一些学者便提出将各领域中数据合理分类，从整体中分离出部分，通过部分简化整体，进而利用一定手段，将部分进行有效的组合，形成从整体到部分、由部分到科学组合的数据处理方法，专家将之称为粒计算（granular computing）。其中粒表示从整体中分离出的部分，计算表示粒之间的关系、组合、推理和运算等。这种描述是对大部分学者关于粒计算直观理解和认识的总结，是原始和朴素的思想。这种思想源于模糊数学的创建者 Zadeh，他在 1979 年首先提出了粒度的概念，随后又有专家学者在各自的研究中体现出了粒或粒度的思想。Zadeh 曾希望把粒度的研究寓于粒数学之中，但什么是粒数学，以及如何发展也许只有 Zadeh 知晓。1997 年 Lin 借鉴 Zadeh 的思想提出了粒计算的研究课题，该名称不仅得到 Zadeh 的赞同，同时也引起众多学者的兴趣，使得粒计算很快成为计算机科学研究的热点。因此研究者不仅在探索将直观意义下粒计算形式化的方法，而且积极展开理论和应用方面的研究，这当然就促成一系列开创性成果的产生。特别随着近年来粒计算国际学术会议和国际论坛的召开，更使粒计算的研究进入到新的阶段。然而粒计算毕竟是新的研究方向，不同的成果均带有研究者自身领域的痕迹，一般难以形成具有概括性的理论体系。但这正说明作为新研究方向的粒计算具有广阔的探究空间，加之一些专家提出利用逻辑、代数或拓扑等方法展开研究的思想，所以针对粒计算的研究必定充满希望和活力。由于作者对数理逻辑基础知识及其分支进行过专门的学习，并且多年从事计算机专业研究生数理逻辑基础的教学，所以对数理逻辑知识具有一定深度的认识和掌握。特别近年来将逻辑方法用于粒计算的研究，形成了数理逻辑与粒计算相互融合的研究方法和相对独立的研究途径，得到了相应的研究成果。本书的目的之一就是要对数理逻辑与粒计算相互融合的研究进行系统化的整理。

由于书中针对粒计算的研究基于逻辑知识之上，并涉及经典和非经典逻辑的相关内容，因此数理逻辑在粒计算研究中的重要性不言而喻，以至形成数理逻辑与粒计算相互融合、互相渗透的研究方法。再者，由于数理逻辑本身可以作为计算机科学基础理论和实际应用的重要支撑，所以学习、了解和掌握逻辑知识对于从事计算机科学的理论研究，推动计算机科学应用的发展都具有重要的意义。因此，书中前半部分对经典数理逻辑的主要内容和非经典的模态逻辑进行了讨论，

分析力求透彻，讲解追求细腻。特别关于形式推理的讨论，尽管主要在自然推理系统中讲解，但论述过程都将自然推理系统与公理系统形式推理的等价性贯穿始终，这有助于读者对形式推理内在涵义的充分理解。不过书中所选内容均是与计算机科学最直接相关、非逻辑专业学者也最感兴趣的数理逻辑基础知识的部分，并不涉及建立在基础知识之上的数理逻辑四大分支（集合论、递归论、模型论、证明论）的问题，这些高深莫测的逻辑分支往往也使从事数学研究的学者心生畏惧。所以书中逻辑方面的部分适合非数理逻辑专业的读者学习，加之推理的讲解渗入了计算机程序化的思想，因此论述过程更着重计算机科学及相关专业方面的考虑。而粒计算是近年来计算机科学研究的热点，再结合专家提出利用逻辑方法研究粒计算的思想，所以为逻辑方法引入粒计算做准备才是书中讨论数理逻辑知识的真正目的。

