思考题

考试准备

- 学习数理逻辑这门课的收获
  - 辩证思维：批判
  - 读书方法：华罗庚读书法
  - 应用实践与理论关系、区别
  - 哥德尔不完备性定理中证明的构造，对科研的启发
  - 形式系统的严谨、形式化思想、对其面临的挑战的思考

第零章

0.1什么是证明

助教：形式化证明，在数理逻辑中，形式化证明并不是以自然语言书写，而是以形式化的语言书写：这种语言包含了由一个给定的字母表中的字符所构成的字符串。而证明则是一种由该些字符串组成的有限长度的序列。这种定义使得人们可以谈论严格意义上的“证明”，而不涉及任何逻辑上的模糊之处。研究证明的形式化和公理化的理论称为证明论。尽管理论上来说，每个非形式化的证明都可以转化为形式化证明，但实际中很少会这样做。对形式化证明的研究主要应用在探讨关于可证明性的一般性质，或说明某些命题的不可证明性等等，

- 证明是通过一些明确的规则（例如形式证明里的规则和归纳证明的三段式）一步步得出结论的。
- 证明是指在一个命题演算体系下，从公理，推理规则和前提集，进行推理，从而证明某个命题的正确性。
- 什么是“证明”？我认为可以有下面三个维度的解释：
  从最广泛的意义来说，“证明”是为了说明一个命题的可靠性，这个过程有许多途径（实验、推理）。
  在数学领域，为了达到理想中的100%可靠（也就是“真”），“证明”通常采用公理化方法：一组规定好的公理+普遍接受的推理过程。问题是，选择的公理本身可能蕴含了矛盾（第三次数学危机体现的尤为明显），而且推理中往往会使用自然语言，这其中有多少直觉的成分（从而影响结论的可靠性）是说不清楚的。
  形式公理化方法就是为了解决这个问题，在这里（比如说我们的谓词演算系统K_N），“证明”被严格定义为符合某个确定规则的公式序列（形式证明），因此能够排除自然语言的含糊成分，剩下的目标就是得到K_N本身没有矛盾（使用K_N的合理性），更进一步我们希望对于“真”的命题（N有效的公式）都可以给出K_N的形式证明（K_N的能力足够强）。
- 数学证明就是严密的逻辑推理过程。逻辑推理过程是要以已经证明是真命题的命题作依据的。逻辑推理就是一个逻辑思维的过程，而思维则是人的脑子所特有的功能

0.2什么是计算

助教：计算理论的“计算”并非指纯粹的算术运算 (Calculation)，而是指从已知的输入透过算法来获取一个问题的答案 (Computation)

- “车赤-图灵论题（一个函数是可计算的，当且仅当该函数是图灵机可计算的）被大量等价性结果的证明和不成功计算模型的识别所确认——**凡公认合理的计算模型都被证明与图灵机等价，凡不与图灵机等价的计算模型都被公认为不合理/不充分。**”计算模型均与图灵机等价意味着我们可以把直观上的“计算”向图灵机的计算靠拢，考虑**能行可计算的递归函数**即可。
- 计算是指在给定符号，元素，计算规则的条件下，对一个“式子”的结果进行**化简（或化繁）**，从而得到我们期望的结果。
- 什么是“计算”？我认为也有两个层次的解释：
  朴素的观点来看，“计算”是我们（人）通过某些特定的方法（算法）给出一类问题在具体情形下的结果，这个算法应该是有限的简单步骤的叠加（无穷的步骤将导致实践中无法停止，没有现实意义）。比如说我们小学时就学过自然数的加法，有了加法的算法（比如个位数加法表+列竖式法），我们对任意两个自然数的和都能给出结果（只是时间上的问题），给出结果的过程就是“计算”，而加法就是能行可计算的。
  但这种定义终究不是严格的，到底怎样才算“简单步骤”？在数论函数领域，Church论题给出了一种合理的方案：用递归函数描述能行可计算，从而“计算”就是三种基本函数和三类规则的应用。
  Turing论题用更加机械的方法（图灵机）描述能行可计算，进一步说明了人与机器在计算（数论函数）的能力上是没有差异的。（由于图灵机可计算函数和递归函数的等价性，这两个论题也是等价的）
  不过我们日常中说的“计算”只包含数论函数的计算吗？我觉得可能不是。不过在计算机领域，由于存储元件表示的状态总是有限的，计算机的状态总可以映射为自然数片段，从而计算机的工作总可以映射为数论函数的计算，在这点上“计算”是和图灵机/递归函数的计算一致的。

- 数学计算可以是手工计算，也可以用计算工具进行计算。计数器，算盘，计算尺，计算机等，都是计算工具，他们都是人们为了提高计算速度而创造的，也只能在人的操作下才能完成计算工作。用计算工具进行计算，只要数据输入正确，口诀与程序正确，计算的结果一定是正确的。虽然人用手工也可以得到与用工具计算相同的结果，但工作量有可能太大，需要的时间很长。这就是用计算工具的好处。这里的关键是人会计算，人可以给出计算公式，并有加减乘除等计算的口诀，用手按口诀去拔动算珠；人也可以把计算的过程编成程序，输入计算机，让计算机自已去完成。这里人仍是关键，人要把计算的初始数据，计算方法，如何计算，计算的步骤等都要"教"给计算工具，让它们去按照原样完成。要让计算工具完成计算任务，首先人就要会计算，至少要知道怎么去计算，计算的方法是什么，也就是说，人首先要会去做这件计算工作，会做这件事

