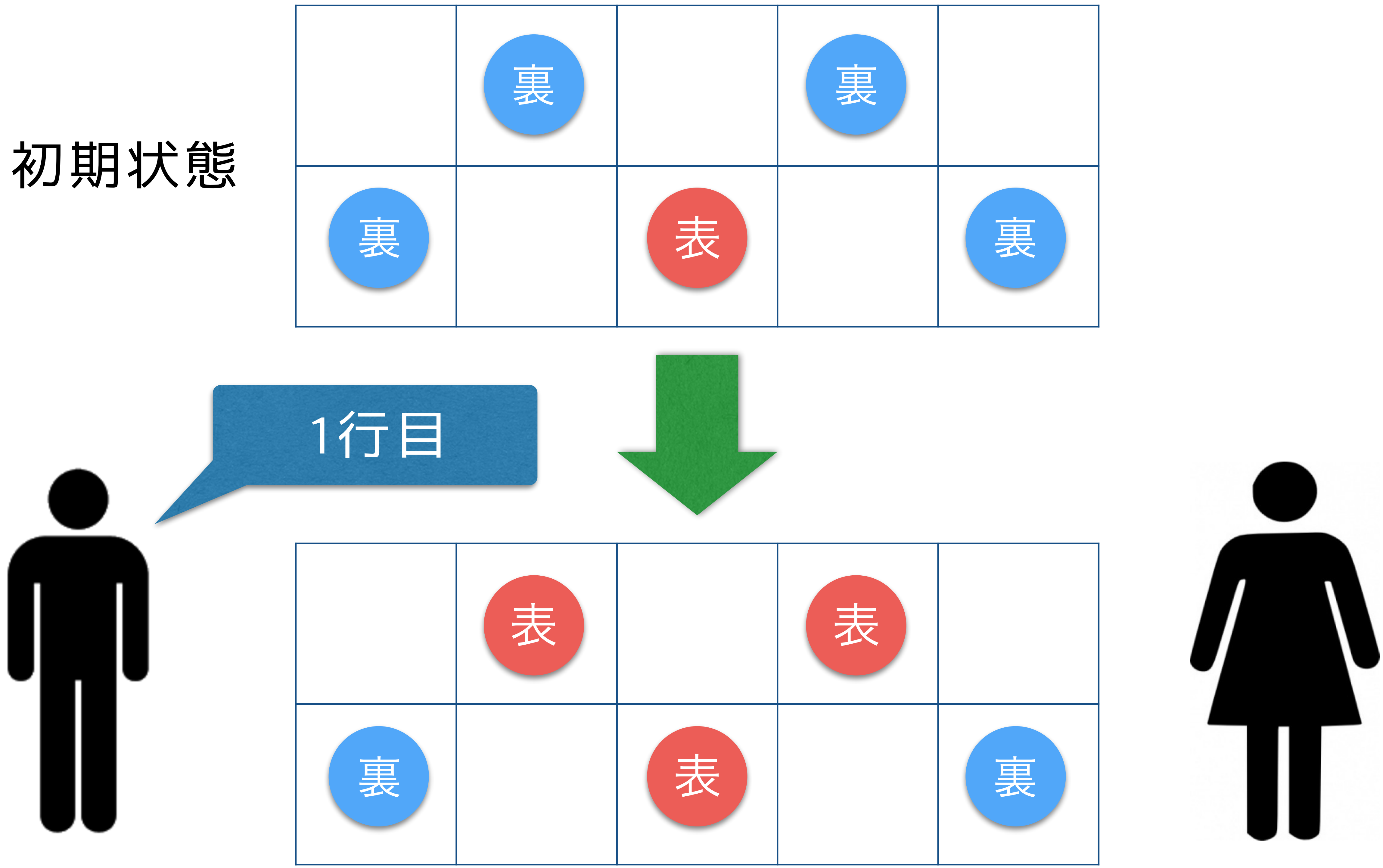
KUPC2017 - L
Coin Game 2017

原案: drafear

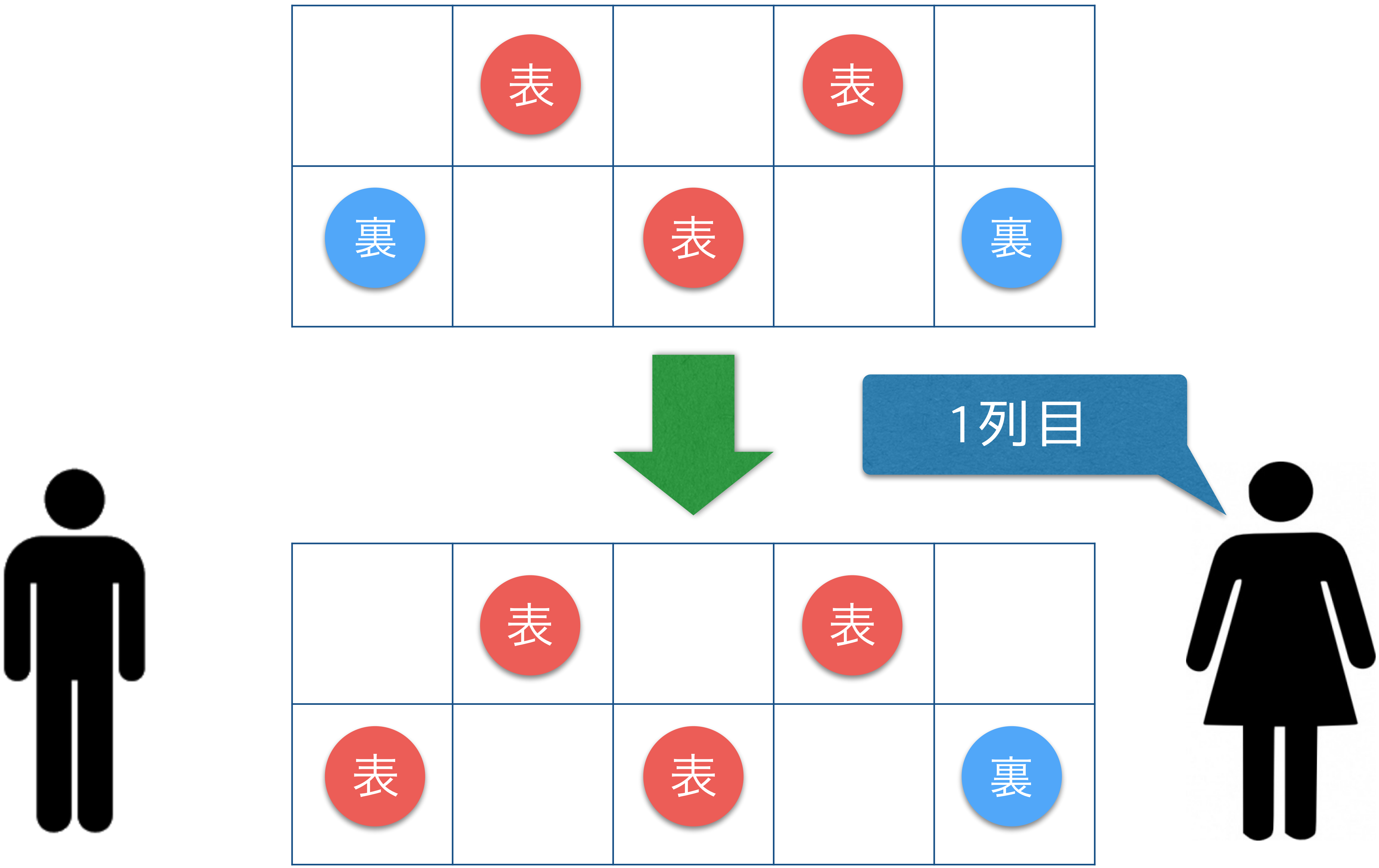
問題概要

- H × W の盤面上でゲームを行う
- 盤面の各マスは以下のいずれか
  - 表向きのコインが1枚
  - 裏向きのコインが1枚
  - 何もなし
- まだ宣言されていない行または列を宣言し、
  その行または列の全コインの裏表を反転させる
- 宣言できなくなるか
  全てのコインが表向きになったら終了
- 最後に宣言したプレイヤーは1点、全てのコインが
  表向きになっていたらお互いに追加で2点を得る
- 自分の合計点数を最大化するように最適に行動する
- 先手の合計点数を求めよ