书中后半部分基于逻辑知识之上，是对前半部分内容的应用和扩展，以近年来作者的科研成果作为支撑。其中包括采用逻辑方法对粒和粒计算的形式化、粒空间中基于粒计算的粒语义推理、粒计算与逻辑推理相互融合的讨论、粒计算的应用等。这些讨论突显了数理逻辑知识的基础地位，并体现了数理逻辑与粒计算之间相互推进的研究理念。整体内容统一在本书引入的粒空间中，通过粒计算与各类问题在粒空间中的相互融合，为粒计算的研究创建了方法，也显示了这些方法的独特性。但独特性并不掩盖对粒和粒计算讨论的概括性，因为分析表明本书对粒和粒计算的形式化具有抽象统一性，即其他针对粒计算的许多讨论都是形式化粒计算的特殊情况。这是本书关于粒计算研究的重要结论，也是作者期望达到的目的。

纵览全书，作者以数理逻辑基础知识作为开始，以逻辑方法与粒计算相互融合的讨论作为结束。逻辑是作者多年学习、教学和研究的内容，而数理逻辑与粒计算的融合是作者近年来关注和研究的方向。特别需要指出的是在将数理逻辑用于粒计算的研究过程中得到了刘清教授的支持和帮助，深受刘清教授研究思想的影响，谨向刘清教授表示谢意。

由于粒计算是一个新的研究课题，所以将数理逻辑与粒计算的融合自然属于探索性的研究。尽管取得了一定进展，并创建了一些方法，但这些成果都是初步的，也难免存在一些疏漏和不足，欢迎大家批评、指正，同时也希望能与各方学者相互交流，共同推进粒计算研究的发展。

目　录

第 1 章　经典命题演算

经典命题演算(classically propositional calculus)也称为经典命题逻辑(classically propositional logic),是研究命题之间运算和命题之间推理的理论。经典命题演算对命题之间推理的研究是通过形式化或符号化的方法进行的,是对数学中采用自然语言对前提与结论进行推理的抽象。因此,这种形式化的推理方法实际上是为人们通常推理习惯所构建的一种数学模型。也许这就是为什么计算机科学要将数理逻辑作为学习和研究对象的根本原因,因为只有构建了推理的数学模型,才有可能实现推理的程序化。然而,本章关于经典命题演算和第 2 章将要讨论的经典谓词演算的基础理论早在一百多年前已经建立,而计算机的出现只有大约六七十年的历史。所以当初数学家创建经典数理逻辑的基础理论时并不是考虑计算机方面的应用,而是出于对自然语言描述命题的精确化和对数学中的证明给予严格定义的考虑而建立起的一套理论体系。但是随着计算机的出现和计算机科学的发展,不仅使计算机科学家看到了数理逻辑研究的推理和进行的逻辑运算是建立许多计算机算法的数学模型,而且通过数学家、逻辑学家以及计算机科学家不懈的努力和深入的研究,扩展了数理逻辑基础部分的研究领域,推动了数理逻辑基础理论的发展,得到了许多非经典的数理逻辑系统,丰富了计算机科学的理论体系。实际上,如果站在数学的角度考虑数理逻辑,或以纯数理逻辑的观点出发考虑数理逻辑,不把数理逻辑与计算机科学相联系,那么除了本书将要介绍的经典命题演算、经典谓词演算、模态逻辑系统和本书不涉及的众多非经典逻辑系统外,数理逻辑还包括“四论”:集合论、递归论、模型论和证明论。这“四论”都是深奥和完整的数理逻辑分支,并且都有重要、完美和丰富的研究成果。特别值得提及的是,针对集合论的研究,具有数理逻辑研究值得骄傲的成果出现:1963 年前后,美国年轻的逻辑学家柯恩(Cohen)证明了“连续统假设(著名数学难题)”是独立于 ZF 公理系统的,为此 1966 年柯恩获得了享有数学诺贝尔之称的菲尔兹奖。这些抽象和深奥的数理逻辑分支如同常人攀登珠峰一样,使从事计算机科学等其他专业的学者望而却步。但除了“四论”外,数理逻辑还有很大一部分内容,它们属于数理逻辑的基础知识,从事计算机科学和其他一些专业的学者所感兴趣的正是这部分内容。我们将要讲授的知识就属于此范围,因此下面提及的数理逻辑均包含在基础知识的范围内。本章所讨论的经典命题演算是数理逻辑中的最基础的知识,属于经典逻辑部分,要想完全弄懂经典数理逻辑和非经典数理逻辑,必须从经典命题演算的学习开始。下面将经典命题演算简称为命题演算(propositional calculus)或命题逻辑

