COMP9020 Math Cheat Sheet

基础概念

- **Floor Function (下取整):** `⌊x⌋` 表示小于或等于 `x` 的最大整数。
- **Ceiling Function (上取整):** `⌈x⌉` 表示大于或等于 `x` 的最小整数。
- **Absolute Value (绝对值):** `|x|` 表示 `x` 的非负值。

难点:

- 上下取整常用于简化整数计算。例如，若 `⌊x⌋ = ⌈x⌉`，则表明 `x` 必为整数。
- **例题**：证明 `⌊x⌋ = ⌈x⌉` 意味着 `x` 是整数。
  - **解答思路**：使用夹逼原理 `⌊x⌋ ≤ x ≤ ⌈x⌉`，当等号两边相等时，`x` 必为整数。

整除性与最大公约数 (GCD) 和最小公倍数 (LCM)

- **Divisibility (整除):** 如果存在整数 `k` 使得 `n = k * m`，则 `m` 整除 `n`，记作 `m | n`。
- **GCD (最大公约数)**：能同时整除 `m` 和 `n` 的最大整数。
- **LCM (最小公倍数)**：能同时被 `m` 和 `n` 整除的最小整数。
- **性质**： gcd 和 lcm 的关系为 `gcd(m, n) * lcm(m, n) = |m| * |n|`。

难点:

- **例题**：证明如果 `ab | bc`，则 `a | c`。
  - **解答思路**：根据整除定义，`ab | bc` 表示存在整数 `k` 使 `bc = kab`，两边同除以 `b` 得 `c = ka`，故 `a | c`。

模运算与余数

- **定义:** `m div n = ⌊m/n⌋` 和 `m % n = m - n * ⌊m/n⌋`。
- **目标:** 给定整数 `m` 和 `n`，求 `m div n` 和 `m % n`。
- **技巧**：对大数取余时可寻找幂的循环模式，以简化计算。

难点:

- **例题**：求 `7^7^7` 的最后两位。
  - **解答思路**：通过模 100 获取最后两位，对 `7` 的幂取模，找到循环规律 `74 ≡ 1 (mod 100)`，然后简化问题为 `73 % 100 = 43`。

例题 2：计算区间 20 到 365 中是 3 或 5 的倍数的整数个数

解题思路

1. **容斥原理**：将问题分解为计算区间内是 3 的倍数、5 的倍数和 15（3 和 5 的最小公倍数）倍数的整数个数。

好的，这里是将公式转化为 ASCII 格式的版本：

对于任意的区间 `[a, b]`，可以使用以下公式计算该区间内能被 `k` 整除的整数个数：

```
count(k) = floor(b / k) - floor((a - 1) / k)
```

其中，`floor(x)` 表示将 `x` 向下取整的操作。

在这个题目中，区间给定为 `[20, 365]`，即 `a = 20`，`b = 365`。接下来，我们计算区间 `[20, 365]` 中分别为 3、5 和 15 的倍数的整数个数：

1. **3 的倍数**：

   ```
   count(3) = floor(365 / 3) - floor(19 / 3)
   ```

2. **5 的倍数**：

   ```
   count(5) = floor(365 / 5) - floor(19 / 5)
   ```

3. **15 的倍数**（这是为了去除重复计数的部分）：

   ```
   count(15) = floor(365 / 15) - floor(19 / 15)
   ```

最后，为了计算区间内能被 3 或 5 整除的整数个数，可以通过容斥原理得到结果：

```
count(3 or 5) = count(3) + count(5) - count(15)
```

这里使用容斥原理的原因是，3 和 5 的倍数有一些重叠的部分（即同时是 15 的倍数的数），所以需要将这些重复计数的部分减去。

欧几里得算法 (Euclidean Algorithm)

- **算法概念:** 欧几里得算法通过递归计算来求解最大公约数 (GCD)，其规则如下：
  - 若 `gcd(m, n)`，则
    1. 当 `n = 0` 时，`gcd(m, 0) = m`。
    2. 否则，递归使用 `gcd(m % n, n)`，直到余数为 0。

详细过程:

1. **步骤1**: 若 `a > b`，计算 `a % b` 得余数 `r`。
2. **步骤2**: 替换 `(a, b)` 为 `(b, r)`，重复步骤1，直到余数 `r = 0`。

