011094，中国科学技术大学，2020年春季学期

数理逻辑讲义

陈小平
计算机科学与技术学院

杨金龙摄

3.5 递归函数

回顾：K_N可表示函数和关系

❖ **定义1（K_N可表示函数）** 一个k元函数g是K_N可表示的，如果存在一个含k+1个自由变元的K_N公式$p(x_1, \ldots, x_{k+1})$，使得对任意对$p(x_1, \ldots, x_{k+1})$中x_{k+1}自由的项u及$n_1, \ldots, n_k, n_{k+1} \in \mathbf{N}$有

1. 如果$g(n_1, \ldots, n_k) = n_{k+1}$则$\vdash_{K_N} p(\underline{n_1}, \ldots, \underline{n_k}, \underline{n_{k+1}})$；
2. 如果$g(n_1, \ldots, n_k) \neq n_{k+1}$则$\vdash_{K_N} \neg p(\underline{n_1}, \ldots, \underline{n_k}, \underline{n_{k+1}})$；
3. $\vdash_{K_N} p(\underline{n_1}, \ldots, \underline{n_k}, u) \to u = \underline{g(n_1, \ldots, n_k)}$.

❖ **注释** “大部分”数论函数不是K_N可表示的。但是，可计算的数论函数都是K_N可表示的。什么是可计算函数？

回顾：判定问题

- **可判定** 一类问题是可判定的，如果该类问题的每一个实例只有肯定/否定二种回答，并且存在一个能行方法A，使得对该类问题的每一个实例：(1)如果回答是肯定的，则A在有限步骤内输出yes；(2)如果回答是否定的，则A在有限步骤内输出no。
- **能行方法的原始定义** 由有限多个机械步骤所组成的过程，其中每个机械步骤的执行都在有限时间内完成。
- **注释** 历史上，先在逻辑学中出现了判定和能行方法的概念，之后产生了可计算性的定义，进而形成了理论计算机科学。关键人物是图灵，既是逻辑学家，又是计算机科学的创始人之一。

回顾：判定问题

- **命题演算L的可判定性** 存在一个能行方法A，对任何L公式p，当$\vdash_L p$成立时，A在有限时间内回答“是”；当$\nvdash_L p$不成立时，A在有限时间内回答“否”。
- **一阶谓词演算K的半可判定** 任给一阶公式p是不是K的内定理($\vdash_K p$是否成立)是半可判定的，即存在一个能行方法A，当$\vdash_K p$成立时A在有限时间内回答“是”。
- **观察** 命题演算的形式证明$\vdash_L p$与一阶谓词演算的形式证明$\vdash_K p$为什么存在可判定性/可计算性的区别？
- **观察** 能行方法最初的严格定义是什么？

3.5 递归函数

❖ **定义1（基本函数）** 以下三种数论函数称为基本函数：

1. 一元零函数z，$z(n) = 0$；
2. 一元后继函数s，$s(n) = n+1$；
3. k元投影函数p_i^k，$p_i^k(n_1, \ldots, n_k) = n_i$, $i=1, \ldots, k$。

◆ **观察** 三种基本函数体现了“能行方法”的直观理解，是“能行方法”直观描述的具体化。因此，三种基本函数都被认为是能行可计算的函数。

3.5 递归函数

❖ 定义2（复合规则） 一个i元函数g和i个k元函数g_1, …, g_i的复合是一个k元函数

$$c(n_1, \ldots, n_k) =_{df} g(g_1(n_1, \ldots, n_k), \ldots, g_i(n_1, \ldots, n_k))$$

◆ 注释 复合规则是从i+1个现有函数g, g_1, …, g_i组成一个新函数c的规则。

❖ 观察 如果函数g, g_1, …, g_i都是能行可计算的，则函数c也是能行可计算的；因此，复合规则具有“保能行可计算性”。

❖ 观察 复合规则从函数组合的角度扩展了能行可计算性概念。

3.5 递归函数

❖ **定义3（递归规则）** 由k元函数g和k+2元函数f使用递归规则生成的k+1元函数r是一个递归函数：

$$\begin{cases} r(n_1, \ldots, n_k, 0) =_{df} g(n_1, \ldots, n_k); \\ r(n_1, \ldots, n_k, n+1) =_{df} f(n_1, \ldots, n_k, n, r(n_1, \ldots, n_k, n)); \end{cases}$$