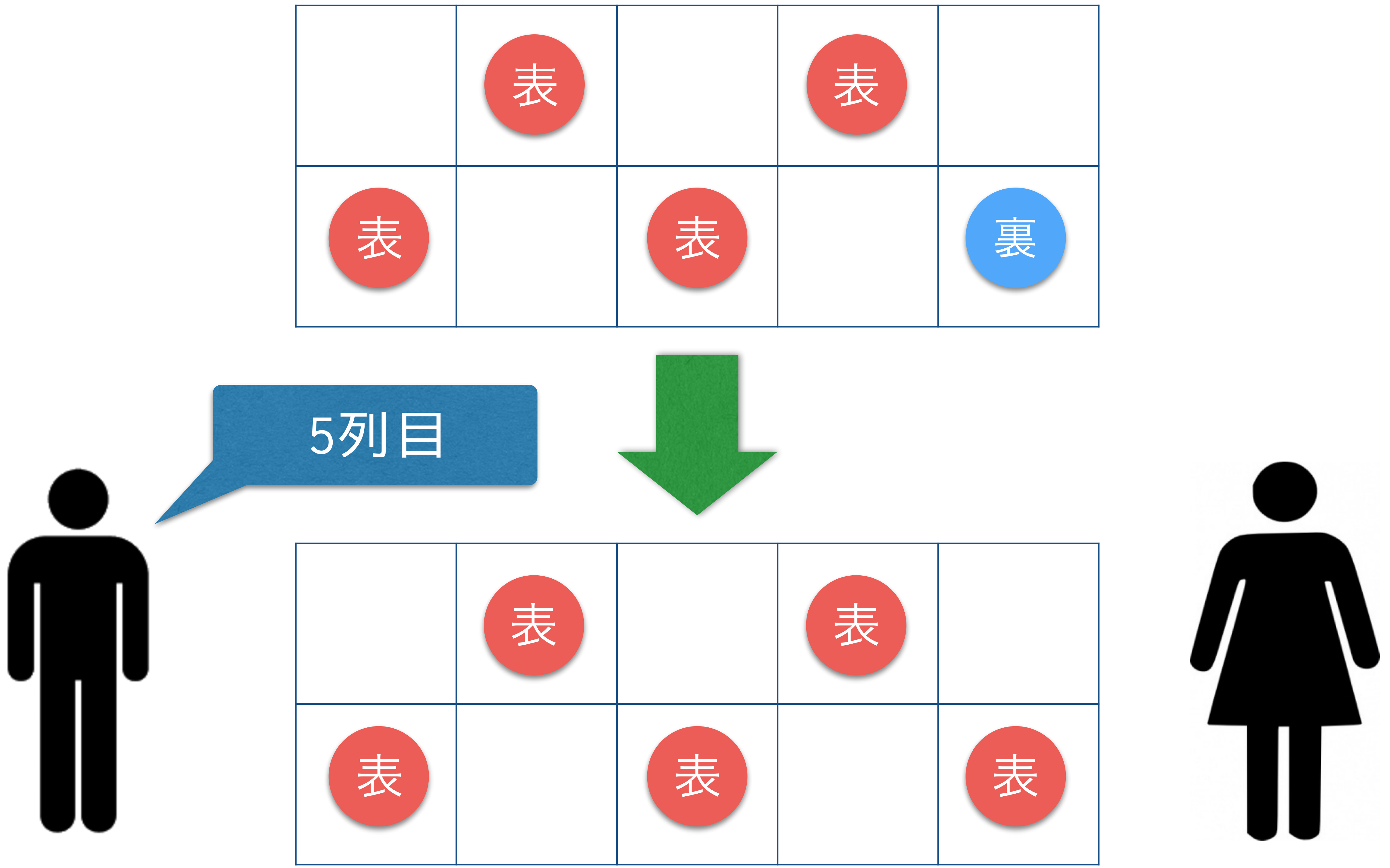
例



例

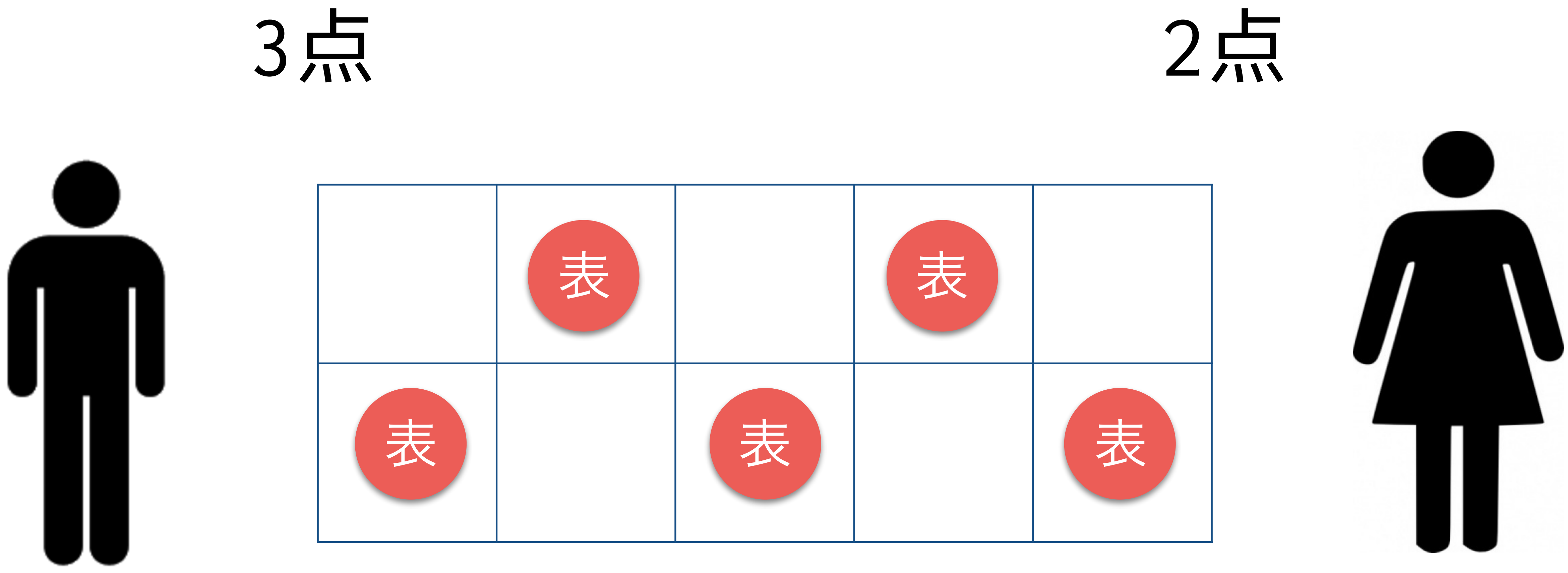


例



例

- 宣言できなくなるか
  全てのコインが表向きになったら終了
- 最後に宣言したプレイヤーは1点、全てのコインが表向きになっていたらお互いに追加で2点を得る



考察

- 1人で自由にプレイして全てを表にできる状態からは全てを表にできる状態になるような操作しかしない

- 初期盤面が全てを表にできない盤面なら
W+Hターンで終了する
  - この場合，W+Hの偶奇を見れば良い

- 初期盤面が全てを表にできるかどうか
その状態が全てを表にできるかどうか
どう判定すれば良いだろうか

考察

- 行または列が等しいコインの連結成分の中では1つ操作を決めれば残りの操作が決まる

R_i := i行目に操作を行う
C_j := j列目に操作を行う

<u>（i行目，j列目）が表</u>

$$R_i \Leftrightarrow C_j$$

$$(\neg R_i \Leftrightarrow \neg C_j)$$

<u>（i行目，j列目）が裏</u>

$$R_i \Leftrightarrow \neg C_j$$

$$(\neg R_i \Leftrightarrow C_j)$$

考察

(i行目，j列目）が表

$$R_i \Leftrightarrow C_j$$

$$(\neg R_i \Leftrightarrow \neg C_j)$$

(i行目，j列目）が裏

$$R_i \Leftrightarrow \neg C_j$$

$$(\neg R_i \Leftrightarrow C_j)$$

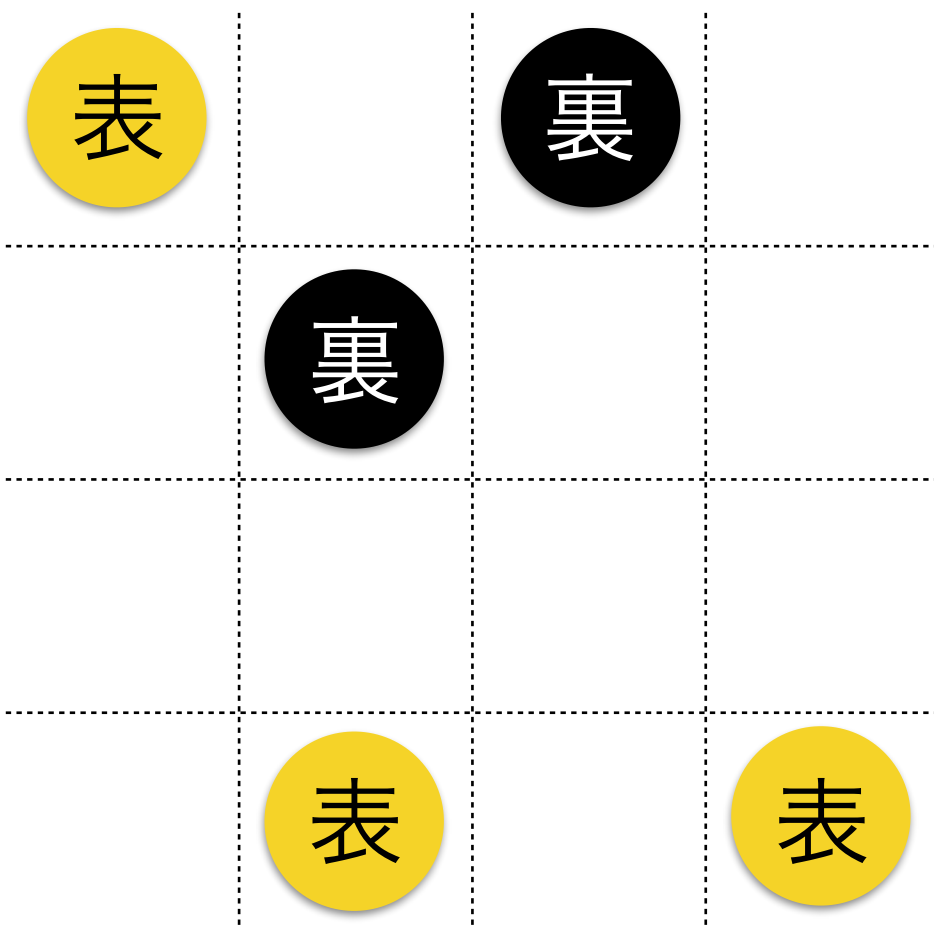
$$(\exists i.\, R_i \Leftrightarrow \neg R_i \vee C_i \Leftrightarrow \neg C_i)$$