3. **步骤3**: 最后一个非零余数即为 `gcd(a, b)`。

- **例题**：使用欧几里得算法计算 `(56, 72)` 的 GCD。
  - **解题过程**:
    1. `gcd(56, 72)`：计算 `72 % 56 = 16`，所以继续计算 `gcd(56, 16)`。
    2. `gcd(56, 16)`：计算 `56 % 16 = 8`，所以继续计算 `gcd(16, 8)`。
    3. `gcd(16, 8)`：计算 `16 % 8 = 0`，因此 `gcd(56, 72) = 8`。

重要结论

- **两个数互质的条件**：若 `gcd(a, b) = 1`，则称 `a` 和 `b` 为互质数。
- **连续整数的互质性**：任意整数 `n` 与 `n+1` 总是互质，因为其 GCD 必为 1。

练习题提示

- 对于模运算问题，如 `7^7^7` 的最后两位，可使用循环节来减少计算。
- 欧几里得算法是求解 gcd 的最有效方法，适用于大数。

Set Theory

集合概念与符号

- **集合 (Set)**：一组不重复对象的集合，用大写字母表示。
- **元素 (Element)**：`x ∈ A` 表示 `x` 是集合 `A` 的元素。
- **子集 (Subset)**：若 `A ⊆ B`，则 `A` 的所有元素均属于 `B`。
  - **真子集 (Proper Subset)**：若 `A ⊂ B` 且 `A ≠ B`，则 `A` 是 `B` 的真子集。
- **空集 (Empty Set)**：记作 `∅`，表示不包含任何元素的集合。
- **全集 (Universal Set)**：包含所有可能元素的集合，记作 `U`。

集合运算

- **并集 (Union)**：`A ∪ B = {x : x ∈ A or x ∈ B}`。
- **交集 (Intersection)**：`A ∩ B = {x : x ∈ A and x ∈ B}`。
- **补集 (Complement)**：`Ac = {x : x ∉ A and x ∈ U}`。
- **差集 (Difference)**：`A \ B = A ∩ Bc`，即 `A` 中不属于 `B` 的元素。
- **对称差 (Symmetric Difference)**：`A ⊕ B = (A \ B) ∪ (B \ A)`，包含仅在 `A` 或 `B` 中的元素。

难点

- **例题**：证明对于任意集合 `A` 和 `B`，若 `A ⊆ B`，则 `A ∩ B = A`。
  - **解答思路**：
    1. **A ∩ B ⊆ A**：若 `x ∈ A ∩ B`，则 `x ∈ A`，所以 `A ∩ B ⊆ A`。
    2. **A ⊆ A ∩ B**：若 `x ∈ A` 且 `A ⊆ B`，则 `x ∈ B`，因此 `x ∈ A ∩ B`，得 `A ⊆ A ∩ B`。

幂集与基数

- **幂集 (Power Set)**：`Pow(A)` 表示所有 `A` 的子集构成的集合。
- **基数 (Cardinality)**： `|A|` 表示集合 `A` 的元素数量。

难点

- **例题**：证明若 `A` 是含 `n` 个元素的有限集合，则 `|Pow(A)| = 2^n`。
  - **解答思路**：每个元素有包含或不包含两种选择，因此共有 `2^n` 个子集。

集合律与代数性质

- **交换律 (Commutativity)**：`A ∪ B = B ∪ A`，`A ∩ B = B ∩ A`。
- **结合律 (Associativity)**：`(A ∪ B) ∪ C = A ∪ (B ∪ C)`，`(A ∩ B) ∩ C = A ∩ (B ∩ C)`。
- **分配律 (Distribution)**：`A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)`，`A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)`。
- **德摩根律 (de Morgan's Laws)**：`(A ∩ B)c = Ac ∪ Bc`，`(A ∪ B)c = Ac ∩ Bc`。

难点

- **例题**：证明 `A \ (A ∩ B) = A \ B`
  - **解答思路**：
    1. `A \ (A ∩ B) = A ∩ (A ∩ B)c`。
    2. 根据德摩根律 `(A ∩ B)c = Ac ∪ Bc`，将其代入得 `(A ∩ Ac) ∪ (A ∩ Bc) = ∅ ∪ (A ∩ Bc) = A \ B`。