0.3计算与证明是什么关系

- 计算和证明的关系：上述"式子"间通过逻辑关系（相等，大于小于等）可以连接成一个"命题"。因此**"计算"能够为命题提供理论依据，而"证明"是经过计算得到的结果**
- 计算与证明的关系：
    1. 计算和证明都具有一定程度的"正确性"，计算和证明都是在一定规则下得到正确的结论：计算得到"结算结果"是正确的，证明得到"证明的命题"是正确的
    2. 因为计算具有正确性，所以计算可以作为证明的步骤，计算可以看作一种证明：如果计算所用的方法是正确的，那么计算的结果就是可证明的
    3. 证明的目的是验证结论的正确性，待验证的这个结论是在证明之前就给出的，但是计算不会先给定计算结果
    4. 车赤图灵论题认为可计算当且仅当图灵机可计算。
    5. 因为计算具有正确性，我们可以通过计算来自动证明。一类可判定的命题的证明可以通过来计算完成，因为存在一个能行方法，而能行方法是图灵机的计算模型。
    6. 并不是所有可被证明的命题的证明过程的推到都是可计算的。
- "计算"与"证明"是什么关系？
  通过Godel定理，我们知道有些"真"的公式在K_N中是不可证的，这说明K_N的表达能力的确有限。如果降低一下要求，问：
  "证明"（K_N中形式证明）是否都能通过"计算"（能行过程/图灵机）来实现？至少这是Leibniz的期望。对于已知K_N可证的公式，我们可以通过计算得到它的形式证明。可惜的是，K_N的不可判定性定理说明了我们不能有效计算出哪些公式是K_N可证而哪些不是。也就是说，如果要去设计K_N推理机，我们永远不能对哪些K_N不可证的公式产生有效输出。因此形式证明仅能部分地转化成能行计算（这点从K的半可判定性也可得）。
- 关系
- 计算是脱离了具体问题的纯数字的计算，无论是手工计算还是用计算工具计算，实际上还都是人在进行计算，只要细心都是会得到正确答案的。如果不细心，就是用了计算工具也是不能得出正确答案的。而数学证明则是不能离开具体问题的
- 所以说数学计算是纯数字的计算，而数学证明是具体问题（命题）的逻辑推理。有时在数学证明中可能还要进行一些数学计算，但数学计算中绝对再不需要再进行数学证明或逻辑推理了，因为数学计算是依据经过了数学证明（或逻辑推理）后是正确的、现成的计算公式进行的。数学计算则是数学证明的具体应用。
- 有纯数值命题的证明是用数学计算的，是可以用计算工具的，而实际上数值命题的证明就是数值计算的问题。而其余两种命题的证明是不能使用计算工具的。

0.4 例5中的天文观察结果是否证明了经典力学是假的，狭义相对论是真的?

- 只能证明经典力学是假的（与演绎定理的保真性相矛盾），但不能证明狭义相对论是真的，因为这里的p只是一个特例，不具有普遍性

  但是个人认为这里所说的"真假"应该放在不同的应用领域来理解，经典力学在一定的前提下是真的（低速宏观领域），但是在高速微观领域可能就不再适用或者需要一定的修正

- 可以证明经典力学为假，因为演绎推理具有保真性，这里的假指的是，经典力学的某些假设在现实世界中并不是恒真的；不能证明狭义相对论为真，因为可能存在某个命题 q 使得 \Gamma_爱 |- - q，而实验结果 q 为真。

0.5 如果在例5的基础上，还存在另一个命题q，使得都成立，而科学实验的结果为：q是真的。这说明什么?

- 结合0.4，这说明经典力学和狭义相对论都是假的。但事实上同0.4，这里的真假与应用领域有关。且存在一种可能：两个前提集可能都能证明出一个公式q，这时两个前提集之间有一部分是等价的，也就是说它们的某部分是相互可证的

0.6 能否证明经典力学和狭义相对论的真假？其中，所谓的“证明”和“真假”是什么意思?

- 证明：以已经证明是真命题的命题作依据的严密的逻辑推理。具体来说，这里是以演绎定理的保真性作为前提，通过某些实际观察到的命题的真假来和推理得到命题作比较，来倒退前提集的真假

  真假：在某一领域是否符合实际情况，符合即为真，不符合即为假

  个人认为不能证明二者的真假。因为两个理论有各自的适用范围，在不同的命题演算系统中二者可能会有不同的真假。如经典力学在宏观低速的范围内为真，到了高速微观领域可能就是假的了。狭义相对论同理

- 如果能存在 0.4 和 0.5 所述的条件，则可证明为假；但是我们永远无法证明为真，因为物理的理论都是基于一些前提和假设的，而我们验证真假，是基于我们对这个世界的观测，由于观测是无穷无尽的，因此我们永远无法使用暴力枚举的方法来验证我们的假设。这里的“证明”是指，如果存在 p 不符合推论，则可证明为假，如果任何 p 都符合推论，则可证明为真。这里的“真假”指是否在任何一次对世界的观测中符合我们的推论

0.7 暴力法和训练法有没有“真假”？应该根据什么来评价它们？如何比较它们的优劣?

- 暴力法有真假，训练法没有

  暴力法是具有保真性的，通过大量的推理可以判定一个问题的真假。但训练法只是用一定的测试集来训练，类似数学中的拟合，不具有保真性。用训练好的神经网络来进行预测只能说大概率为真，但不保证为真。

  评价标准：算法的效率、准确性、算法的复杂性等等

  两种方法各有优劣。暴力法需要大量的知识（前提集）作为保证，且时间开销大，但保证了结果的正确性。训练法用具有人工标注的数据来训练神经网络，时间开销相对较小，但是训练集的获取难度较大，一般需要用现成的训练集而无法快速得到自己想要的、具有某种特点的训练集，且不保证结果为真

- 在搜索空间有限的前提下，暴力法的结果为真，但是搜索空间无限时，暴力法可能无法在有效时间内给出结论，因此讨论真假没有意义。训练法依赖于训练集的正确性，但即使训练集正确，训练法的结果也不一定为真。评价他们可以用正确率和时空效率。

0.8 什么是常识？常识有什么应用？机器能具备并应用常识吗?

- 常识：在某一领域公认的、符合实际情况的、正确的知识

  应用：可以通过第三章的形式化理论，将某一领域的常识形式化为一系列公式作为前提集，通过推理机在语义上进行推导，进而得到更多的“真”的领域知识

  上述应用说明机器可以具备并运用常识

- 常识指的是根据经验或观测，人们认为为真的某些结论，即使部分结论实际上不始终为真。常识可以帮助人们推断出一些结论，当所使用的常识为真且推断过程正确时，推出的结论也为真。在预先设置的情况下，机器可以拥有常识，并通过某种算法运用他们。

第一章

1.1试用复合命题表达自然语言条件句“如果...则...”。

p->q

1.2同一律的证明是否必须使用(L1)？证明你的结论。

助教：使用赋值函数?