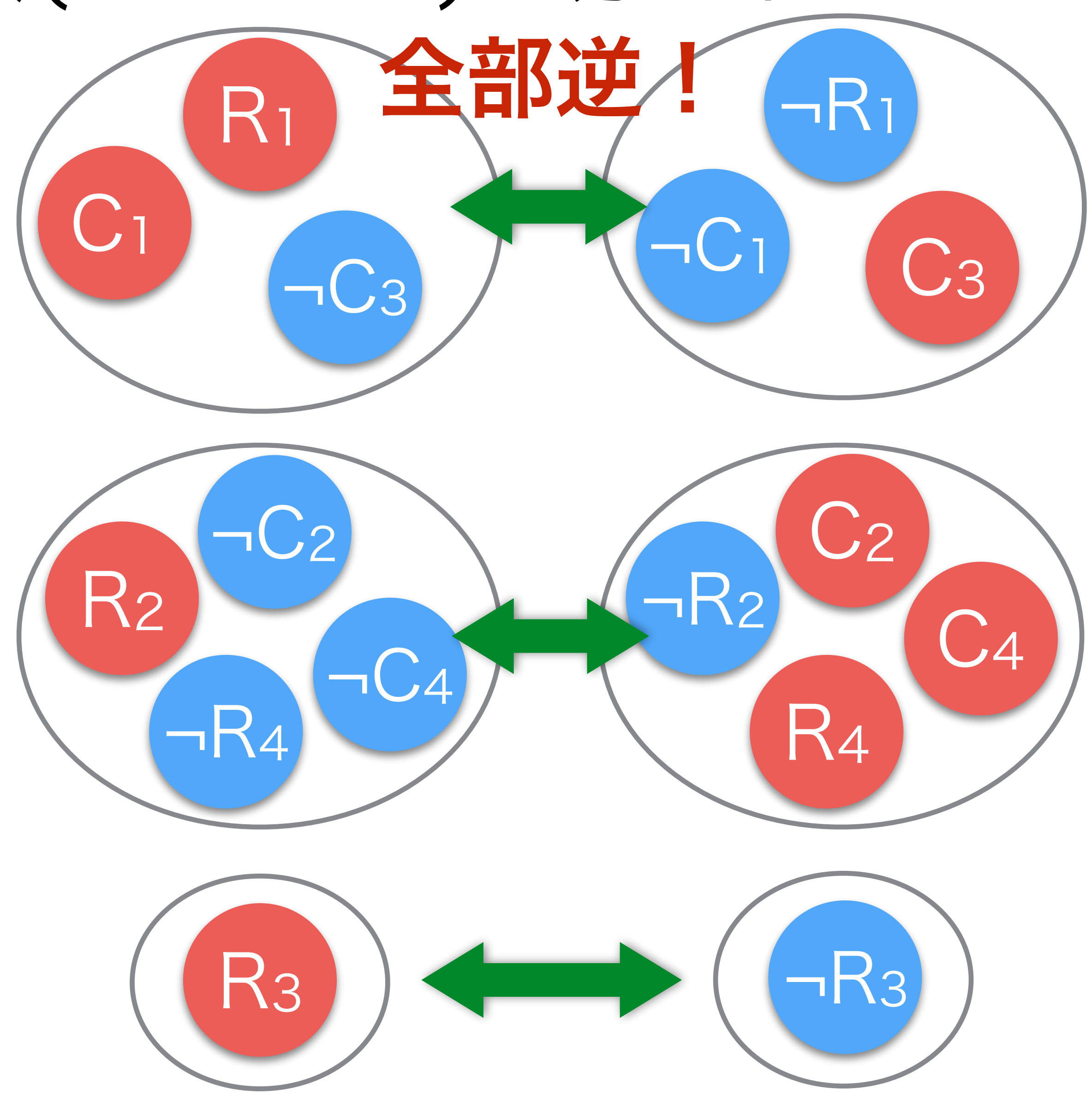
$\Leftrightarrow$ 全てを表にできない

この実装は 2-SAT(SCC) で可能だが, 同値関係なので Union-Find や 2-彩色, 単純なdfs/bfs で行えば良い

考察

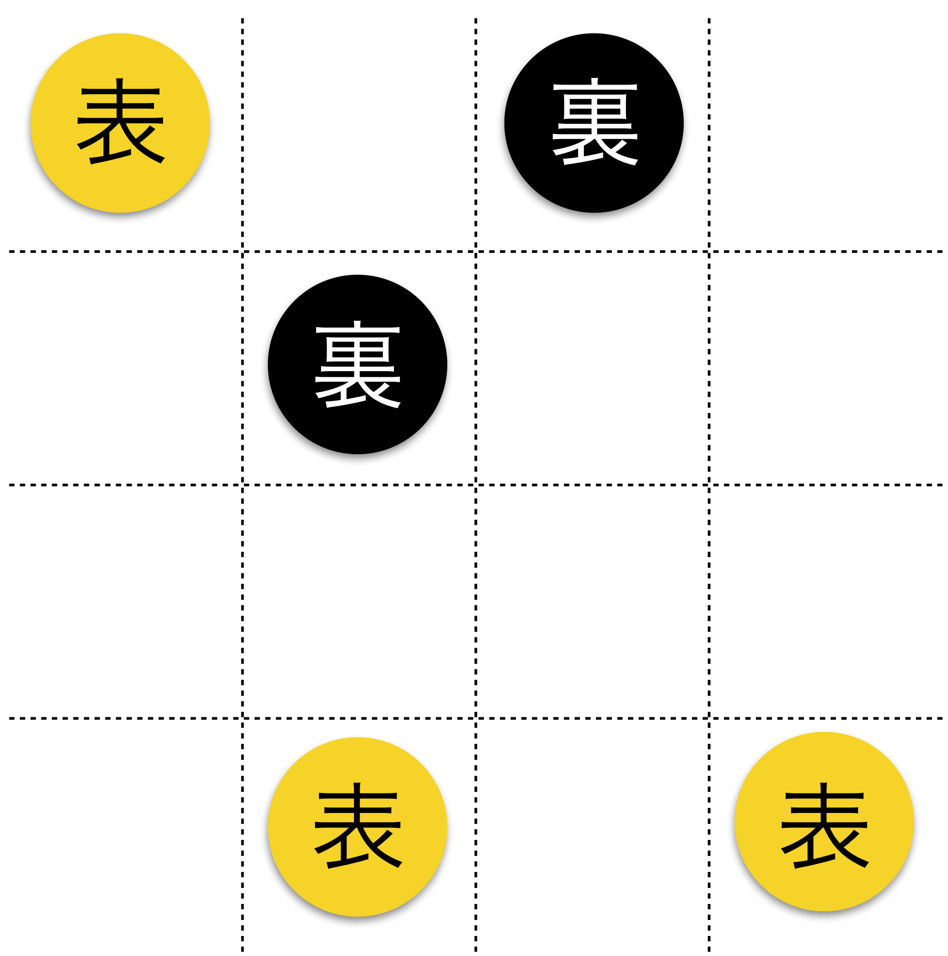
- 初期盤面が全て表にできない盤面の場合はW+Hの偶奇を見ればよかった
- 以降，初期盤面が全て表にできる盤面の場合を考える
- 2点は得られる．残りの1点を得られるか判定できれば良い
- Ri，Ciがどのような同値類(グループ)に分かれるか見てみる