(propositional logic)。

1.1 命　题

1. 元语言和形式语言

自然语言(如汉语、英语、法语、俄语等)是人们用来表达思想和描述事物的工具,用自然语言表达的命题人们非常自然地就可以接受。但是计算机是不可能接受自然语言的,它只能接受符号语言或称为形式语言(formal language)。因此我们所面对的是两种语言,即自然语言和形式语言。自然语言是人们使用并表达思想的语言,形式语言是为了某种目的人为构造的语言。自然语言也称为元语言(metalanguage),形式语言也称为对象语言(object language)。我们要用自然语言讲解形式语言,如同用汉语讲解英语一样。

在数理逻辑中,我们关注的是用元语言和形式语言所描述的命题。元语言所表达的命题能否在形式语言中得以准确完整的描述是下面要讨论的内容。从以后的讨论中我们将看到元语言表达的命题基本上都可以用数理逻辑形式语言中所定义的公式进行表示。不过目前我们必须明确的是元语言和形式语言位于不同的层次,元语言是用来讲解形式语言的语言,是自然语言;形式语言是被讨论的语言,是对象语言。而数理逻辑所关注的是这两种语言表达的命题,特别是用形式语言所定义的公式来表示元语言中的命题以及命题之间的关系。

2. 命题与命题的真值

通常人们用元语言表达命题,一般数学中的命题也是用元语言叙述的。命题演算将命题作为讨论的对象,因此首先应搞清楚什么是命题。如果从元语言的角度考虑,命题可以如下描述:能够辨别真假的陈述句称为命题。

这种看待命题的观点是经典逻辑的观点,经典逻辑认为一个命题不是真的就是假的,在真与假之间不存在其他情况。但是一些非经典逻辑系统认为在真与假之间还存在着其他逻辑值。不过本章讨论的是经典逻辑,因此非真即假的陈述句才是命题,并把命题取真或取假的情况称为命题的真值(truth value),用 1(或 T)表示真,用 0(或 F)表示假。现考虑下面的例子:

(a) 煤炭是黑颜色的。

(b) 地球是方的。

(c) 上海不是国家的首都。

(d) 北京市的中学生不仅学习语文并且还学习英语。

(e) 张三学过法语或者张三学过日语。

(f) 如果气温低于 0℃,那么水是液体。

(g) 水沸腾当且仅当水温达到 100℃。

上述这些用元语言叙述的语句都是陈述句,并且真假一目了然,显然(a)、(c)、(d)和(g)都是真的,它们的真值都是 1;(b)和(f)都是假的,它们的真值都是 0;至于(e)的真与假很容易判定,因此(a)～(g)都是命题。再考虑如下几个例子:

(h) 今天下雨吗?

(i) 祖国啊,母亲!

(j) 赶快过马路。

(k) 别的星球上存在着生命。

(l) 张三是个高个子同学。

这几个例子都不是命题。首先(h)、(i)和(j)都不是陈述句,所以不是命题;其次尽管(k)和(l)都是陈述句,但它们的真值到底是 1 还是 0 难以确定。就(k)来说,别的星球上是否存在生命目前是不知道的,真与假不能确定。对于(l)来说,在确定高个子的标准之前,个子的高与低并不明确,所以(l)所表述陈述句的真值就不能确定。因此从经典逻辑的观点考虑,(k)和(l)都不是命题。但是,某些非经典的多值逻辑把(k)看作命题,这时它的真值取 u,表示不确定。不过这不是本书所讨论的问题,具体内容可以参阅文献。