$k = 0$时，

$$\begin{cases} r(0) =_{df} g; \\ r(n) =_{df} f(n, r(n)). \end{cases}$$

❖ **观察** 如果g和f都是能行可计算的，则r也是能行可计算的。

3.5 递归函数

❖ **定义4（μ算子）** 设k+1元函数g满足根存在条件：对任意自然数n_1, …, n_k存在自然数x使得$g(n_1, \ldots, n_k, x) = 0$。应用μ算子于g生成k元函数f:

$$f(n_1, \ldots, n_k) =_{df} \min\{x \mid g(n_1, \ldots, n_k, x) = 0\}$$

记为$f(n_1, \ldots, n_k) =_{df} \mu x[g(n_1, \ldots, n_k, x) = 0]$。

❖ **观察** 如果函数g是能行可计算的并且满足根存在条件，则函数f也是能行可计算的。

3.5 递归函数

❖ 定义5(递归函数)

1. 三个基本函数及由它们经有限次应用三个规则生成的函数称为(一般)递归函数；

2. 不使用μ算子生成的递归函数称为原始递归函数；

3. μ算子的使用不要求根存在条件的递归函数称为部分递归函数。

❖ 观察 三个基本函数是能行可计算的，三个规则的应用保持能行可计算性，所以一般递归函数是能行可计算的。

❖ 观察 任何可计算函数由三个基本函数和三种规则组合而成。

3.5 递归函数

❖ 例1 (和函数的递归性) 2元函数+ (加法运算)是一个递归函数，其递归定义如下：

$$\begin{cases} n_1 + 0 = p_1^1(n_1); \\ n_1 + (n + 1) = s(p_3^3(n_1, n, n_1+n)). \end{cases}$$

因此，和函数+是由投影函数p_1^1和 p_3^3 以及后继函数s，使用递归规则生成的。

❖ 类似可证2元函数×(乘法运算)是一个递归函数。

3.5 递归函数

❖ 例2（符号函数的递归性）1元函数sg（sg(0)=0; sg(n)=1, n≥1）是一个递归函数，其递归定义如下：

$$\begin{cases} sg(0) = 0; \\ sg(n+1) = sz(p_1^2(n, sg(n)). \end{cases}$$

因此，符号函数sg是由投影函数、后继函数s和零函数z使用递归规则生成的。

3.5 递归函数

❖ **例3（余数函数的递归性）** 2元函数余数函数：

$$\begin{cases} rem(n_1, n_2) = 0, n_1 = 0; \\ rem(n_1, n_2) = n_1\text{除}n_2\text{的余数}, n_1 > 0. \end{cases}$$

是一个递归函数（证明略）。

❖ **其他例** 常值函数、截差函数、指数函数、累加函数∑、累乘函数∏、极大值函数max、极小值函数min……是递归函数。

❖ **注释** 递归函数是事实上第一个可计算函数的严格定义，也是原理上第一个函数式程序设计语言，但未明确提出可计算性和程序的概念。后由图灵和车赤完成可计算性的定义。

3.5 递归函数

- 定义6(递归关系） 若k元关系R的特征函数C_R是递归函数，则称R为递归函数；一元递归关系称为递归集。
- 例4 二元关系=、≤、<是递归关系。证明：自修。
- 例5 二元关系divi $=_{df}$ $\{((n_1, n_2) \mid n_1=0$或者n_1能整除$n_2\}$是递归关系。证明：自修。
- 例6 全体素数的集合是递归集。证明：自修。

3. 5 递归函数

❖ 思考题

3.4 递归函数与常见的程序设计语言(如过程式程序设计语言C语言）有哪些相似之处？