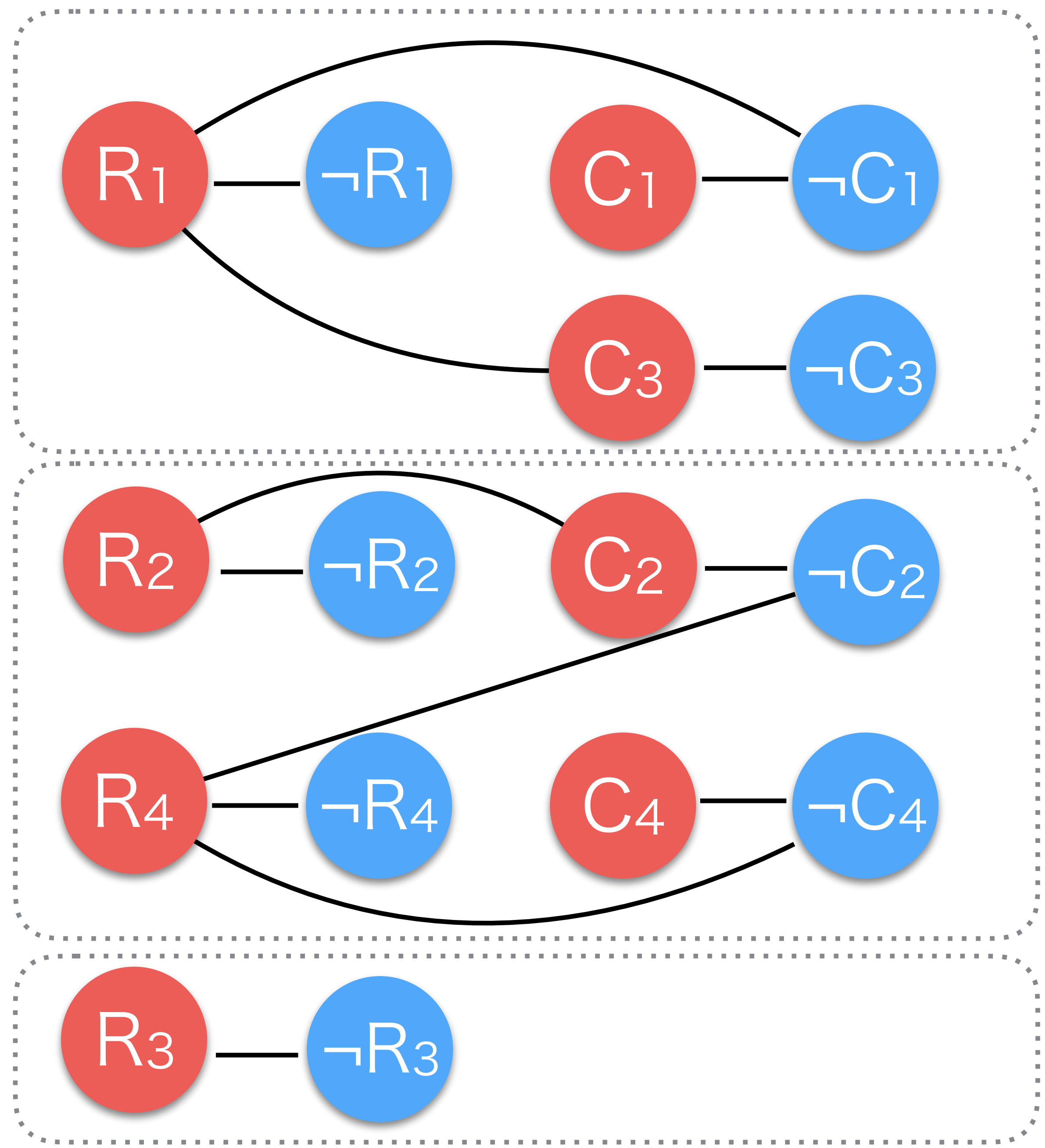


考察

- 同値類[Ri]⇔に対応する同値類[¬Ri]⇔が存在する
  - $[\neg X] = \{\neg x \mid x \in [X]\} \neq [X]$
  - 2-彩色の各連結成分を考えれば分かりやすいかも