是。证明如下，假设 S 是由 L2 、L3 与 MP 规则但不使用 L1 可得的公式，在 $\mathcal{Z}_3 = \{0, 1, 2\}$ 上构造如下的赋值函数

$f_{\to}$	0	1	2
0	2	2	2
1	0	0	2
2	0	0	2

	0	1	2
$f_{\neg}$	2	1	0

并有

$$v(p \to q) = f_{\to}(v(p), v(q))$$
$$v(\neg p) = f_{\neg}(v(p))$$

这样的赋值可以保证由 L2 或 L3 得到的公式，赋值一定为 2：

(	p	$\to$	(	q	$\to$	r	))	$\to$	((	p	$\to$	q	)	$\to$	(	q	$\to$	r	))
	0	2		0	2	0		2		0	2	0		2		0	2	0	
	0	2		0	2	1		2		0	2	0		2		0	2	1	
	0	2		0	2	2		2		0	2	0		2		0	2	2	
	0	2		1	0	0		2		0	2	1		2		0	2	0	
	0	2		1	0	1		2		0	2	1		2		0	2	1	
	0	2		1	2	2		2		0	2	1		2		0	2	2	
	0	2		2	0	0		2		0	2	2		2		0	2	0	
	0	2		2	0	1		2		0	2	2		2		0	2	1	
	0	2		2	2	2		2		0	2	2		2		0	2	2	
	1	2		0	2	0		2		1	0	0		2		1	0	0	
	1	2		0	2	1		2		1	0	0		2		1	0	1	
	1	2		0	2	2		2		1	0	0		2		1	2	2	
	1	0		1	0	0		2		1	0	1		2		1	0	0	
	1	0		1	0	1		2		1	0	1		2		1	0	1	
	1	2		1	2	2		2		1	0	1		2		1	2	2	
	1	0		2	0	0		2		1	2	2		0		1	0	0	
	1	0		2	0	1		2		1	2	2		0		1	0	1	
	1	2		2	2	2		2		1	2	2		2		1	2	2	
	2	2		0	2	0		2		2	0	0		2		2	0	0	
	2	2		0	2	1		2		2	0	0		2		2	0	1	
	2	2		0	2	2		2		2	0	0		2		2	2	2	
	2	0		1	0	0		2		2	0	1		2		2	0	0	
	2	0		1	0	1		2		2	0	1		2		2	0	1	
	2	2		1	2	2		2		2	0	1		2		2	2	2	
	2	0		2	0	0		2		2	2	2		0		2	0	0	
	2	0		2	0	1		2		2	2	2		0		2	0	1	
	2	2		2	2	2		2		2	2	2		2		2	2	2	

(	¬	p	→	¬	q	)	→	(	q	→	p	)
	2	0	2	2	0		2		0	2	0	
	2	0	2	2	0		2		0	2	0	
	2	0	2	2	0		2		0	2	0	
	2	0	0	1	1		2		1	0	0	
	2	0	0	1	1		2		1	0	0	
	2	0	0	1	1		2		1	0	0	
	2	0	0	0	2		2		2	0	0	
	2	0	0	0	2		2		2	0	0	
	2	0	0	0	2		2		2	0	0	
	1	1	2	2	0		2		0	2	1	
	1	1	2	2	0		2		0	2	1	
	1	1	2	2	0		2		0	2	1	
	1	1	0	1	1		2		1	0	1	
	1	1	0	1	1		2		1	0	1	
	1	1	0	1	1		2		1	0	1	
	1	1	0	0	2		2		2	0	1	
	1	1	0	0	2		2		2	0	1	
	1	1	0	0	2		2		2	0	1	
	0	2	2	2	0		2		0	2	2	
	0	2	2	2	0		2		0	2	2	
	0	2	2	2	0		2		0	2	2	
	0	2	2	1	1		2		1	2	2	
	0	2	2	1	1		2		1	2	2	
	0	2	2	1	1		2		1	2	2	
	0	2	2	0	2		2		2	2	2	
	0	2	2	0	2		2		2	2	2	
	0	2	2	0	2		2		2	2	2	

同时由 $f_{\to}$ 的真值表可知，当 $v(p)=2$ 且 $v(p\to q)=2$ 时，一定有 $v(q)=2$，因此由 MP 规则得到的公式也一定有赋值为 2。但 L1 并不能保证：

p	$\to$	(	q	$\to$	p	)
0	2		0	2	0	
0	2		0	2	0	
0	2		0	2	0	
0	2		1	0	0	
0	2		1	0	0	
0	2		1	0	0	
0	2		2	0	0	
0	2		2	0	0	
0	2		2	0	0	
1	2		0	2	1	
1	2		0	2	1	
1	2		0	2	1	
1	0		1	0	1	
1	0		1	0	1	
1	0		1	0	1	
1	0		2	0	1	
1	0		2	0	1	
1	0		2	0	1	
2	2		0	2	2	
2	2		0	2	2	
2	2		0	2	2	
2	2		1	2	2	
2	2		1	2	2	
2	2		1	2	2	
2	2		2	2	2	
2	2		2	2	2	
2	2		2	2	2	

因此可以集合 S 的特征函数当且仅当 $v(p) \equiv 2$ 时取值 1

而同一律的赋值 $v(p \to p)$ 由 $f_{\to}$ 的真值表可知并不一定为 2，因此不可由 L2，L3 和 MP 规则推之

- 是。首先明确同一律中无否定词出现，所以在直接证明中也用不到L3。单纯地用L2和MP规则，那么需要L2的后件中出现(p->q)->(p->p)的形式，且需要证明出前件p->(q->p)为真，这显然是不能单纯地通过L2和MP证明出来的，需要L1作为前提。
- 是的。若不用L1。因为同一律里无否定词，所以也不会用到L3。而由L2的公理模式 (p->(q->r))->((p->q)->(p->r)), 要得到 p->p的形式，必须在L2的后件里出现这个形式，再用MP来推出。而L2后件里若有p->p形式，则必然是 (p->q)->(p->p) 或者 (p->p)->(p->r)(在蕴含词前面，舍去),一来p->q的真是无法证明的，另一方面对应前件应有 p->(q->p)，这就回到了L1。

1.3你以前理解的严格证明与命题演算L中的形式证明有何异同？L中的形式证明/推理有什么必要性?