3. 联结词、简单命题和复合命题

分析上述例子(a)～(g)可以看出,(a)和(b)中的命题(即能够辨别真假的陈述句)都不能再进一步分解,否则它们就不是完整的陈述句,自然就构不成命题;而(c)～(g)中的命题都可以进一步分解为更简单的命题,比如,“上海不是国家的首都”由“上海是国家的首都”加否定“不(或非)”所构成。“如果气温低于 0℃,那么水是液体”由“气温低于 0℃”和“水是液体”经过“如果,那么”联结而得到。“北京市的中学生不仅学习语文并且还学习英语”、“张三学过法语或者张三学过日语”和“水沸腾当且仅当水温达到 100℃”分别由更简单的命题通过“并且”、“或者”和“当且仅当”联结而生成。因此我们给出如下的概念:

① “非”、“并且”、“或者”、“如果,则”(或“蕴涵”)和“当且仅当”称为联结词(connective);

② 不出现联结词的命题称为简单命题(simple proposition);

③ 出现联结词的命题称为复合命题(compound proposition)。

联结词“非”习惯上有时叙述为“不”;联结词“并且”有时也说成“且”或“与”;联结词“如果,则”同“如果,那么”意思相同,并且“如果,则”这个联结词也可用“蕴涵”来代替;联结词“当且仅当”也经常叙述为“充分必要”或“等值于”。需要注意的是联结词“或者”表示可兼或,即由该联结词联结的两个命题的真值都可以是 1(真)。

如“张三学过法语或者张三学过日语”中的“张三学过法语”和“张三学过日语”都可以是真的。而在自然语言或元语言中,“或者”有时表示的是不可兼或,即由该联结词联结的两个命题若一个真,那么另一个必然假。如在命题“张三在家看电视或者张三去电影院看电影”中,“张三在家看电视”和“张三去电影院看电影”的真值不可能同时为真。因此需要注意我们这里联结词中的“或者”表示可兼或。另外还需强调的是上边的讨论是在元语言中进行的,不涉及形式语言的概念,因为到目前为止还没有讨论形式语言的任何内容。

4. 元语言的疑义问题

元语言表示的语句有时可使人产生误解。比如,某班的学生中,一个同学的名字叫“李丰”,另一个同学的名字叫“李丰收”。当该班接到了一封信,信皮上写着“李丰收”三个字时,大家可能弄不清楚此信到底是“李丰”收,还是“李丰收”收。这就是自然语言在某些情况下的不精确之处。此处的“李丰”或“李丰收”由于并非陈述句,所以它们不是命题。而元语言表达的命题有时也会让人们产生误解,请看如下例子:

如果张三非进城去,则我也去。

此处我们想表达并且已经表达了很清楚的思想,但是如果外人不了解这种思想的话,观察此命题后会感觉到一头雾水。到底是张三一定要(“非”解释为“一定”)进城去呢?还是张三不(“非”解释为“不”)进城去呢?实际上这两者都不是我们要表达的思想。如果把“张三非”看作一个人的名字,那么我们所要表达的意思便一目了然,这才是我们真正要表达的意思。

自然语言在某些情况下的不精确性对计算机来说接受是困难的,因为计算机只能在明确指令的指派下进行工作,消除这些不精确性对计算机推理是非常重要的。尽管逻辑学家创建数理逻辑的初衷并不是出于对计算机推理的考虑,而一个重要的目的是要消除元语言中的某些异议,但多年后的计算机科学将数理逻辑作为推理的理论基础则进一步表明了基础研究的重要性。数理逻辑中的一种重要方法就是要对元语言表达的命题符号化或形式化。下面构建命题演算的形式语言,并在形式语言中定义命题公式。命题公式属于形式语言的范围,它可以对用元语言所表示的许多命题进行准确的描述,并且这些形式化的命题——命题公式不会产生疑义。

1.2 形式语言及命题公式

1. 形式语言的定义

形式语言是为了某种目的所构造的字符串的集合。下面给出形式语言的定

义，为此先给出字母表和字符串的概念。

定义 1.2.1　字符(symbol)的集合称为字母表(alphabet)。

字符就是讨论问题时所涉及的对象，字母表就是由这些对象组成的集合，它可以是有限集也可以是无限集。例如，26 个英文字母可以构成字母表，10 个阿拉伯数字也可以构成字母表等。