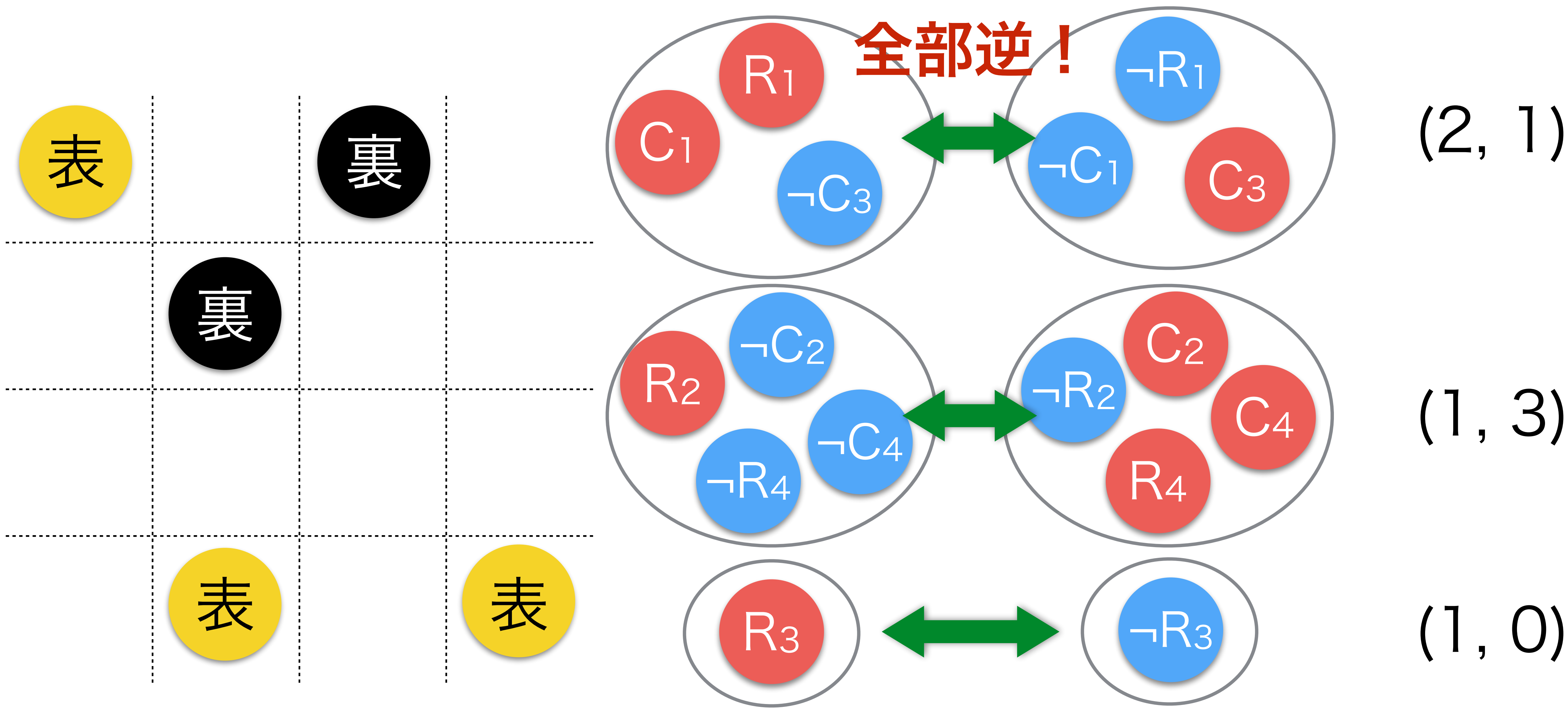


それぞれ塗り方2通り



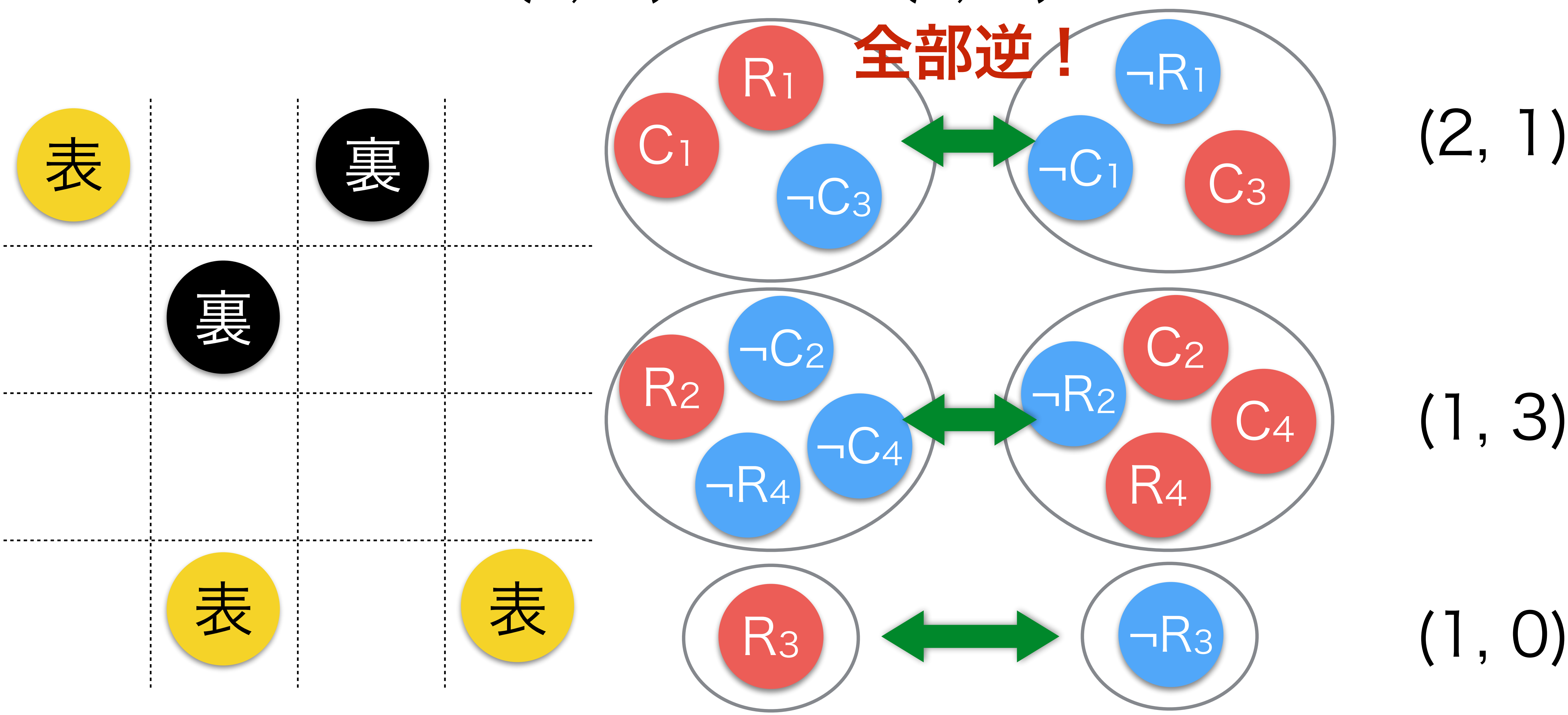
考察

- 従って，各集合から赤色の(¬の付いていない)変数を交互に1つずつ取り除いていき，最後に取り除けたら追加で1点がもらえるゲームと考えられる
- どちらかが左の集合を選んだら今後右の集合は選べず逆も同様



考察

- つまり次のゲームと等価
  - 自然数の組 (a_i, b_i) がn個ある
  - 各手番では，ある組 (a, b) ≠ (0, 0) を選んで
    (a-1, a-1) (a=0なら不可) または
    (b-1, b-1) (b=0なら不可) に置き換える
  - 全ての組を (0, *) または (*, 0) にできたら勝ち



解法(部分点)

- 初期盤面が全てを表にできない盤面の場合
  - W+Hが偶数なら0を出力する
  - W+Hが奇数なら1を出力する
- 初期盤面が全てを表にできる盤面の場合
  - (i, j)が表 $\Rightarrow (R_i \Leftrightarrow C_j)$
    (i, j)が裏 $\Rightarrow (R_i \Leftrightarrow \neg C_j)$
    に従って同値類に分解
  - 各同値類について，
    否定されていないリテラルの数を数える
  - 対応する同値類の組 (a, b) の列を作る
    - a, bは否定されていないリテラルの数

解法(部分点)

- 部分点の制約の場合，初期状態で (0, x), (x, 0) なる組は現れない
- 今後(0, x), (x, 0)が現れても (0, 0) の形しかない
- 各(a, b) についてgrandy数を計算してxorをとり
  - それが0なら後手必勝(先手は2+0=2点)
  - それが正なら先手必勝(先手は2+1=3点)
- (a, b)のgrandy数は
  (a, b)と(b, a)が等価なことに注意すれば
  - (偶数, 偶数): 0 … (奇,奇)にしか移れず(0, 0)は0
  - (偶数, 奇数): 2 … (奇,奇),(偶,偶)に移れる
  - (奇数, 奇数): 1 … (偶,偶)にしか移れない

解法(満点)

解法

- 初期状態で（0，*）や（*，0）の組があった場合が問題
- （0，*），（*，0）は同じなので（*，0）を考える
- 組の列［(a_1，b_1),⋯,(a_n，b_n)］を
  （*，0）と そうでないものに分ける
  そうでないものの列をS,
  (偶，0)の列をE，(奇，0)の列をOとする
- 列Xの要素数を|X|で表す

S:（>0，>0），⋯	E:（0，偶），⋯	O:（0，奇），⋯

解法

- 命題1．ゲームSが後手必勝で|O|が偶数の場合，
  ゲームSEOは後手必勝
  - Σmax(x，y) に関する累積帰納法で示す
  - 先手ができる操作：
    - Sから選ぶ
      → Sが後手必勝になるように後手が選べる
    - Eから選ぶ
      → 新たに追加された(奇,奇)を選ぶと同じ形になる
    - Oから選ぶ
      → Oから選ぶと同じ形になる
      (偶，偶)がSに追加されるが，そのgrandy数は0
  - また，Sが後手必勝の盤面は終了状態になり得ない

S: (>0, >0), …	E: (0, 偶), …	O: (0, 奇), …

解法

- 命題1(再掲). ゲームSが後手必勝で|O|が偶数の場合,
  ゲームSEOは後手必勝

- 系1. ゲームSが先手必勝で|O|が偶数の場合,
  ゲームSEOは先手必勝
  - 先手がSから選んで後手必勝の形にでき,
    命題1の形になる

- 系2. ゲームSが後手必勝で|O|が奇数の場合,
  ゲームSEOは先手必勝
  - 先手が|O|から選べば命題1の形になる

S: (>0, >0), …

E: (0, 偶), …

O: (0, 奇), …

解法

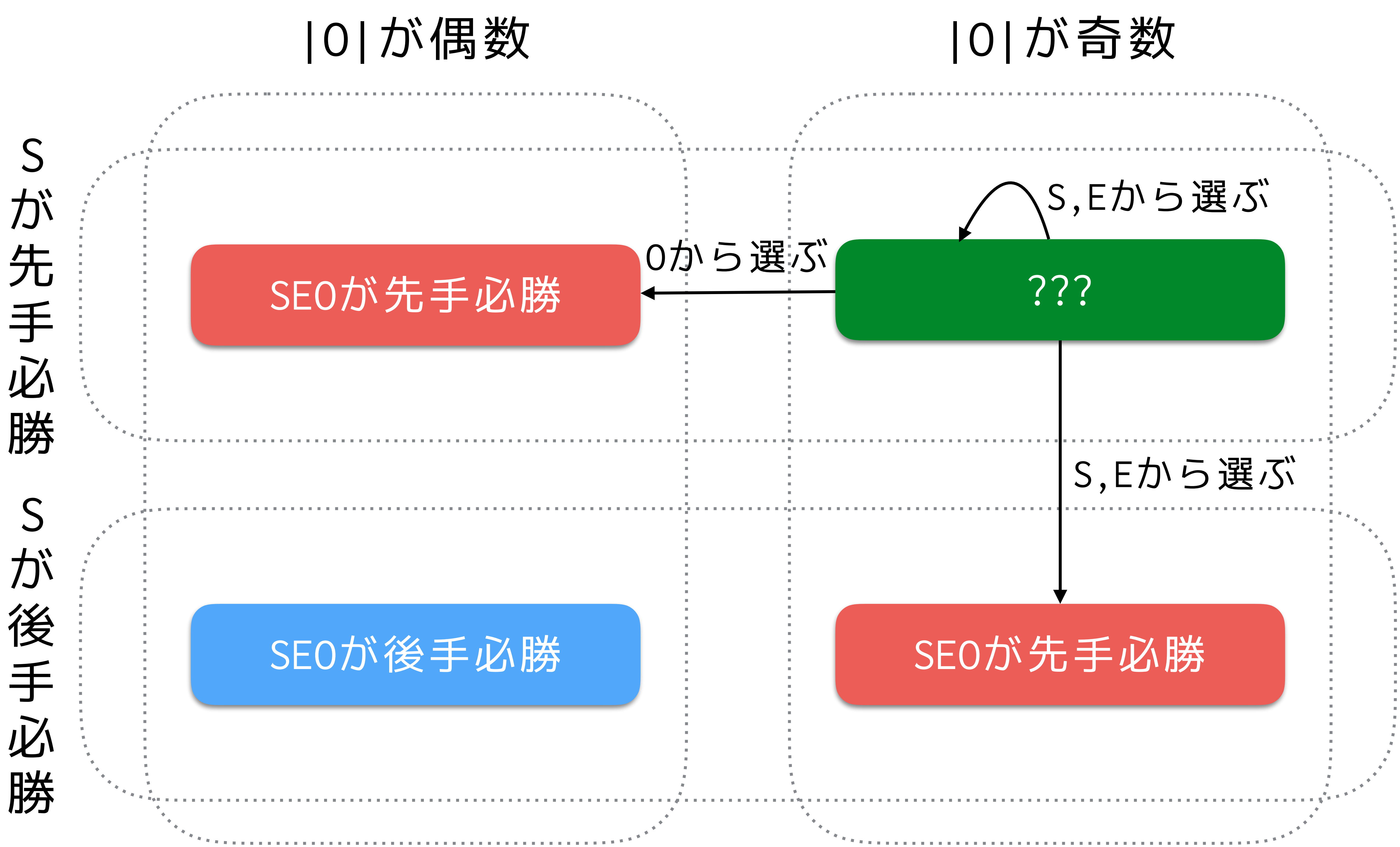
	|O|が偶数	|O|が奇数
Sが先手必勝	SEOが先手必勝	???
Sが後手必勝	SEOが後手必勝	SEOが先手必勝