- 同：都具有一定的逻辑规则，通过一系列已知为真的命题来证明结论的正确性

  异:

  1.严格证明的方法更为丰富，如反证法、归纳法等等，而L中的形式证明只能通过3条公理模式和MP规则来进行推理。

2.严格证明中使用了一些数学中的符号或者一些自然语言的表述，但这些并没有在L中定义，L相对比较单一

3.严格证明中的前提来源更为广泛，可以为一些显然为真的结论或者通过各种方法得到的真命题，但L的形式证明只能使用3条公理或者已经通过形式证明证出来的公式作为前提

4.严格证明中可以使用无限的定义，如数学中的归纳法，可能存在n->无穷的情况，但形式证明必须是有限的公式序列

必要性：

1.形式推理/证明提供了一种形式化的证明方法，有利于计算机、推理机等进行机器推理

2.形式推理/证明适用于各种领域，可以将其领域知识形式化为具体的公式作为前提集，再借助推理机等推理工具来进行推理，这提供了一种解决不同领域问题的一种普适性方法

- 同: 均是用已知公理来推导出所证结果

  异：严格证明:可能有前提条件，更类似于形式推理。但是其推理规则是在实质公理系统内的 常识，经验，直观推理等。

  形式证明:无前提，只有三条公理模式以及 MP规则来推理，是形式公理系统上的推理。

  必要性: 是形式公理系统中证明公式的方法体系，以形式化的语言，通过有有限长度的证明序列做出推理，在计算机等领域具有严密性的意义，而且其有限长度也使得推理机推理等成为可能，从而使得形式证明成为一种在不同领域解决问题的普适性方法。 此外，严格证明中有一些人类直觉或观察到的事实作为前提，而这些所谓的事实在一定范围下是矛盾的，这样证明所得的结果相应的在该范围内也不正确。但是形式证明则基于重言式，在所有领域为真，解决了证明中国可能存在的矛盾。

1.4演绎定理说明了什么?

助教：每个有效的蕴涵语句都描述了一个正确的推理

- 若仍认可可靠性与完全性的成立，则演绎定理强迫->解释为实质蕴涵：因为|-p->q 成立<=> {p}|-q <=>{p}|=q，即要求p真的情况下q为真，p假的情况下对q 不做要求
- 这也说明了演绎定理体现了蕴含词->和形式推理Γ|-p之间的某种一致性；推出是公式序列间的推导关系，而蕴含是命题与命题间的真值关系（or公式中各个子公式的关系or公式的逻辑结构）

1.5直接证明 $\vdash (\neg p \to p) \to p$ **最少需要多少步?**

19步

最少需要 19 步，证明如下

证明.

(1) $(\neg\neg(p \to (\neg p \to p)) \to \neg p) \to (p \to \neg(p \to (\neg p \to p)))$ (L3)

(2) $((\neg\neg(p \to (\neg p \to p)) \to \neg p) \to (p \to \neg(p \to (\neg p \to p))))$
$\to (\neg p \to \Big((\neg\neg(p \to (\neg p \to p)) \to \neg p)$
$\to (p \to \neg(p \to (\neg p \to p)))\Big))$ (L1)

(3) $\neg p \to \Big((\neg\neg(p \to (\neg p \to p)) \to \neg p)$
$\to (p \to \neg(p \to (\neg p \to p)))\Big)$ (1), (2), MP

(4) $\Big(\neg p$
$\to ((\neg\neg(p \to (\neg p \to p)) \to \neg p) \to (p \to \neg(p \to (\neg p \to p))))\Big)$
$\to \Big((\neg p \to (\neg\neg(p \to (\neg p \to p)) \to \neg p))$
$\to (\neg p \to (p \to \neg(p \to (\neg p \to p))))\Big)$ (L2)

(5) $(\neg p \to (\neg\neg(p \to (\neg p \to p)) \to \neg p))$
$\to (\neg p \to (p \to \neg(p \to (\neg p \to p))))$ (3), (4), MP

(6) $\neg p \to (\neg\neg(p \to (\neg p \to p)) \to \neg p)$ (L1)

(7) $\neg p \to (p \to \neg(p \to (\neg p \to p)))$ (5), (6), MP

(8) $(\neg p \to (p \to \neg(p \to (\neg p \to p))))$
$\to ((\neg p \to p) \to (\neg p \to \neg(p \to (\neg p \to p))))$ (L2)

(9) $(\neg p \to p) \to (\neg p \to \neg(p \to (\neg p \to p)))$ (7), (8), MP

(10) $(\neg p \to \neg(p \to (\neg p \to p))) \to ((p \to (\neg p \to p)) \to p)$ (L3)

(11) $((\neg p \to \neg(p \to (\neg p \to p))) \to ((p \to (\neg p \to p)) \to p))$
$\to ((\neg p \to p)$

$$\to ((\neg p \to \neg(p \to (\neg p \to p))) \to ((p \to (\neg p \to p)) \to p))) \qquad \text{(L1)}$$

(12) $(\neg p \to p)$

$$\to \Big((\neg p \to \neg(p \to (\neg p \to p)))$$

$$\to ((p \to (\neg p \to p)) \to p)\Big) \qquad (10), (11), \text{MP}$$

(13) $\Big((\neg p \to p)$

$$\to ((\neg p \to \neg(p \to (\neg p \to p))) \to ((p \to (\neg p \to p)) \to p))\Big)$$

$$\to \Big(((\neg p \to p) \to (\neg p \to \neg(p \to (\neg p \to p))))$$

$$\to ((\neg p \to p) \to ((p \to (\neg p \to p)) \to p))\Big) \qquad \text{(L2)}$$

(14) $((\neg p \to p) \to (\neg p \to \neg(p \to (\neg p \to p))))$

$$\to ((\neg p \to p) \to ((p \to (\neg p \to p)) \to p)) \qquad (12), (13), \text{MP}$$

(15) $(\neg p \to p) \to ((p \to (\neg p \to p)) \to p)$ (9), (14), MP

(16) $((\neg p \to p) \to ((p \to (\neg p \to p)) \to p))$

$$\to (((\neg p \to p) \to (p \to (\neg p \to p))) \to ((\neg p \to p) \to p)) \qquad \text{(L2)}$$

(17) $((\neg p \to p) \to (p \to (\neg p \to p))) \to ((\neg p \to p) \to p)$ (15), (16), MP

(18) $(\neg p \to p) \to (p \to (\neg p \to p))$ (L1)