定义 1.2.2　由字母表中的字符构成的有限长的序列称为字母表上的字符串(symbol string)。字符串中字符的个数称为字符串的长度。长度为零的字符串称为空串(empty string)，用 ε 表示，空串是没有任何字符的字符串，是一个特殊的字符串。若 A 是字母表，则用 A^* 表示字母表 A 上所有字符串的集合(包括空串)。

例 1　设字母表 $A=\{0,1\}$，则

$$A^* = \{\varepsilon,0,1,00,01,10,11,000,001,\cdots\}。$$

有了上述的准备，便可以给出形式语言的定义。

定义 1.2.3　设 A 是字母表，A^* 是字母表 A 上所有字符串的集合，A^* 的任意子集称为字母表 A 上的形式语言(formal language)。形式语言中的任意一个字符串称为该形式语言的一个语句(sentence)。

A 上的形式语言是 A^* 的子集，子集是多种多样的，如果有目的的选取 A 和 A^* 的子集，那么得到的形式语言就具有特殊的用处。

例 2　设字母表 $A=\{0,1,2,3,4,5,6,7,8,9\}$，$A^*$ 是由 10 个阿拉伯数字组成的所有十进制数的集合且包括空串 ε 和有限个“0”构成的字符串(如：00，000，0000 等)。下列①～④均为字母表 A 上的形式语言：

① $L_1=\{0,5,10,15,20,25,30,35,40,45,\cdots\}$ 表示可以被 5 整除的所有十进制数的集合；

② $L_2=\{0000,0001,0010,0011,0100,0101,0110,\cdots,1111\}$，该形式语言中的字符串是由“0”和“1”构成的所有长度为 4 的数字的集合，可以看成是长度为 4 的所有二进制数的集合；

③ $L_3=\{1,3,5,7,9,11,13,15,\cdots\}$，该形式语言表示所有奇数的集合；

④ $L_4=\{0,1,4,9,16,25,36,49,\cdots,n^2,\cdots\}$，该形式语言表示所有十进制数的平方的集合。

例 2 中几种形式语言中的字符串(实际上是数字)之间的排序都是有规律的，所以这些语言实际上都可以用一种算法来生成。利用算法有目的生成的形式语言当然有特定的用处，但本书并不打算进一步讨论形式语言的有关内容，这方面知识可参阅文献。给出形式语言定义的目的是为了说明命题演算中命题公式的全体是一种形式语言。

2. 命题公式

下面将要定义命题公式，命题公式的全体是一种形式语言，所以第一步需要给

出字母表。

前面已经表明命题是能够辨别真假的陈述句，但陈述句是用元语言表示的句子，为了使命题形式化，需要采用字符串表示命题，约定用如下的小写英文字母表示简单命题，称为命题符号(propositional symbol)：

$$p,q,r,p_1,p_2,p_3,\cdots,p_i,\cdots$$

此处既给出字符 p,q,r 又给出带下标的字符 $p_1,p_2,p_3,\cdots,p_i,\cdots$ 的目的是当以后定义的公式中出现表示简单命题的命题符号小于等于三个时一般用 p,q,r 表示，大于三个时一般用 $p_1,p_2,p_3,\cdots,p_i,\cdots$ 中的某些表示。由此可知命题符号的个数是可数的，即与自然数一样多。

与 1.1 节中五个连接词“非”、“并且”、“或者”、“如果，则”和“当且仅当”相对应，引入如下五个符号，称为联结符号(connective symbol)：

$$\neg,\quad \wedge,\quad \vee,\quad \rightarrow,\quad \leftrightarrow。$$

这五个联结符号的名称分别为：“$\neg$”称为否定符号(negative symbol)；“$\wedge$”称为合取符号(conjunctive symbol)；“$\vee$”称为析取符号(disjunctive symbol)；“$\rightarrow$”称为蕴涵符号(implicative symbol)；“$\leftrightarrow$”称为等值符号(equivalent symbol)。