S: (>0, >0), …

E: (0, 偶), …

O: (0, 奇), …

解法



S: (>0, >0), …

E: (0, 偶), …

O: (0, 奇), …

解法

- ゲームSが先手必勝で|O|が奇数の場合を考える
- その手で全てを(*, 0)にできるときを除いて
  SまたはEから選んで
  　G = Sのgrandy数のxorsum
  が G > 0 となるように選ばなければならない
- 先手ができる行動は
  - (偶, 奇)から偶を選ぶ: G <- G xor 3
  - (偶, 奇)から奇を選ぶ: G <- G xor 2
  - それ以外から選ぶ: G <- G xor 1
- EO = {(偶, 奇) ∈ S}
  P = Σmin(x, y) forall (x, y)∈S - EO とする
- Pはゲームを終了させるために
  少なくともGにxor1しなければならない回数を表している

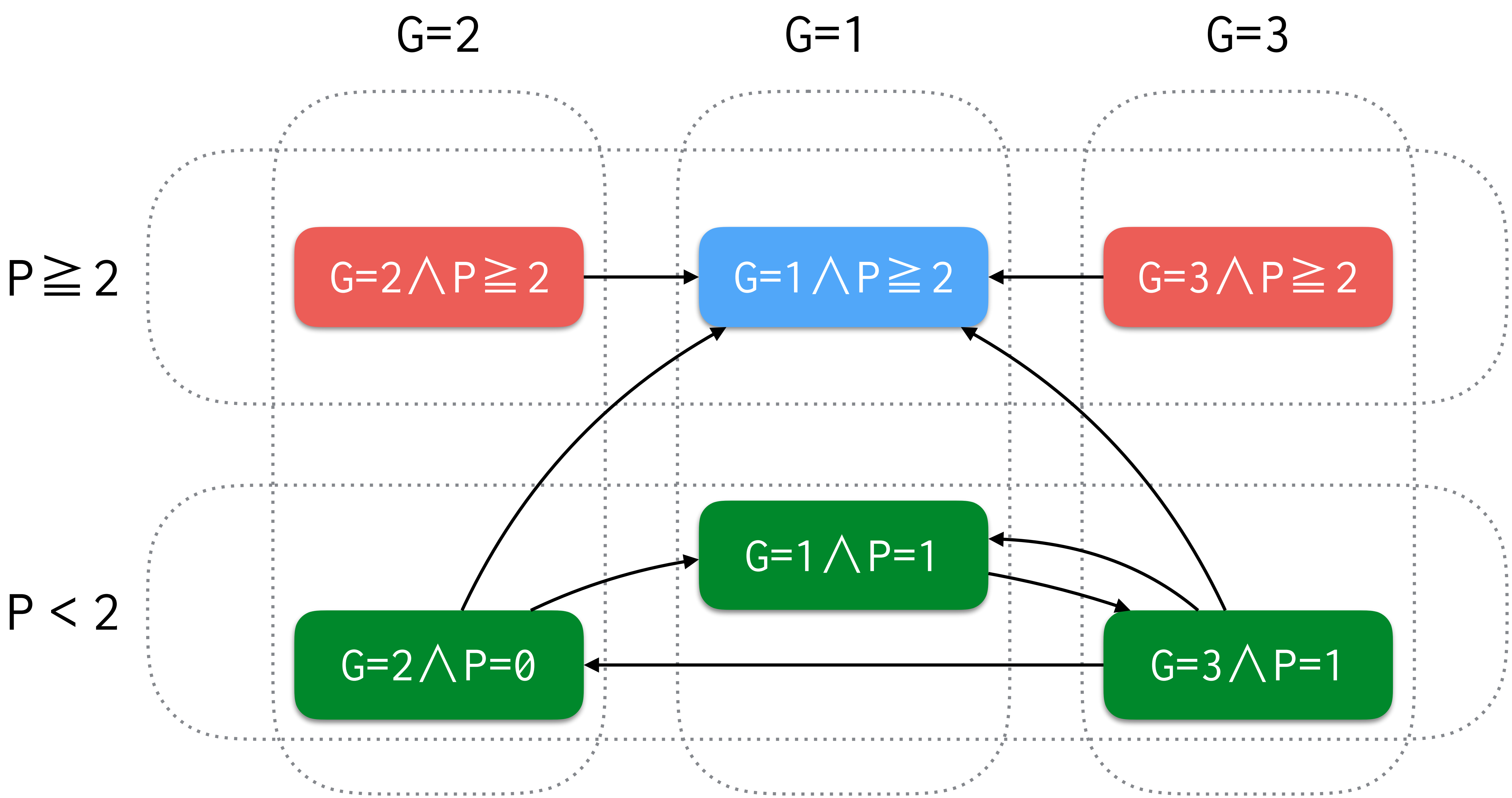
S: (>0, >0), …

E: (0, 偶), …

O: (0, 奇), …

解法

方針：以下のように分類して示していく



EO：（偶>0，奇） | P：xor1必要回数 | E：（0，偶） | O：（0，奇）

解法

- 今後、Sが先手必勝(G>0)で|O|が奇数の場合を考える
- |O|から選ぶと負けるので，今後考えないこととする
- 命題2．G=1 ∧ P≧2 ⇒ 後手必勝
  - xor2/xor3の選択肢しかなく，
    S'から選ばなければならない
  - G=1より|EO|は偶数なので
    相手はこれに合わせてEOから選んで
    G=1 ∧ P≧2 であって |EO| が2少ない状態にできる
  - 最終的に|EO|=0となり，
    ゲーム終了または選択肢がなくなる
    (xor1しなければならない)

EO:（偶>0，奇）	P: xor1必要回数	E:（0，偶）

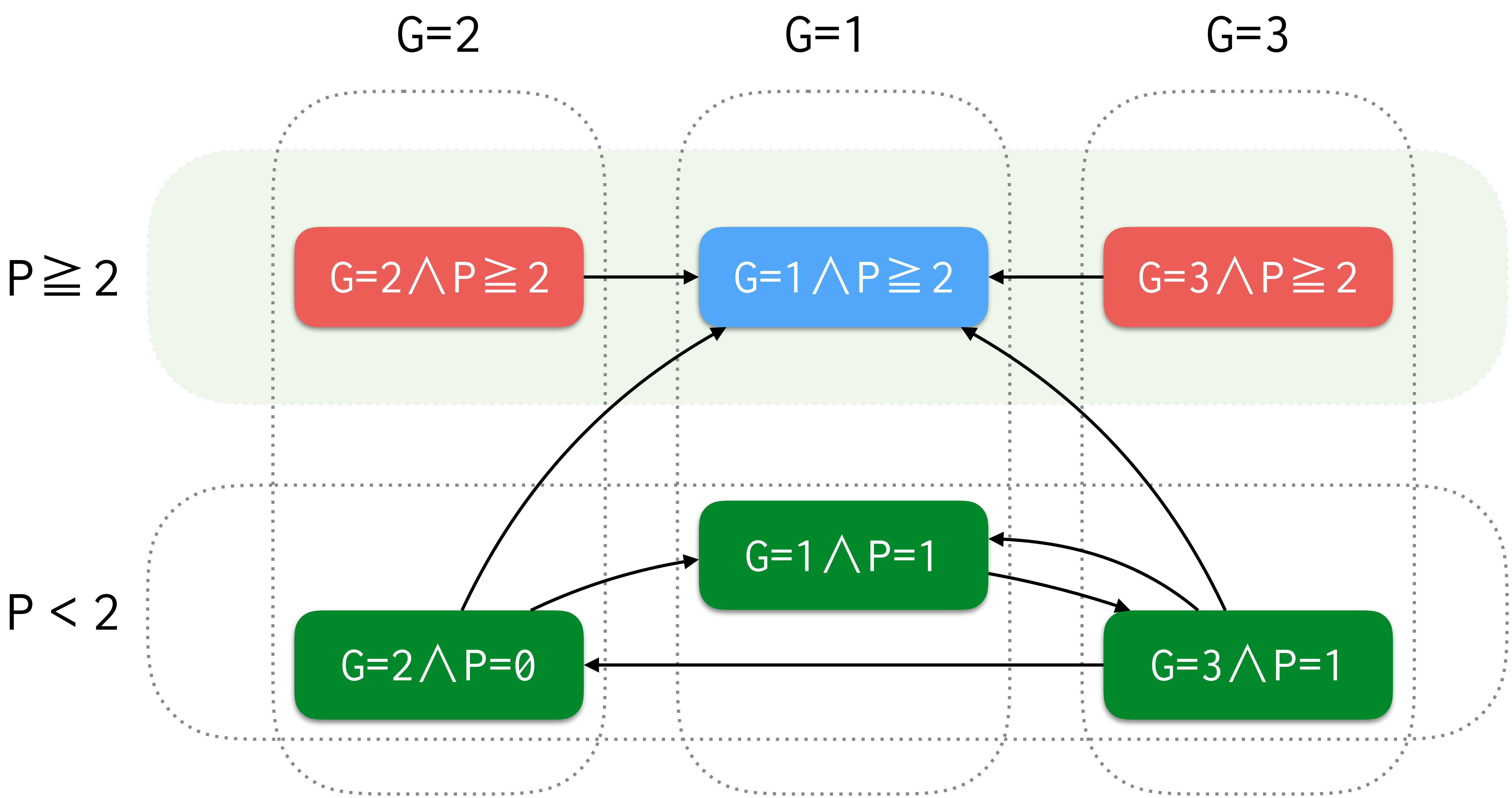
解法

- 命題2(再掲). G=1 ∧ P≧2 ⇒ 後手必勝
- 系3. G=2 ∧ P≧2 ⇒ 先手必勝
  - G=2 より |EO|>0 なのでxor3して命題2の形にできる
- 系4. G=3 ∧ P≧2 ⇒ 先手必勝
  - 同上