(19) $(\neg p \to p) \to p$ (17), (18), MP

□

1.6编程实现一个命题演算中形式推理的程序。

1.7语义后承与重言式有何关系？下述论断是否成立？任给L(X)公式p和公式集Γ，存在公式q，Γ|=p当且仅当|=q。

助教：重言式是空集的语义推论

若 Γ 是有限集，设 $\Gamma = \{p_1, p_2, \cdots, p_n\}$，令

$$q = p_1 \to (p_2 \to \underbrace{(\cdots(}_{n-2\text{个}(} p_n \to p \underbrace{)\cdots)}_{n-2\text{个})})$$

使用 n 次语义演绎定理即可得 $\Gamma \vDash p \Leftrightarrow \vDash q$。

若 Γ 是无限集，参考 1.9 及维基百科-递归可枚举集合

- 关系：重言式是语义后承中的一个特殊情况，即前提集为空集时的情况
- 构造 Γ={p1，p2，....，pn}, q=(p1 ^ p2 ^ .....^ pn)->p;
- 充分性: 对任意语义解释I, 当Γ中所有公式为真时，由语义后承知I(p)=t,此时在I下，q的前件后件均为真，所以I(q)=t;若Γ中有公式为假时，此时由实质蕴含知I(q)=t,所以q是重言式；
- 必要性: 对任意语义解释I, 若Γ中所有公式为真，则由I(q)=t以及I(p1 ^ p2 ^ .....^pn)=t知 I(p)=t。
- 当Γ无限时，通过紧致性可令Γ转化为有限集，则充分性可证；再利用语义后承的单调性，（先推出一个有限的γ'，再通过单调性得到γ）必要性亦可证。

1.8是否存在L公式p和公式集Γ，使得 Γ ⊢ p 并且 Γ ⊢ ¬p?

- 存在，比如公式集就取 p和非p。一般的，可以构造不相容公式集，由其平凡性可知其能推出一切公式，也就能推出p和非p。

1.9问题“Γ ⊢ p”是不是可判定的

助教

> 问题“$\Gamma \vdash p$”是半可判定的。
> 若 Γ 是有限集，设 $\Gamma = \{p_1, p_2, \cdots, p_n\}$，令
>
> $$q = p_1 \to (p_2 \to \underbrace{(\cdots(}_{n-2\text{个}(} p_n \to p \underbrace{)\cdots)}_{n-2\text{个})})$$
>
> 使用 n 次语义演绎定理即可得 $\Gamma \vDash p \Leftrightarrow \vDash q$。通过真值表可对 q 是否为重言式做判定，由命题逻辑的一致性，即可判定 $\Gamma \vdash p$。
> 若 Γ 是无限集，由紧致性定理，若 $\Gamma \vdash p$ 成立，则有有限子集 $\Delta \subset \Gamma$，使得 $\Delta \vdash p$，则由上可知可以判定 $\Gamma \vdash p$ 确实成立。但若 $\Gamma \vdash p$ 不成立，则无法给出有限的判断算法。

- 整体上是半可判定的

  Γ**为有限集时可判定**，利用可靠性和完全性定理，等价于Γ|=p，因为p一定是有限长的公式，故其含有的原子命题个数有限，设为n，可通过列真值表进行求解（最多有2^n个指派）

  Γ**为无限集时半可判定**，取决于Γ|-p是否成立。如果成立，则可利用紧致性原理找到对应的有限子集Γ'同上进行求解。如果不成立，则在有限时间内不一定能回答yes。

- 由可靠性和完全性定理，可转化为语义后承来证明。

  当Γ有限时，可通过真值表来证明，一定可以在有限时间内列出所有情况并给出是与否的回答，所以是可判定

  当Γ无限时， 则当Γ|-p不成立时，在有限时间里无法给出回答。 当Γ|-p成立时，可以通过紧致性来化为有限子集，再加上L中公式集的可数性，所以可以在有限时间里得到“是”的回答。此时Γ为半可判定。

第二章

2.1(K4)和(K5)中的约束条件有何意义？举例说明。

助教：

(K4) $\forall x\, p(x) \to p(t)$，其中项 t 对 $p(x)$ 中的 x 是自由的

(K5) $\forall x\, (p \to q) \to (p \to \forall x\, q)$，其中 x 不在 q 中自由出现

(K4)和(K5)中的约束条件有何意义？举例说明。

对公理的限制保证了其在谓词逻辑的任何解释域中都是有效式。

对(K4)而言，直观上这是一个弱化结论的过程，从一个公式对任意 x 均成立得到其对某个具体的项 t 成立，在这个过程中，由于公式 $p(x)$ 是抽象的，所以必须要考虑到其内部可能存在的约束条件。举例说明，对于解释域 $\mathbb{Z}$，$\mathbb{Z}$ 上一元关系 $\overline{R_1^2}$ 为 $>$，则使用没有限制条件的 K4 公理得到

$$\forall x\, \exists y\, R_1^2(x,y) \to \exists y\, R_1^2(y,y)$$

前件的解释为“对任意的 $x \in \mathbb{Z}$ 均存在 $y \in \mathbb{Z}$ 使得 $x > y$”，这是恒真的；后件的解释为“存在 $y \in \mathbb{Z}$ 使得 $y > y$”，这是恒假的，公式在解释域 $\mathbb{Z}$ 上并不恒真。对于(K5)而言，这是一个具体化约束条件的过程，将对 x 的量词约束范围从整个蕴涵式 $p \to q$ 具体到后件 q 上。那么约束条件存在的意义是保证前件实际上并不受量词约束，即保证没有变元 x 逸出了约束范围。举例说明，对于解释域 $\mathbb{Z}$，$\mathbb{Z}$ 上一元关系 $\overline{R_1^1}$ 为 >1，$\overline{R_2^1}$ 为 >0，则使用没有限制条件的 K5 公理得到

$$\forall x\, ((R_1^1(x) \to R_2^1(x))) \to ((R_1^1(x) \to \forall x\, R_2^1(x)))$$