Venn 图示

Venn 图是一种用重叠圆圈展示集合关系的工具，便于直观理解集合运算。

难点

- **例题**：利用 Venn 图证明 `A ∩ (B ∪ C) ≠ (A ∩ B) ∪ C`
  - **解答思路**：绘制两侧表达式的 Venn 图，观察其中差异。对于不同的情况选择元素验证左、右两式结果不等。

形式语言 (Formal Languages)

- **字母表 (Alphabet)**：有限的符号集，通常记作 `Σ`。
- **单词 (Word)**：由 `Σ` 中符号组成的有限序列。
- **语言 (Language)**：由 `Σ*` 中符合某种特定规则的单词集合。
- **闭包运算 (Kleene Star)**：`A*` 表示 `A` 的所有可能串联组合，包括空串。

难点

- **例题**：给定 `A = {ab, ba}`，求 `A0`, `A1`, 和 `A2`。
  - **解答思路**：
    1. `A0 = {λ}` (空串)。
    2. `A1 = A = {ab, ba}`。

    3. `A2 = {abab, abba, baab, baba}`，即 `A` 中元素的所有两次串联组合。

语言的运算性质

- **串联 (Concatenation)**：`AB = {ab : a ∈ A and b ∈ B}`。
- **交集的闭包性**：如果 `w ∈ (L1 ∩ L2)*`，则 `w ∈ L1* ∩ L2*`。

难点

- **例题**：证明 `(L1 ∩ L2)* ⊆ L1* ∩ L2*`
  - **解答思路**：
    1. 若 `w ∈ (L1 ∩ L2)*`，则 `w = w1w2...wn`，其中每个 `wi ∈ L1 ∩ L2`。
    2. 因为 `wi` 同时属于 `L1` 和 `L2`，因此 `w ∈ L1*` 且 `w ∈ L2*`，即 `w ∈ L1* ∩ L2*`。

重要结论

- **集合互斥性**：`A` 和 `Ac` 互为补集，因此 `A ∩ Ac = ∅` 且 `A ∪ Ac = U`。
- **幂集大小**：若集合 `A` 含有 `n` 个元素，则 `|Pow(A)| = 2^n`。
- **形式语言的运算封闭性**：语言 `A*` 包含所有可能的串联组合，是包含空串的无限集合。

Relations

二元关系的定义

- **关系 (Relation)**：集合 `A` 和 `B` 之间的关系 `R` 是 `A × B` 的子集。如果 `(a, b) ∈ R`，则称 `a R b`。
- **图表示法**：用点代表元素，用箭头表示关系。
- **矩阵表示法**：在矩阵中用 `•` 表示关系 `(a, b) ∈ R`。

难点:

- **例题**：给定 `A = {2, 3, 4, 6}` 和 `B = {1, 2, 3}`，`a R b` 表示 `gcd(a, b) = 1`。
  - **解答思路**：
    1. **关系集表示**：列出满足条件的 `(a, b)`。
    2. **图表示**：将关系绘制成带箭头的图。
    3. **矩阵表示**：在对应的矩阵位置填 `•`。

关系的运算

- **逆关系 (Converse)**：`R← = {(b, a) ∈ B × A : a R b}`。
- **复合关系 (Composition)**：`R;S = {(a, c) ∈ A × C : 存在 b ∈ B 使 a R b 且 b S c}`。

难点:

- **例题**：计算 `R;R←`，并确定其是否为 `A × A` 的子集。
  - **解答思路**：逐一检查 `(a, c)` 是否存在 `b` 使得 `a R b` 且 `b R← c`，找出所有满足条件的 `(a, c)`。

关系的性质

- **自反性 (Reflexive)**：对于所有 `a ∈ A`，有 `(a, a) ∈ R`。
- **反自反性 (Antireflexive)**：对于所有 `a ∈ A`，有 `(a, a) ∉ R`。
- **对称性 (Symmetric)**：若 `(a, b) ∈ R`，则 `(b, a) ∈ R`。
- **反对称性 (Antisymmetric)**：若 `(a, b) ∈ R` 且 `(b, a) ∈ R`，则 `a = b`。
- **传递性 (Transitive)**：若 `(a, b) ∈ R` 且 `(b, c) ∈ R`，则 `(a, c) ∈ R`。

难点:

- **例题**：确定 `R = {(a, b) ∈ Z×Z : a ≤ b}` 的自反性、对称性等性质。
  - **解答思路**：
    - **自反性**：`a ≤ a` 对所有 `a ∈ Z` 成立，故 `R` 自反。
    - **对称性**：若 `a < b`，则不一定有 `b < a`，故非对称。
    - **传递性**：若 `a ≤ b` 且 `b ≤ c`，则 `a ≤ c`，所以传递。

等价关系与等价类

- **等价关系 (Equivalence Relation)**：关系满足自反性、对称性和传递性。
- **等价类 (Equivalence Class)**：给定 `a ∈ A`，`[a] = {b ∈ A : a R b}`。

难点:

- **例题**：证明 `w1 ~ w2` 表示 `w1` 和 `w2` 含有相同字母数是等价关系。
  - **解答思路**：
    1. **自反性**：任意单词与自身含有相同字母数。
    2. **对称性**：若 `w1 ~ w2`，则 `w2` 也与 `w1` 含相同字母数。
    3. **传递性**：若 `w1 ~ w2` 且 `w2 ~ w3`，则 `w1 ~ w3`。

偏序关系与 Hasse 图

- **偏序关系 (Partial Order)**：关系满足自反性、反对称性和传递性。
- **Hasse 图**：用于表示偏序关系的简化图，仅显示具有直接关系的节点。
  - **极小元素**：没有其他元素小于该元素。
  - **极大元素**：没有其他元素大于该元素。

难点:

- **例题**：在集合 `{2, 4, 6, 9, 12, 36, 72}` 上定义关系 `|`，绘制 Hasse 图并找出极小/极大元素。
  - **解答思路**：
    1. **Hasse 图**：将每个元素和其因子关系用箭头表示。
    2. **极小/极大元素**：找出图中没有指向的元素（极小）和没有被指向的元素（极大）。

格与全序关系

- **格 (Lattice)**：偏序集满足每对元素都有最小上界 (lub) 和最大下界 (glb)。
- **全序关系 (Total Order)**：所有元素对 `(a, b)` 中，满足 `a ≤ b` 或 `b ≤ a`。

- **拓扑排序 (Topological Sort)**：对偏序集进行全序排列，使得 `a ≤ b` 表示 `a` 排在 `b` 前。

难点：

- **例题**：对偏序集 `Pow({a, b, c})` 进行拓扑排序。
  - **解答思路**：按子集的大小排序，得到 `{}`, `{a}`, `{b}`, `{c}`, `{a, b}`, `{a, c}`, `{b, c}`, `{a, b, c}`。

重要结论

- **自反性和对称性**：一个关系在满足对称和传递性时不一定自反，但若加上自反性，则为等价关系。
- **唯一极大元素**：偏序集中若存在极大元素，则该元素唯一。

关系的表示范例

关系的矩阵表示和图表示

例题：设集合 `A = {1, 2, 3}` 和 `B = {a, b}`，定义关系 `R ⊆ A × B` 满足以下条件：

- `(1, a), (2, a), (2, b)` 为 `R` 的元素。

我们可以将 `R` 表示成如下的矩阵和图：

矩阵表示

	a	b
1	•	
2	•	•
3		

- 其中 `•` 表示 `(a, b) ∈ R`。
- 行代表集合 `A` 的元素，列代表集合 `B` 的元素。

图表示

```
A   B

1 → a
2 → a
2 → b
```

- 图表示法中，箭头从 `A` 中的元素指向 `B` 中的元素，以表示关系的方向。

Hasse 图的范例

例题：设集合 `A = {1, 2, 4, 8}`，定义偏序关系 `|` (整除关系)，即若 `a | b`，则存在 `(a, b) ∈ R`。

Hasse 图表示

1. 绘制元素之间的整除关系，并只保留直接关系。
2. 省略自反关系 (即箭头自指) 和传递关系 (即间接整除)。

- 在 Hasse 图中，每个节点指向其被整除的元素，方向从下到上。
- `1` 是极小元素 (最小的整除因子)，`8` 是极大元素 (所有其他元素的倍数)。

这些图表帮助我们更直观地理解关系及其在偏序集中的应用。

布尔逻辑

- **布尔值集合**：`B = {0, 1}`，主要运算包括：
  - **非 (NOT)**：`!x = 1 - x`
  - **与 (AND)**：`x && y = min{x, y}`
  - **或 (OR)**：`x || y = max{x, y}`
- **布尔代数定律**：
  - **交换律**：`x || y = y || x`，`x && y = y && x`
  - **结合律**：`(x || y) || z = x || (y || z)`，`(x && y) && z = x && (y && z)`
  - **分配律**：`x || (y && z) = (x || y) && (x || z)`，`x && (y || z) = (x && y) || (x && z)`