EO: (偶>0, 奇)	P: xor1必要回数	E: (0, 偶)

解法

上半分は示せた



EO：（偶>0，奇）　P：xor1必要回数　E：（0，偶）

解法

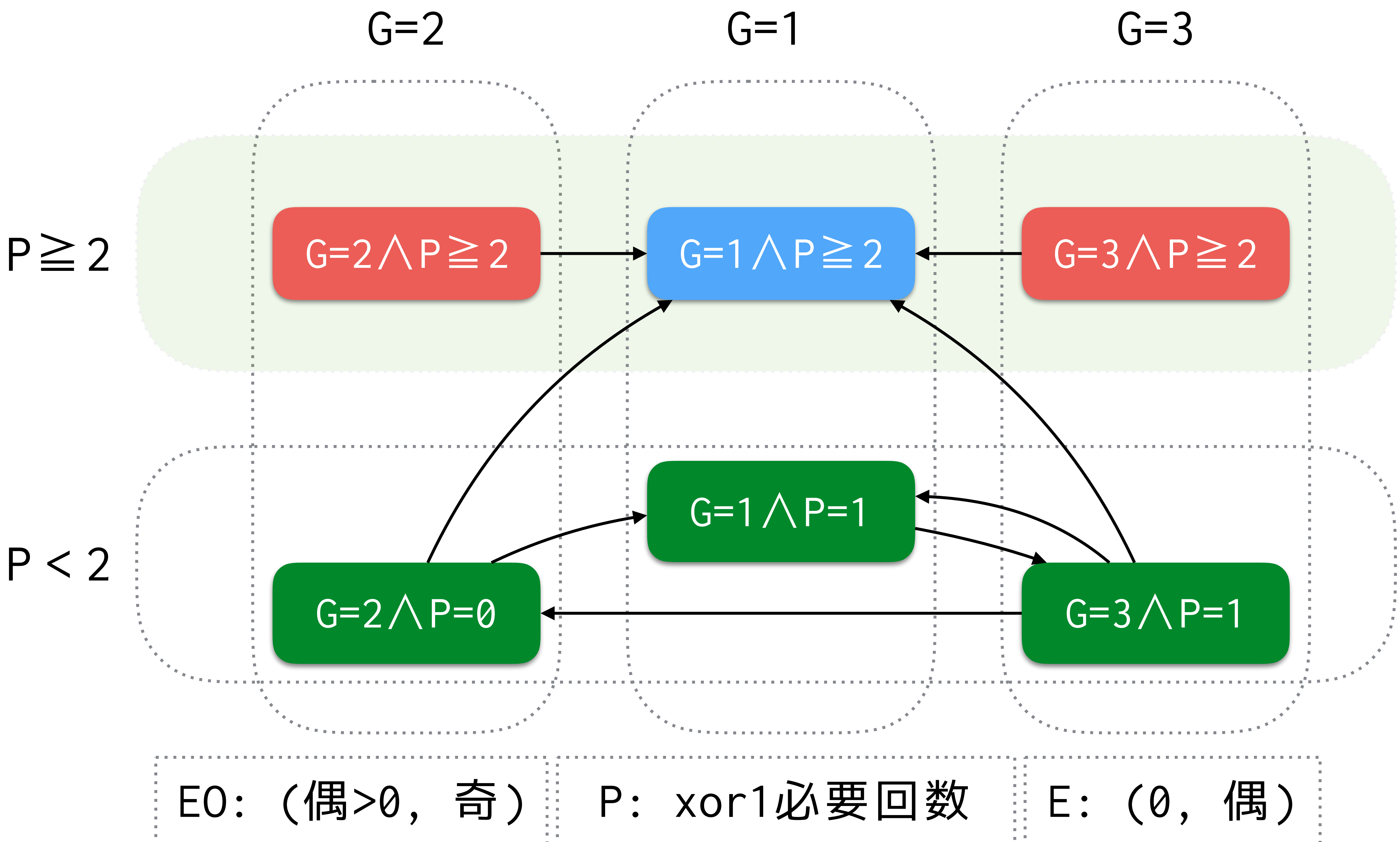
- 命題3. P＜2 ⇒ P = G mod 2
  - G mod 2 = 1 のとき
    少なくとも1つはgrandy数が1のもの:(奇, 奇)がある
  - G mod 2 = 0 のとき
    grandy数が1のもの:(奇, 奇)は0個か2個以上

- Pの定義:
  P = Σmin(x, y) forall (x, y)∈S - EO
  = Σmin(偶, 偶) + Σmin(奇, 奇)

EO: (偶>0, 奇)	P: xor1必要回数	E: (0, 偶)

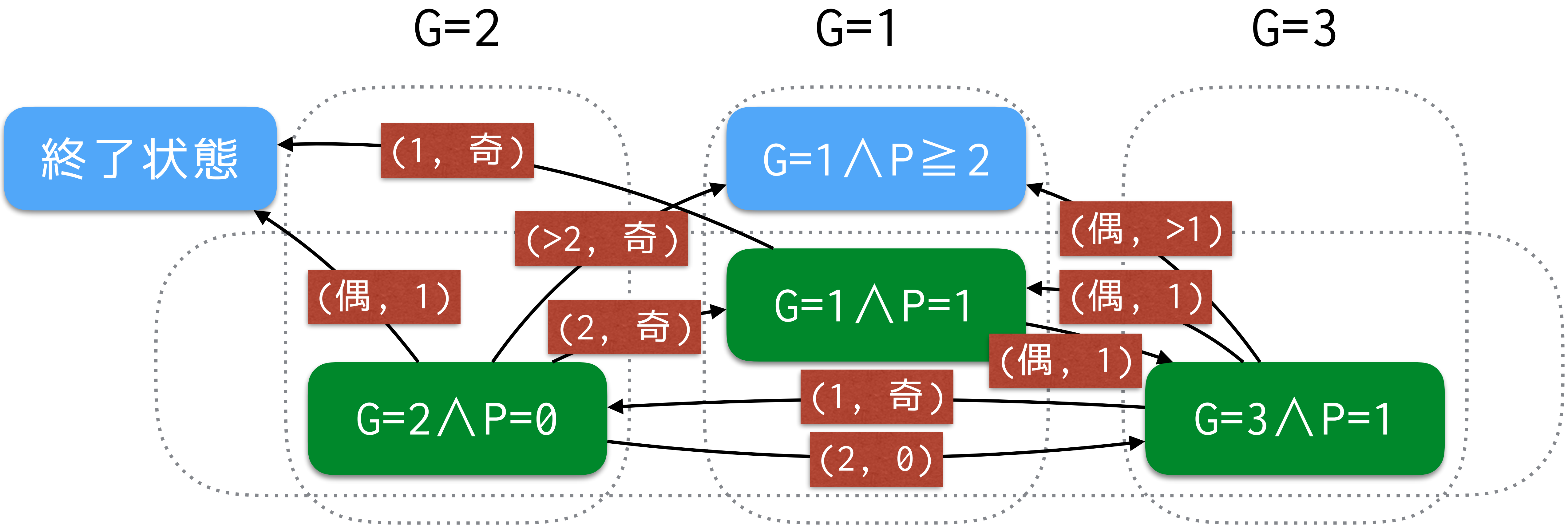
解法

下半分の状態はこの3つのみであることがわかる



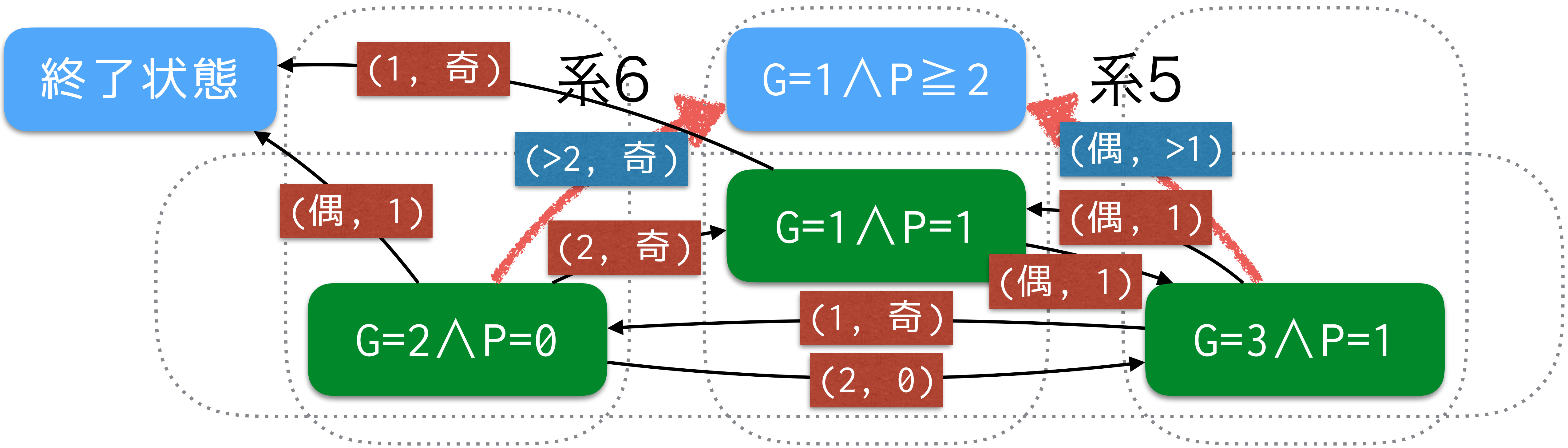
解法

下半分の状態の各要素数は以下の通りなので
勝てる見込みのある遷移もこれだけ



(奇, 0)	奇数個	奇数個	奇数個
(偶, 0)	任意	任意	任意
(偶, 奇)	奇数個	偶数個	奇数個
(奇, 奇)	なし	{(1, *)}	{(1, *)}
(偶, 偶)	なし	なし	なし