前件的解释为“对任意的 $x \in \mathbb{Z}$，若 $x > 1$，则 $x > 0$”，这是恒真的；后件的解释为“若 $x > 1$，则对任意的 $x \in \mathbb{Z}$ 有 $x > 0$”，注意这里后件里的 x 既有自由出现也有约束出现，可以改写变元为“若 $x > 1$，则对任意的 $y \in \mathbb{Z}$ 有 $y > 0$”，这是恒假的，公式在解释域 $\mathbb{Z}$ 上并不恒真。

2.1

保证了 K4、K5 在任何 M 下恒成立

K4 是从一般到特殊的公理，保证了不会出现更多的约束条件

反例：M = {R , φ , 不等于}，则有∀ x∃yR$_1^2$(x , y)→∃yR$_1^2$(y , y)，显然不成立，由语法和语义的关系可知，前提恒为真但结论恒为假，整体为假，不是重言式，因此在语法上此公理不成立。即将 x 替换为 y 受到了新的约束

K5 是从特殊到一般的公理，保证了不会有约束条件被忽略掉

反例：M={R，φ，{R$_1^1$：小于 2，R$_2^1$：小于 1}}，则有∀ x（R$_2^1$(x)-> R$_1^1$(x)）->(R$_2^1$(x)-> ∀ x R$_1^1$(x))，显然不成立，由语法和语义的关系可知，前提恒为真但结论恒为假，整体为假，不是重言式，因此在语法上此公理不成立。即使用 K5 后缺少了约束条件

- 限制条件的意义在于保证K4，K5在任何解释M下均恒真
- K4：∀xp(x)→p(t)，项t对p(x)中的x是自由的

  K4是从一般到特殊的公理，保证了不再有出现更多的约束条件，若没有该限制条件，则该公式的从任意到特殊的推理是不合理的，比如如下反例:

  M = {R , φ , <} , 则有∀ x∃yP(x , y)→∃yP(y , y)（P为二元谓词，映射到 <）显然不成立，前提恒为真但结论恒为假，由实质蕴含可知整体为假，不是重言式，因此在语法上此公理不成立，此时x替换为y受到了∃y的约束

- K5：∀x (p->q) -> (p -> ∀xq)，x不在p中自由出现

  K5是从特殊到一般的公理，保证了不会有约束条件被忽略掉，若没有该限制条件，那么p将没有约束，而∀x (p->q)对p,q都有约束，推到p -> ∀xq时p就没有了约束，显然是不对的，比如以下反例：

M={R，φ，{R11：小于2，R21：小于1}}，则有∀ x（R21(x)-> R11 (x)）->(R21(x)-> ∀ x R11 (x)), 显然不成立，因为前提恒为真但结论恒为假，由实质蕴含可知整体为假，不是重言式，因此在语法上此公理不成立。缺少了对p的约束条件

2.2下列判断是否成立?

若Γ|=p, 则对一切解释I，如果对所有 q ∈Γ 有I(q)=t，则I(p)=t。

> 不成立，由语义推论的定义，公式 p 是公式集 Γ 的语义推论，记作 $\Gamma \vDash p$，指 p 在 Γ 的所有模型中都恒真。但在不是公式集 Γ 模型的解释域中，Γ 的公式的真值是不确定的，因此判断并不成立。

不成立↵

Γ|= p，则根据语义后乘的定义可知，对任何一阶结构 M,只要 M|=Γ 成立，则有 M|=p。对比题目，题目中所说的是"对于一切解释 I"，对于具体的每个解释 I=（M，V，v），其中的 M 不一定是 Γ 的模型。显然若 M 是 Γ 的模型，则结论成立。但 M 若不是 Γ 的模型，则可按照如下构造反例：↵

↵

由 UG 规则的有效性，对任何的一阶结构 M,M|=p 等价于 M|=∀xp,即{p}|=∀xp。取一阶结构 M=（R，φ，<），p=R(x,y)，取 V：将 x 指派为 0，y 指派为 1,则对某个特定的 I 有 I（p）=t（即 0<1），满足对所有 q∈Γ，都有 I（q）=t；但此时 I（∀xp）=f（若将 x 指派某个比 1 大的实数，此时 x>y，故 I（∀xp）=f），即违反了 UG 规则的有效性↵

综上，题目中所述的判断不成立↵

不成立。

Γ|=p 即表示了对任何一阶结构M,只要M|=Γ(即所有以M为一阶结构的解释I，满足对 所有q ∈Γ, 都有I(q)=t)，则M|=p

而题目的条件为 "对一切解释I"，此处每次的解释I=(M,V,v)是取定的，但实际上M未必是 Γ 的模型。在此情况下，可通过构造一阶结构M来推翻思考题所述

eg：由语义性质：UG有效性可知，对任何一阶结构M，M |= p当且仅当M|= ∀xp，所以易得{p} |= ∀xp。（即取Γ={p},p=∀xp）。此时取一阶结构M = (R, φ,>),p= P(x,y)(P为二元谓词符号，其映射为M中的 >)，取定指派V，使得x指派为2，y指派为1，则对特定解释I = (M,V,v)，有I(p) = t，此时满足题述对所有 q ∈Γ 有I(q)=t，但是取I的变体I x/0,即令x指派为0时，有I x/0(p) = f，所以I(∀x p)=f，与题设矛盾

2.3“真”在一阶逻辑中有哪几个层次?

> 三个层次，分别为
>
> - 解释域中的可满足公式，公式 p 是在某个解释域 M 中的非恒假，$\exists \varphi \in \Phi_M,\ |p|(\varphi) = 1$
> - 解释域中的恒真公式，公式 p 在某个解释域 M 中恒真，$\forall \varphi \in \Phi_M,\ |p|(\varphi) = 1$，记为 $|p|_M = 1$
>   - 语义推论，公式 p 在任意一个公式集 Γ 的模型 M 中 p 恒真，记为 $\Gamma \vDash p$
> - 有效式，公式 p 在 K 的所有解释域中恒真，记作 $\vDash p$
>
> 其中注意到语义推论是对某个公式集而言的，有效式也可以看做是空集的语义推论，因此对谓词逻辑整体而言可以认为是一个层次。

三个层次:

1. M可满足。若存在一个一阶解释 I = (M, V, v) ,使得 I(p) = t ，则p是M可满足的。
2. M有效。取定一个M，若对一切V，p在 I=(M,V,v) 下有I(p) = t，则 p是M有效的，称M为p的一个模型，记为M|=p
3. M逻辑有效。对一切一阶结构M，M|=p成立，则p是逻辑有效的。

由定义可知三个层次对真的要求逐渐增加

补充:

- 不同类型的真
  - 经验命题：符合论
  - 数学命题：贯通论
  - 祈使句、问句等等

第三章

3.1 本节(3.1节)尝试给出的?Γ = {(P1),(P2),(P3),(P4),(P5)}是否完全表达了自然数的Peano定义?