难点：

- **例题**：化简 `[x && (x && !y)] || [(x && y) || (y && !x)] = x || y`
  - **解答思路**：
    1. **观察整体结构**：将表达式分成几个小部分，尝试应用布尔代数的交换律、结合律、分配律逐步简化。
    2. **应用恒等律和补全律**：将不必要的项去除，使用 `x || (!x) = 1` 和 `x && (!x) = 0` 等基础公式。
    3. **整理结果**：最后得到简化形式 `x || y`。

函数与其性质

- **函数 (Function)**：对于集合 `X` 到 `Y` 的二元关系 `f ⊆ X × Y`，若每个 `x ∈ X` 对应唯一的 `y ∈ Y`，则 `f` 是一个函数。
- **单射 (Injective)**：若 `f(a) = f(b)` 则 `a = b`。
- **满射 (Surjective)**：对于每个 `y ∈ Y`，存在 `x ∈ X` 使得 `f(x) = y`。
- **双射 (Bijective)**：若函数既单射又满射。

难点：

- **例题**：判断给定的二元关系是否为函数，并确定其定义域、值域和映射结果。
  - **解答思路**：
    1. **检查唯一性**：确保每个 `x ∈ X` 都对应唯一的 `y ∈ Y`。

2. **确定函数的性质**：根据映射关系判断其是否为单射或满射。
3. **求解定义域、值域和映射结果**：确认 `Dom(f)`、`Codom(f)` 和 `Im(f)`，理解函数的作用范围。

复合函数与逆函数

- **复合函数 (Composition)**：若 `f : X → Y` 和 `g : Y → Z`，则 `(g ∘ f)(x) = g(f(x))`。
- **逆函数 (Inverse Function)**：若 `f` 为双射，则存在唯一的逆函数 `f−1`，满足 `f−1(f(x)) = x` 且 `f(f−1(x)) = x`。

难点：

- **例题**：证明 `g(x) = (x - 5) / 3` 是 `f(x) = 3x + 5` 的逆函数。
  - **解答思路**：
    1. **求解 `g(f(x))` 和 `f(g(x))`**：分别计算 `g(f(x))` 和 `f(g(x))`，验证结果为 `IdX` 和 `IdY`。
    2. **结论**：若 `g(f(x)) = x` 且 `f(g(x)) = x`，则 `g` 是 `f` 的逆函数。

主范式：合取范式 (CNF) 和析取范式 (DNF)

- **合取范式 (CNF)**：布尔表达式以**最大项**形式存在，即形如 `(m1 && m2 && ... && mn)` 的表达式。
- **析取范式 (DNF)**：布尔表达式以**最小项**形式存在，即形如 `(m1 || m2 || ... || mn)` 的表达式。

难点：

- **例题**：将 `(x || y) && (!x || !y)` 转换为析取范式。
  - **解答思路**：
    1. **列出所有可能输入组合**：构建真值表，找出使表达式为真的组合。
    2. **写出对应的最小项**：对于每个满足条件的组合，写出相应的 `x` 和 `y` 形式的最小项。
    3. **组合最小项**：将所有满足条件的最小项用 `||` 连接起来，即得到析取范式结果 `(!x && y) || (x && !y)`。

大-O 记号

- **O-符号 (Big-O Notation)**：若 `f(n) ∈ O(g(n))`，则存在常数 `n0` 和 `c > 0`，使得对所有 `n ≥ n0`，有 `f(n) ≤ c * g(n)`。
- **Ω-符号 (Big-Omega Notation)**：若 `f(n) ∈ Ω(g(n))`，则存在常数 `n0` 和 `c > 0`，使得对所有 `n ≥ n0`，有 `f(n) ≥ c * g(n)`。
- **Θ-符号 (Big-Theta Notation)**：若 `f(n) ∈ Θ(g(n))`，则 `f(n)` 同时属于 `O(g(n))` 和 `Ω(g(n))`。