为了技术上的需要，再引入两个符号：“(”和“)”，前者称为左括号(left bracket)，后者称为右括号(right bracket)。

于是如果用 T 表示命题演算的字母表，则有

$$T=\{\neg,\wedge,\vee,\rightarrow,\leftrightarrow,(,),p,q,r,p_1,p_2,p_3,\cdots,p_i,\cdots\}。$$

T^* 表示字母表 T 上所有字符串的集合(包括空串)。命题公式的全体是 T^* 的子集，定义如下。

定义 1.2.4 每一个命题符号称为命题演算中的原子公式(atomic formula)。原子公式全体组成的集合记作 $\mathrm{Atom}(L^p)$，即 $\mathrm{Atom}(L^p)=\{p,q,r,p_1,p_2,p_3,\cdots,p_i,\cdots\}$。

其中的 L^p 表示命题逻辑，以下都使用 L^p 表示命题逻辑或命题演算。从定义 1.2.4 看出，每一个原子公式都是一个命题符号，所以我们可以用命题符号表示自然语言或元语言中的简单命题，因此这里的重要之处就是可把元语言中的简单命题进行符号化。

从原子公式出发，通过联结符号，可以定义 L^p 中的命题公式。

定义 1.2.5 L^p 中的命题公式(propositional formula)是通过如下规则生成的字符串：

① $\mathrm{Atom}(L^p)$ 中的每一原子公式是命题公式；

② 若 A 是命题公式，则 $(\neg A)$ 是命题公式；

③ 若 A 和 B 是命题公式，则 $(A\wedge B),(A\vee B),(A\rightarrow B),(A\leftrightarrow B)$ 都是命题公式；

④ 只有有限次使用①、②或③得到的字符串是命题公式。

如果将 L^p 中命题公式的全体记作 Form(L^p)，则显然 Form(L^p)是 T^* 的子集，因此由定义 1.2.3 知 Form(L^p)是一种形式语言。任意一个命题公式都是 Form(L^p)中的一个语句。

注意：定义 1.2.5 称为归纳定义(inductive definition)，归纳定义用来定义集合，即给出一个由某些对象构成的集合。此处我们并不严格讨论归纳定义本身的定义，只说明归纳定义的特点，这就是：首先确定最基本的对象，该步称为基始；然后利用已有的对象，通过若干法则生成新的对象，该步称为归纳；最后对整个过程进行说明，该步称为总结。定义 1.2.5 中的①是基始，②和③是归纳，④是总结。归纳定义的方法在以后的章节中还会使用。

由定义 1.2.5 可知，除原子公式外，命题公式的最外层都是有括号的(由该定义中的②和③所决定)，书写时为了简单往往省略公式的最外层括号。如果再约定联结符号 $\neg$、$\wedge$、$\vee$、$\rightarrow$、$\leftrightarrow$的优先次序为从左到右，即 $\neg$ 的优先级第一，$\wedge$ 的优先级第二，$\vee$ 的优先级第三，$\rightarrow$的优先级第四，$\leftrightarrow$的优先级最弱，那么命题公式还可以进一步简化。例如，$(((\neg p)\wedge q)\rightarrow q)\vee r$ 按照约定可以简化为 $(\neg p\wedge q\rightarrow q)\vee r$，但不能简化为 $\neg p\wedge q\rightarrow q\vee r$，因为按照联结符号的优先级该公式应还原为 $((\neg p)\wedge q)\rightarrow(q\vee r)$，这显然有别于命题公式 $(((\neg p)\wedge q)\rightarrow q)\vee r$。因此，不能省略的括号一定要保留，有时为了使命题公式的意义明确，仍保留某些可以省略的括号，甚至可以用其他的括号，如“{，}”及“[，]”。当严格讨论命题公式的结构时，不应省略任何括号。以后我们用 A、B、C 等表示命题公式。

上述对形式语言和命题公式这两个问题进行了讨论，实际上命题公式的全体就是命题逻辑字母表上的形式语言。至此我们已经有了两种语言：元语言和形式语言。元语言是我们用于表达思想的语言，命题是能够辨别真假的陈述句，是用元语言描述的；形式语言是为了某种目的所构建的语言，而构建命题公式这种形式语言的目的是为了对元语言表示的许多命题进行符号化或形式化的描述。以后若某些命题用命题公式进行了表示，则称这些命题可被符号化或形式化(formalization)。