> 没有完全表达↵
> 首先没有给出满足自然数中相等关系的“=”的定义，“=”的解释是不确定的，可能不满足自然数中的相等关系；另一方面，由非正规模型的存在性定理可知，数学中的相等关系无法被完全形式化描述，这与 Peano 中相等的定义矛盾，故没有完全表达↵

没有完全表达，因为二元谓词=没有定义，这意味着在任何M模型中，=的解释是不确定的，甚至可能不是自然数集上的相等关系

而且由3.2节的非正规模型存在性可知，数学中的相等关系无法被完全形式化，也就是说即使3.2节引入的K+也无法将peano定义完全形式化，因为存在模型使得K+中的等词无法解释为相等，与Peano中的相等定义产生矛盾。

3.2 L是否“强迫”->解释为实质蕴涵?

> 是的，由于 MP 规则与（语义）演绎定理的存在，蕴涵词 $\rightarrow$ 必须解释为实质蕴涵，可以尝试参考前文 1.2 的方法，以 L1、L2 与 MP 规则确定蕴涵词的真值表。

L“强迫”->解释为实质蕴涵，考虑与 L1、L2、L3 及 MP 规则在语义上的真假作对比↵
考虑 p->q 的真假，对 p、q 进行指派，共有四种指派，在实质蕴涵下只有 I（p）=t，I（q）=f 时 I（p->q）=f，其他情况下 I（p->q）=t，故考虑以下四种情况↵

1. 当 I（p）=t，I（q）=f 时，若 I（p->q）=t,则和 MP 规则相矛盾（由 MP 规则，I（p）=t，I（p->q）=t 可推出 I（q）=t，矛盾）↵
2. 当 I（p）=f，I（q）=f 时，若 I（p->q）=f，则同理可得 I（q->（p->q））=f，由语法和语义的关系可知，这与 L1 为永真式矛盾↵
3. 当 I（p）=t，I（q）=t 时，若 I（p->q）=f，则 I（q->（p->q））=f（由 1 推出矛盾得），由语法和语义的关系可知，这与 L1 为永真式矛盾↵
4. 当 I（p）=f，I（q）=t 时，若 I（p->q）=f，取某个公式 r，并使得 I（r）=t，结合前面推出的矛盾，有 I（p->(q->r)）=f(前件 p 为假，后件 q->r 为真),I((p->q)->(p->r))=t(前件 p->q 为假，后件 p->r 为假)，这时 I((p->(q->r))->((p->q)->(p->r)))=f，由语法和语义的关系可知，这与 L2 为永真式矛盾↵

综上可知，L“强迫”->解释为实质蕴涵↵

L“强迫”->解释为实质蕴涵：考虑I(p->q)的真假，

1. 当 I(p) = f, I(q) = f 时，若I(p->q) = f, 则I(q->(p->q))=f,与**L1为永真式的要求矛盾**；
2. 当 I(p) = t, I(q) = f 时，若I(p->q) = t，则此时**MP规则**矛盾；
3. 当I(p)=t，I(q)=t，若I(p->q)=f，那么 I(q->(p->q))=f （由2），与L1永真式要求矛盾；
4. 当 I(p) = f, I(q) = t 时，若I(p->q) = f，则任取I(r)=t，可推知I((p->(q->r))->((p->q)->(p->r))) = f，（用到了1,2,3的证明）与L2为永真式的要求矛盾；

综上所述，L强迫->解释为实质蕴涵，否则会推出L1,L2,L3以及MP规则恒成立的矛盾

这也说明等词和蕴含词有本质上的区别：等词并不能“强迫”

3.3 Frege组合原则在一阶语义中的具体表现是什么？并举例说明你的看法。

3.3↵
具体表现↵
一阶语义包含了一阶结构（包括函数符号的语义【K（Y）中 n 元函数符号到 F 中 n 元函数的映射】和谓词的语义【K（Y）中 n 元谓词符号到 P 中 n 元关系的映射】）、个体变元的解释（指派）、个体常元的解释（K（Y）中个体常元到论域 D 中个体的映射）以及联结词的语义（真值表）、量词的语义，任何一阶公式都是由上述若干个部分复合而成的↵
故任何一阶公式都可以层层拆解为多个原子公式的组合，通过原子公式的真假并结合联结词、量词的含义可以推导出更高层公式的真假↵
举例↵
对于一阶公式$\forall x\exists y R_1^2(x,y)\rightarrow\exists y R_1^2(f_1^1(x),y)$和一阶结构 M=（D，$\{f_1^1\},\{R_1^2\}$），由一阶结构 M 来解释 $R_1^2(x,y)$，$R_1^2(f_1^1(x),y)$的语义，再由一阶解释 I=（M，V，v）解释联结词的语义并进行指派，确定 $R_1^2(x,y)$，$R_1^2(f_1^1(x),y)$的真假，并结合量词、联结词的含义，最终通过结构归纳（复合）得到整个一阶公式的语义（真假）↵

表现:一阶公式的语义由非逻辑符号的映射，个体变元指派(相对于一阶结构M)，和联结词的语义(标准赋值)，全称量词的解释经复合而成,复合即指结构归纳

举例:即可以取定一个一阶结构M，以及一个相对于M的个体变元指派，这样，对于任意一个公式p，在Frege组合原则作用下，I(p)由其结构层次被层层拆开，I依次作用于其子结构，最后得到p的真值。

3.4 递归函数与常见的程序设计语言有哪些相似之处

- 编程可实现的函数都是递归函数（可计算的）
- 可计算性理论是计算机科学的理论基础之一，递归函数论和图灵机是其两大理论支柱。图灵机是一种抽象计算机，它为可计算数的计算提供了强的有力的计算手段，是计算机的雏形和冯.诺依曼计算机的理论模型，影响了传统计算机的设计思想。**递归函数理论作为元计算机科学理论基础，明确了计算机可计算的对象——递归函数类。**图灵论题：一个函数是可计算的当且仅当图灵机是可计算的。而图灵机可计算的函数正是递归函数类，确切地说是一般递归函数类。图灵机和递归函数论有着天然的联系。递归函数论与计算机科学的实践有着密切的关系，**递归的思想直接影响了程序设计语言的构造，进一步影响了计算机系统的结构**
- 什么是递归