难点：

- **例题**：确定 `f(n) = 4n + 2` 与 `g(n) = n^2 - 4` 的增长关系。
  - **解答思路**：
    1. **验证 O-记号条件**：找到 `c` 和 `n0` 使得对所有 `n ≥ n0`，`f(n) ≤ c * g(n)`，从而满足 `f(n) ∈ O(g(n))`。

2. **验证 Ω-记号条件**：判断是否存在 `c` 和 `n0` 使得 `f(n) ≥ c * g(n)` 成立，若不满足则 `f(n) ∉ Ω(g(n))`。
3. **综合结论**：若 `f(n)` 满足 `O(g(n))` 但不满足 `Ω(g(n))`，则 `f(n) ∉ Θ(g(n))`。

重要结论

- **布尔代数简化技巧**：使用交换、结合、分配、恒等、补全和幂等律简化复杂表达式。
- **双射函数的逆**：双射函数存在唯一的逆函数，满足 `f(f−1(x)) = x`。
- **大-O 符号的对偶关系**：若 `f(n) ∈ O(g(n))`，则 `g(n) ∈ Ω(f(n))`。

卡诺图 (Karnaugh Map)

- **定义**：卡诺图 (Karnaugh Map) 是一种简化布尔函数的方法，适用于2到4个变量的布尔表达式，帮助最小化最小项数量，以获得最简析取范式 (DNF) 表达式。
- **使用规则**：
  - 将布尔表达式中的每种变量组合在卡诺图上标出。
  - 通过合并相邻的"1"形成方块（覆盖区域），每个方块大小必须为 `2^n` (如1、2、4个格子)。
  - 方块可沿卡诺图边界环绕，找到覆盖所有"1"所需的最少方块数量。

例题

- **题目**：将 `(x̄y) ∨ (xy)` 转换为最小项数量的析取范式 (DNF)。
  - **解题步骤**：
    1. **构建卡诺图**：将各项 `(x̄y)` 和 `(xy)` 的对应格子标记为"1"。
    2. **合并项**：观察相邻的"1"，合并成一个方块，尽量减少表达式中的项数。
    3. **写出最简表达式**：最终得到最小项形式。

逻辑符号 (Logical Symbols) 与逻辑运算

- **命题 (Proposition)**：能够判断真假（真/假）的陈述。
- **基本逻辑符号**：
  - `∧` (Conjunction)：和 (and)，表示"并且"。
  - `∨` (Disjunction)：或 (or)，表示"或"。
  - `¬` (Negation)：非 (not)，表示"不是"。
  - `→` (Implication)：条件 (if...then)，表示"若...则..."，充要条件。
  - `↔` (Biconditional)：双向条件 (if and only if)，表示"当且仅当"。

难点题

- **例题**：将自然语言翻译为逻辑符号表示。
  - **题目**：若"天气晴朗"则"我会遛狗"。
  - **解题思路**：
    1. **定义命题符号**：假设 `p = "天气晴朗"` 和 `q = "我会遛狗"`。
    2. **转换为逻辑表达式**：根据题意，翻译为 `p → q`。

良构公式 (Well-Formed Formula, WFF)

- **定义**：良构公式 (WFF) 是满足逻辑结构和语法规范的公式，通常由基本命题和逻辑连接词构成。
- **构造规则**：
  - 单个命题变元、`>` (总真命题)、`⊥` (总假命题) 本身就是良构公式。
  - 若 `φ` 和 `ψ` 是良构公式，则 `¬φ`、`φ ∧ ψ`、`φ ∨ ψ`、`φ → ψ` 和 `φ ↔ ψ` 也是良构公式。

例题

- **题目**：判断公式 `((p ∨ p) ∧ (p¬q))` 是否为良构公式。
  - **解题步骤**：
    1. **逐项分析**：检查每个子公式的结构是否符合良构公式的定义。
    2. **判断正确性**：确保每个逻辑操作符的用法和位置合理。例如，`p¬q` 不符合良构公式定义。

真值赋值 (Truth Assignment)

- **定义**：真值赋值是一种函数，为每个命题赋予一个真值（0表示假，1表示真）。
- **公式**：
  - `v(>) = 1`，表示恒为真；`v(⊥) = 0`，表示恒为假。
  - **逻辑运算**：
    - `v(¬φ) = 1 - v(φ)` (取反)。
    - `v(φ ∧ ψ) = min{v(φ), v(ψ)}` (取最小值表示与)。
    - `v(φ ∨ ψ) = max{v(φ), v(ψ)}` (取最大值表示或)。
    - `v(φ → ψ) = max{1 - v(φ), v(ψ)}` (条件运算)。
    - `v(φ ↔ ψ) = (1 + v(φ) + v(ψ)) % 2` (双向条件)。