1.3　命题公式的语义

命题是用元语言表述的陈述句，命题公式是用 L^p 字母表中字符组成的字符串，是形式语言中的语句。命题和命题公式之间的关系如何呢？我们的回答是元语言表述的许多命题都可以用命题公式进行表示。这里说许多命题而没有说所有命题都可以用命题公式表示是因为一些用元语言表示的命题用命题公式表示时，并不能完整描述命题的性质或命题所涉及对象之间的关系，需要引入谓词的概念

才能描述清楚，这是第 2 章的内容。要用命题公式表示命题，需对命题公式的语义(semantics)进行定义。什么是命题公式的语义呢？命题公式的语义是与命题的真值相对应的概念，也就是命题公式被指定为 1(对应命题的真)或 0(对应命题命题的假)的情况，因此命题公式的语义就是该命题公式被指派为 1 或 0 的取值情况。命题公式是 L^p 字母表中字符组成的字符串，本身无任何意义。所以要想使命题公式有语义(即可取 1 或 0)，必须进行相应的定义。需要说明的是：对于元语言所表述的命题取真或取假的情况我们称为真值，对于形式语言所表述的命题公式取 1 或取 0 的情况我们称为语义。前者是元语言中的表述，后者是形式语言中的表述，但习惯上两者均用“真值”进行表述，只要分清是在元语言中对命题的讨论还是在形式语言中对命题公式的讨论即可。

1. 联结符号的含义

首先再进一步指出形式语言中的五个联结符号“$\neg$”、“$\wedge$”、“$\vee$”、“$\rightarrow$”、“$\leftrightarrow$”与自然语言中的五个联结词“非”、“并且”、“或者”、“蕴涵(如果，则)”、“当且仅当”之间的关系，我们希望有如下的对应：

“$\neg$”表示：“非”；

“$\wedge$”表示：“并且”；

“$\vee$”表示：“或者(可兼或)”；

“$\rightarrow$”表示：“蕴涵”(或“如果，则”)；

“$\leftrightarrow$”表示：“当且仅当”(或“等值”)。

此处，我们希望联结符号与联结词之间有上述这样的对应表示，要想实现这种表示，需对含有相应联结符号命题公式的语义进行严格的定义。

2. 赋值

定义 1.3.1(对原子公式的赋值) 任意一个从 $\mathrm{Atom}(L^p)$ 到 $\{0,1\}$ 的映射或函数 $v:\mathrm{Atom}(L^p)\rightarrow\{0,1\}$ 称为命题演算的一个赋值(valuation)。并且对 $p\in\mathrm{Atom}(L^p)$，将 $v(p)$ 记作 p^v，自然 $p^v\in\{0,1\}$。

由于 $\mathrm{Atom}(L^p)$ 中有可数多个原子命题，因此从 $\mathrm{Atom}(L^p)$ 到 $\{0,1\}$ 的映射的个数与实数的个数一样多(注：如果对基数的概念没有较多的了解，不必对此问题过多的考虑)。所以针对命题演算来说，赋值非常之多。当 $p^v=1$ 时，则称在赋值 v 下 p 的真值(即语义)为真，当 $p^v=0$ 时，则称在赋值 v 下 p 的真值(即语义)为假。

赋值对所有原子公式指派了真值(也就是语义)，即对于任一赋值 v 和任一原子公式 p，有 $p^v=0$ 或 $p^v=1$。利用赋值 v，可以对任意的命题公式 $A\in\mathrm{Form}(L^p)$ 指派真值(即语义)，如果将 v 对 A 真值的指派记作 A^v，则有如下定义。

定义 1.3.2(对命题公式的赋值)　设 v 是命题演算的任意一个赋值，$A\in \mathrm{Form}(L^p)$是任意一个命题公式，v 对 A 真值的指派 A^v 递归定义如下：