  递归是一种定义对象类的方法，这个方法称为递归定义。存在一定的原始对象，在被定义的类中，我们给出一些方法，运用这些方法要给定某些已知在类中的对象中，可以产生类中的新对象。被定义的类中的全体成员恰好而且仅仅是那些由原始对象开始，反复使用所给定的获取新对象的规则而引进的新对象。这种定义类的方法称为递归定义。递归函数类可以用递归来定义，它由原始对象经过复合运算、原始递归运算和极小化运算而得到。
- 这三个基本函数的运算能力是强有力的。原始函数通过复合生成基本递归函数类，继而根据原始递归运算生成递归函数类，再经过极小化运算即可得到部分递归函数类。**由此可见复合运算、原始递归运算和极小化运算在数学体系中的关键作用，这也使得我们推想它们在程序设计中的重要性，它们与程序基层控制机制密切相关。**
- **构造性是计算机系统的最根本特征，也是能行过程的重要特征，而递归是最具代表性的构造性数学方法。递归函数类就是递归构造得到的，而程序是构造性证明，图灵机理论已经蕴含了程序设计的思想。我们用函数方法构造一个可计算函数f（或者等价地构造性证明f是可计算的），就等于对f进行程序设计。**Turing-Church论题告诉我们：递归函数类等同于能行可计算函数。而**复合运算、原始递归运算和极小化运算发挥了很关键的作用，所以旨在提供通用计算功能的计算机必须实现这些运算。程序设计语言中的许多设施都是为了构造递归函数类而设置的。过程(Procedure)和宏(Macro)为运用复合提供了明显的便利。为了应付复合和原始递归运算的定义，一些程序设计语言，如ALGOL68,还提供了以函数作为输入参数和输出结果的特殊过程。这样复合运算可以解释成提供过程的功能。极小化运算实质上包含了对具有检测出口条件的循环进行程序设计的能力，它使用了结构程序设计中的Do While运算。原始递归运算不像极小化运算的功能那样强，它也包含循环，但在某些情况下，循环次数是预先确定的，这与使用任意检测在每一步确定终止条件是否满足的情形正好相反。使用Do While和FOR循环均可实现原始递归运算。**
- 结构化程序设计最初由Dijkstra提出，1966年G.Jacopini和C.Bohn从理论上证明了：**任何单入口、单出口程序仅使用序列、循环和条件三种结构就可以表示出来。这种程序设计方法采用了反复运用几个简单运算的思想，用这三种结构我们可以很方便完成递归函数理论的基本运算，从而可以表达任意计算过程。**
- **在原始递归中如果我们开始就知道执行的次数，则完全可以用for循环来实现。由于全函数经过极小化运算后可能不是全函数，即不存在最小y的解，从而结构程序设计中Do While循环可能出现死循环，这在程序设计中要值得注意。**利用结构程序设计语言提供的控制机制，我们可以很容易构造出原始函数和复合、原始递归和极小化运算。下面是用C语言函数实现的三个原始函数和三个运算，利用这六个函数可以计算任何可计算函数，完成递归理论的运算。可见程序设计的能力和递归函数理论表达能力是相同的，函数的构造和程序设计是等价的。

  ◆ **零函数** ◆原始递归运算

  int zero(int x) int recursion(int y)

  { {

  return(0); int k,h;

  } h=f(x1,…,xn);

  ◆ **后继函数** while(y<>k)

int successor(int x) {

{ h=g(x1,…,xn,k,h);

return(x+1); k++;

} }

◆ **广义单位函数** return(h);

int projection(int i,int x1,int x2,…,int xn) }

{ ◆**极小化运算**

if(i==1)return(x1); int minmalisation()

if(i==2)return(x1); {

… int y,k;

if(i==n)return(xn); y=0;

} k=f(x1,…,xn,y);

◆ **复合运算** while(k<>0)

int composition() {

{ y++;

int y1,y2,…,yk; k=f(x1,…,xn,y);

y1=f1(x1,…,xn); }

y2=f1(x1,…,xn); return(y);

… }

yk=f1(x1,…,xn);

return(g(y1,…,yk));

}

- **程序设计是计算机实践的基本活动，而递归思想直接影响了程序设计的构造，它决定了程序设计语言的许多关键技术，从而递归过程、递归程序就十分自然了。**简单地说，递归过程就是自身调用自身过程，包含递归过程的程序我们称为递归程序。在程序设计中使用递归算法往往可以简化求解问题的复杂性。如著名的梵天塔问题，这个问题只有用递归方法解决，而不能用其他方法有效求解。用递归过程描述如下：

  void 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);

  }

  }

- **在编译技术中递归过程使用更为普遍，处理符号表达式递归过程尤为自然，因为这样的程序结构和数据结构相匹配。**下面是Pascal语言一个子集TINY语言的文法（图7），从文法中我们可以看到许多递归结构，所以语法分析采用递归下降文法进行分析自然就很简单。我们为每一文法成分构造一个函数，这些函数之间相互调用，直接或间接就构成了递归。

- 体系结构方面的影响
    - 递归程序中的变量与机器中的存储单元对应，当程序被它本身调用时，它将使用相同的存储单元改写它们原先的内容。所以有了栈这样的数据结构，用来存储必须保存的寄存器内容。这一存储在进入子程序前由调用程序完成或在子程序使用前完成。为了增强递归程序设计的效率，现在的计算机体系结构中设有硬件栈数据结构，它由一组寄存器组成，并且用专用的指令push和pop来处理栈操作。由此可见，递归思想在计算机系统中是非常普遍的，它甚至影响了计算机的体系结构。**
- 总结：递归函数理论作为计算机科学的元科学，对计算机实践具有非常重要的意义，而程序设计作为计算机科学基本的实践活动，**递归思想直接影响了程序设计语言的构造，也影响了计算机的体系结构。递归是一种普遍的思维机制，这在计算机科学的理论和实践中得到了很广泛的应用，对计算机的发展起了至关重要的作用。**