难点题

- **例题**：给定赋值 `v(p) = 1`，`v(q) = 0`，`v(r) = 1`，计算 `¬((p→ q) ∨ ¬r)` 的真值。
  - **解题步骤**：
    1. **计算内部子表达式**：首先求 `p → q` 和 `¬r` 的真值。
    2. **逐步代入求值**：根据给定赋值逐步代入每个子表达式的值，最后得出整个表达式的真值。

真值表 (Truth Table)

- **定义**：真值表用于枚举每种真值组合下逻辑表达式的结果。
- **结构**：每一行表示一种真值组合，列表示每个子公式的真值，最后一列为整个公式的真值。

难点题

- **例题**：为命题 `¬((p ∧ q) → (p ∧ q))` 构建真值表。
  - **解题步骤**：
    1. **列出所有组合**：列出 `p` 和 `q` 的所有可能真值组合。
    2. **分步求解**：逐步计算子公式的真值，填入每一列。
    3. **得出最终结果**：最后一列显示整个公式的真值。

逻辑等价 (Logical Equivalence)

- **定义**：若公式 `φ` 和 `ψ` 在所有真值赋值下的结果相同，则称 `φ` 和 `ψ` 逻辑等价，记作 `φ ≡ ψ`。
- **证明方法**：
  - **真值表比较**：逐行检查真值表，确保所有组合下 `φ` 和 `ψ` 真值一致。
  - **使用等价律**：使用逻辑等价律直接变换。
  - **构建恒真公式**：证明 `φ ↔ ψ` 恒为真。

难点题

- **例题**：证明 `¬p → (q ∨ r) ≡ q → (¬p → r)`。
  - **解题步骤**：
    1. **构建真值表**：列出所有可能的真值组合，计算每个表达式的真值。
    2. **比较结果**：逐行检查两边表达式的真值是否一致，以判断等价性。

蕴涵与有效性 (Entailment and Validity)

- **定义**：蕴涵关系 `φ1, ..., φn |= ψ` 表示在所有满足前提的真值赋值下，结论 `ψ` 也为真。
- **验证方法**：
  - **真值表法**：构建前提和结论的真值表，验证在所有前提为真时，结论也为真。
  - **恒真公式法**：证明 `(φ1 ∧ ... ∧ φn) → ψ` 恒为真。

难点题

- **例题**：证明 `p ∧ ¬p |= q`。
  - **解题步骤**：
    1. **观察矛盾**：`p ∧ ¬p` 本身构成矛盾，逻辑上为假。
    2. **推导**：在矛盾条件下，可以推导出任意结论，因此 `p ∧ ¬p` 可以蕴涵任何命题 `q`。

逻辑等价律 (Logical Equivalence Laws)

- **常用等价律**：
  - **交换律 (Commutativity)**：`p ∨ q ≡ q ∨ p`，`p ∧ q ≡ q ∧ p`
  - **结合律 (Associativity)**：`(p ∨ q) ∨ r ≡ p ∨ (q ∨ r)`，`(p ∧ q) ∧ r ≡ p ∧ (q ∧ r)`
  - **分配律 (Distribution)**：`p ∨ (q ∧ r) ≡ (p ∨ q) ∧ (p ∨ r)`，`p ∧ (q ∨ r) ≡ (p ∧ q) ∨ (p ∧ r)`
  - **双重否定 (Double Negation)**：`¬¬p ≡ p`
  - **逆否命题 (Contrapositive)**：`p → q ≡ ¬q → ¬p`
  - **德摩根律 (De Morgan's Laws)**：`¬(p ∨ q) ≡ ¬p ∧ ¬q`，`¬(p ∧ q) ≡ ¬p ∨ ¬q`

重要结论

- **逻辑等价的证明**：可以通过真值表、等价律或构造恒真公式来证明两个命题的等价性。
- **矛盾蕴涵**：在矛盾条件下，可以推导出任意结论，因此矛盾是逻辑上最强的蕴涵条件。