① 如果 A 是原子公式 p，则 $A^v=p^v$ 且 $p^v\in\{0,1\}$；

② 如果 $A=\neg B$ 且 $B^v\in\{0,1\}$，则当 $B^v=1$ 时，规定 $A^v=(\neg B)^v=0$；当 $B^v=0$ 时，规定 $A^v=(\neg B)^v=1$；

③ 如果 $A=B\wedge C$ 且 $B^v,C^v\in\{0,1\}$，则当 $B^v=1$ 且 $C^v=1$ 时，规定 $A^v=(B\wedge C)^v=1$；当 $B^v=0$ 或 $C^v=0$ 时，规定 $A^v=(B\wedge C)^v=0$；

④ 如果 $A=B\vee C$ 且 $B^v,C^v\in\{0,1\}$，则当 $B^v=0$ 且 $C^v=0$ 时，规定 $A^v=(B\vee C)^v=0$；当 $B^v=1$ 或 $C^v=1$ 时，规定 $A^v=(B\vee C)^v=1$；

⑤ 如果 $A=B\rightarrow C$ 且 $B^v,C^v\in\{0,1\}$，则当 $B^v=1$ 且 $C^v=0$ 时，规定 $A^v=(B\rightarrow C)^v=0$；当 $B^v=0$ 或 $C^v=1$ 时，规定 $A^v=(B\rightarrow C)^v=1$；

⑥ 如果 $A=B\leftrightarrow C$ 且 $B^v,C^v\in\{0,1\}$，则当 $B^v=C^v$ 时，规定 $A^v=(B\leftrightarrow C)^v=1$；当 $B^v\neq C^v$ 时，规定 $A^v=(B\leftrightarrow C)^v=0$。

注意：定义 1.3.2 中出现了递归定义(recursive definition)的概念。此处我们不讨论递归定义本身的定义，只说明递归定义所完成的工作：递归定义用来定义函数或定义一些概念。递归定义可以通过已知的函数定义新的函数，或由最基本的概念出发定义与最基本概念有紧密联系的扩充意义下的概念。递归定义有别于用来定义集合的归纳定义，递归定义用来定义函数或概念，归纳定义用来定义集合。

在定义 1.3.2 中，我们利用赋值 v(从 $\mathrm{Atom}(L^p)$到$\{0,1\}$的映射)定义了一个从 $\mathrm{Form}(L^p)$到$\{0,1\}$的新映射 v(最好把这个 v 记作 v'，因为通过递归定义得到的从 $\mathrm{Form}(L^p)$到$\{0,1\}$的新映射 v'不同于从 $\mathrm{Atom}(L^p)$到$\{0,1\}$赋值 v。v'实际上是 v 的扩充，是由 v 确定的。为了简单我们仍把 v'记作 v，这样不会造成任何混乱)。

由该定义，对任意的 $A\in\mathrm{Form}(L^p)$，有 $A^v\in\{0,1\}$。以后把扩充的 v(即 v')：$\mathrm{Form}(L^p)\rightarrow\{0,1\}$仍称为赋值。实际上，有一个从 $\mathrm{Atom}(L^p)$到$\{0,1\}$的赋值，便有一个从 $\mathrm{Form}(L^p)$到$\{0,1\}$的赋值。由于赋值为每一个命题公式指派了真值，这样作为字符串的命题公式就有了语义，即命题公式可以被指派为 1(真)或 0(假)。

定义 1.3.2 中的②～⑥是对五个联结符号所构成的命题公式进行的真值定义，实际上也是对这五个联结符号含义的定义。如下的表格称为真值表(truth table)，通过真值表可进一步说明联结符号的含义。

当 $A=\neg B$ 时，对应真值表 1.3.1。由表 1.3.1 可知：当 B 的真值被指派为 1(即 $B^v=1$)时，$\neg B$ 的真值被定义为 0(即$(\neg B)^v=0$)；当 B 的真值被指派为 0 时，$\neg B$ 的真值被定义为 1。这种情况与定义1.3.2中的②相对应，是对联结符号“$\neg$”含

表 1.3.1

B	$A(=\neg B)$
1	0
0	1