广义纯正九莲宝灯听牌型的唯一性

A Mahjong Game Formalization

钟星宇 @sun123zxy

北京理工大学

2026-07-21

目录

目录

麻将（Mahjong）和广义麻将

- 限定清一色、门前清
- n 阶麻将共 $3n$ 种点数，单一点数在一副手牌中至多出现 4 次
- 面子（meld）分为刻子（triplet）$\{i,i,i\}$ 与顺子（sequence）$\{i,i+1,i+2\}$
- 和牌型（winning hand）由一个雀头（pair）$\{i,i\}$ 与 $n+1$ 个面子组成，共 $3n+5$ 张牌
- 听牌型（wait）为和牌型去掉一张牌，共 $3n+4$ 张，不允许空听
- $n=3$ 还原经典麻将

九莲宝灯（Nine Gates）

- 标准麻将的纯正九莲宝灯是 1112345678999，摸入 1 至 9 中任意一张都能和牌，即全面听（full-sided wait）
- 它是唯一的全面听；程序验证易
- 广义 n 阶麻将，类似标准构造 $1^3 23 \cdots (3n-1)(3n)^3$ 仍全面听
- 仍然唯一？

图: 纯正九莲宝灯的和牌分解

唯一性

定理

n 阶广义麻将清一色全面听牌型列举如下：

- 当 $n = 1$ 时，共有六种：

$$\{1,2,2,2,3,3,3\} \quad \{1,1,2,2,3,3,3\} \quad \{1,1,2,2,2,3,3\}$$
$$\{1,1,1,2,2,2,3\} \quad \{1,1,1,2,2,3,3\} \quad \{1,1,1,2,3,3,3\}$$

- 当 $n \geq 2$ 时，唯一的牌型是

$$\{1,1,1,2,3,4,\ldots,(3n-1),3n,3n,3n\}$$

目录

一些历史

- Mathematical aspect of the combinatorial game “Mahjong”
  - Python verification of some small n
- 如何证明九莲宝灯是麻将中唯一的清一色九面听？- Goldenrain 的回答 - 知乎
  - 2026 年 1 月手写图片稿，7 月文字稿更新
  - Folklore，比较可信
- 我们的证明原稿为 2026 年 6 月由 Rethlas + GPT 5.5 在 ~30min 内生成，得到了和第二篇参考极其类似的证明路线. 考虑到此时上述知乎回答还是图片手写稿，难以确认 Rethlas 是否为独立完成证明.

面子分解（Meldable）

- 和牌型去掉雀头，剩余部分可面子分解（meldable）
- 若某点数的重数为 0，任何面子都不能跨过该位置，左右两侧可独立分解
- 每个刻子与顺子的点数和都被 3 整除，故可面子分解牌型的点数和被 3 整除

扫描分解

- 从左到右扫描重数，状态 (a, b) 记录前两处开始但尚未完成的顺子数
- 若 $c_i < a + b$ 则失败；否则更新为 $((c_i - a - b) \bmod 3, a)$
- 剩余牌优先组成尽可能多的刻子，余数只能在当前位置开启新顺子
- 扫描结束时状态回到 $(0, 0)$，当且仅当原牌型可面子分解

内部正性

全面听牌型中每个点数 $2 \sim 3n-1$ 都至少出现了一次.

- 假设内部某点数 k 的重数为 0，在这个缺口处拆分所有进张后的和牌型
- 分别从缺口左侧、右侧进张，左右牌数模 3 的约束迫使进张侧可面子分解
- 同一侧相邻两个进张使点数和相差 1，不可能同时满足被 3 整除的条件
- 矛盾，内部正

三消

设某 n 阶全面听的重数向量中有 $(1,1,1)$ 的模式，直接挖掉这一段得到的 $n-1$ 阶牌型还是 $n-1$ 阶全面听.

- 扫描状态依次经历 $(a,b)\to(1-a-b,a)\to(b,1-a-b)\to(a,b)$
- 删除前后的扫描状态相同，可面子分解性得到保持
- 进一步加入雀头和进张

反例约化

- 取阶数最小的非标准全面听反例；由三消，它的内部不能出现连续三个 1
- 内部正性 $+$ 总张数限制高于基线的内部位置至多有 6 个
- 用这至多 6 个位置隔开连续的 1，可得内部长度上界，最终推出 $n \leq 7$
- 因此一般唯一性只需排除有限范围 $2 \leq n \leq 7$ 的反例

目录

Archon 速通

- Archon 隔夜证明反例约化
  - AI，组合与大力出奇迹
- ~2500 lines
- 人工目标：更好 API，更少行数，更优理解

API 设计

```
-- Alternative choices:
-- `Multiset` (inductive, but not aligned with math)
-- `Finsupp` (aligned with finset sum, but noncomputable)
def Hand (n : ℕ) := Fin (3 * n) → ℕ

structure Meldable.Placement (n : ℕ) where
  tripNum : Fin (3 * n) → ℕ
  seqNum : Fin (3 * n - 2) → ℕ

def Meldable.toHand : Placement n →+ Hand n where
  toFun p :=
    ∑ i, p.tripNum i • triplet i + ∑ j, p.seqNum j • sequence j

def Meldable.IsWitness (p : Meldable.Placement n) (c : Hand n) : Prop :=
  p.toHand = c

def Meldable (c : Hand n) : Prop :=
  ∃ p : Meldable.Placement n, Meldable.IsWitness p c
```

```
abbrev IsCounterexample (c : Hand n) : Prop :=
  IsFullWait c ∧ c ≠ standardFullWait n

theorem full_wait_reduction {n : ℕ} (hn : 2 ≤ n) {c : Hand n}
    (hc : IsCounterexample c) :
    ∃ m, 2 ≤ m ∧ m ≤ 7 ∧ ∃ d : Hand m, IsCounterexample d := by
  sorry

theorem full_wait_unique (n : ℕ) (hn : n ≥ 2)
    (c : Hand n) (hc : c.IsFullWait) : c = standardFullWait n := by
  sorry
```

扫描分解

```
abbrev ScanM := StateT ScanState
  (WriterT (List StepPlacement) (Except ScanFailure))
def scanStep (x : ℕ) : ScanM Unit := do
  let s ← get
  unless s.1 + s.2 ≤ x do
    throw .failEarly
  let rem := x - (s.1 + s.2)
  let p : StepPlacement := ⟨rem / 3, rem % 3⟩
  tell [p]
  set (p.seqNum, s.1)

def scanList (xs : List ℕ) : ScanM Unit := do
  xs.forM scanStep
  unless (← get) = (0, 0) do throw .failFinally
def scanHand (c : Hand n) := scanList (List.ofFn fun i ↦ count i c) (0, 0
theorem Meldable.iff_scan {n : ℕ} (c : Hand n) :
  Meldable c ↔ (scanHand c).isOk := by sorry
```

Problems

- Monadic verification sucks
  - 不只是直接计算，还要证明程序满足的性质
  - Hoare Monads? `mvcgen`?
- 如何形式化枚举算法的正确性
- tagging `grind`

Thank you for your attention![1][2]

图: slides available online

[1] Acknowledgement: Thanks Shubin Xue for sharing this problem with me. His initial work turns into valuable prompt for AI.

[2] All graphics in this slide is generated by ChatGPT Images 2.0.