解法

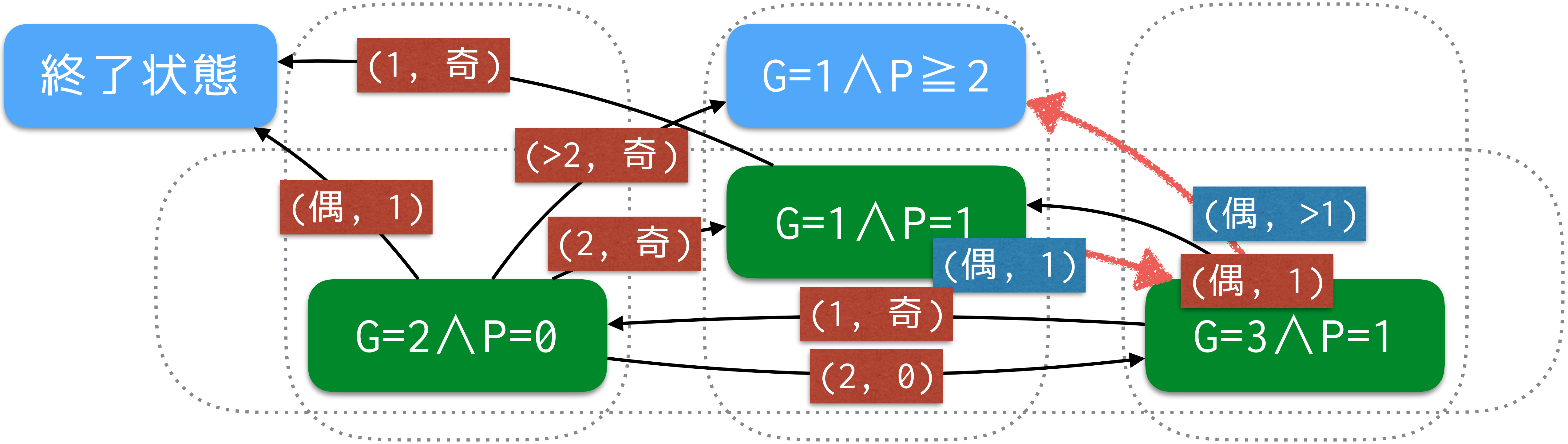


(偶，奇)	奇数個	偶数個	奇数個
(奇，奇)	なし	{(1，*)}	{(1，*)}

- 系5．G=3∧∃(偶，奇>1)∈EO ⇒ 先手必勝
  - (偶，奇>1)から奇>1を選べば G=1∧P≧2 の形になる

- 系6．G=2∧∃(偶>2，奇)∈EO ⇒ 先手必勝
  - (偶>2，奇)から偶>2を選べば G=1∧P≧2 の形になる

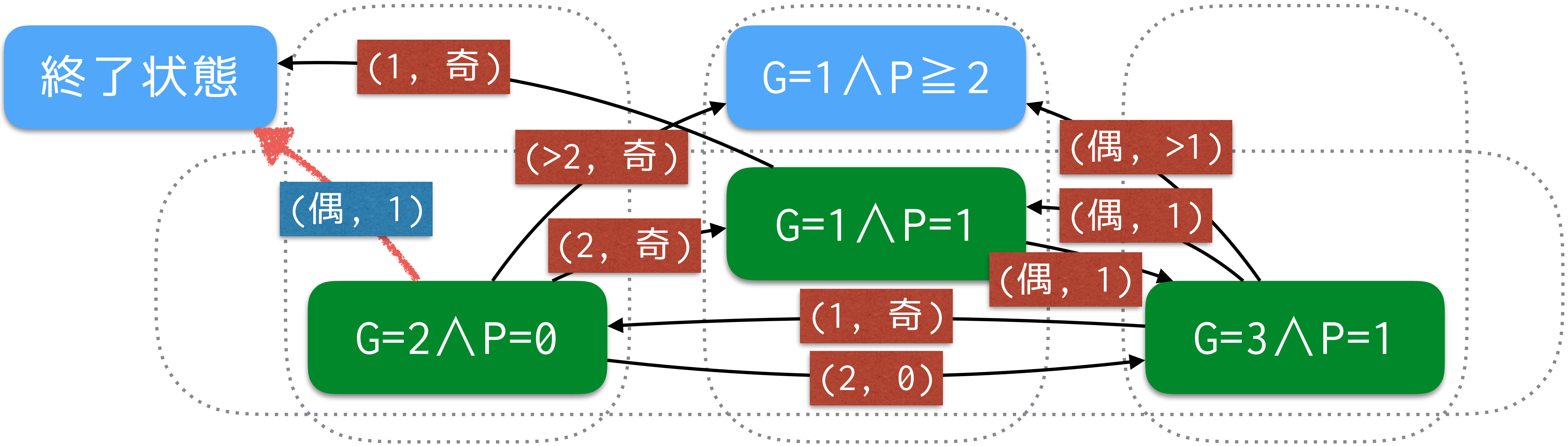
解法



(偶，奇)	奇数個	偶数個	奇数個
(奇，奇)	なし	{(1，*)}	{(1，*)}

- 系7． G=1∧P=1のとき ∃(偶，奇>1)∈E0 ⇒ 後手必勝
  証） |E0|は2以上の偶数で先手はE0から奇(=1)を選ばなければならない
  後手がE0から奇>1を選べば G=1∧P≧2

解法

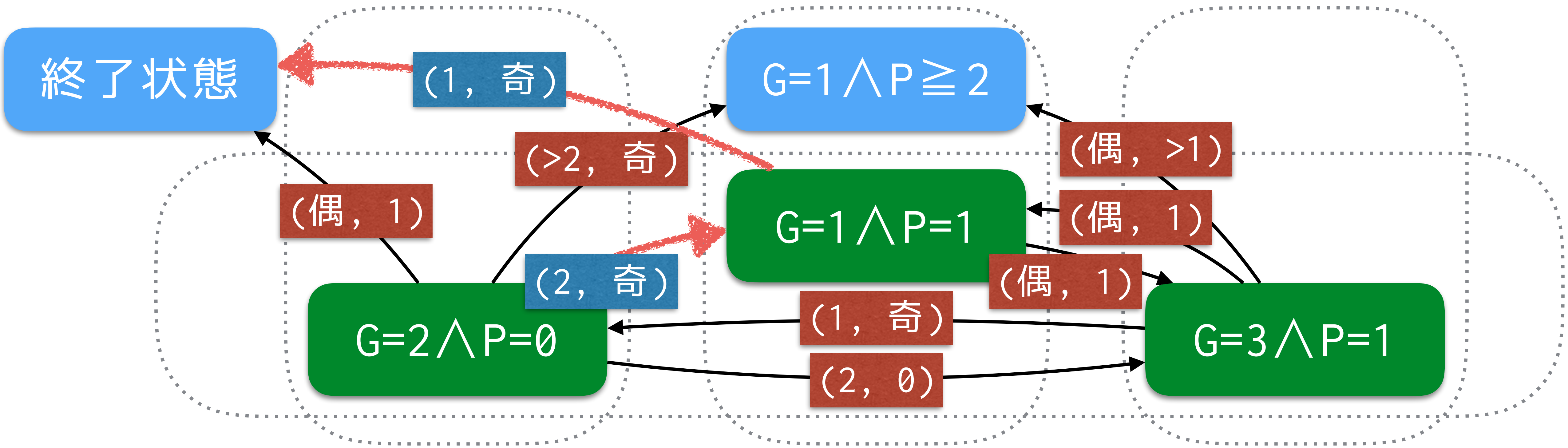


(偶, 奇)	奇数個	偶数個	奇数個
(奇, 奇)	なし	{(1, *)}	{(1, *)}

- 系8. G=2∧EO={(偶, 奇)}のとき
  先手必勝 ⇔ 奇=1

<=) 奇=1 のときそれを選べば全て(*, 0)にでき勝利

解法

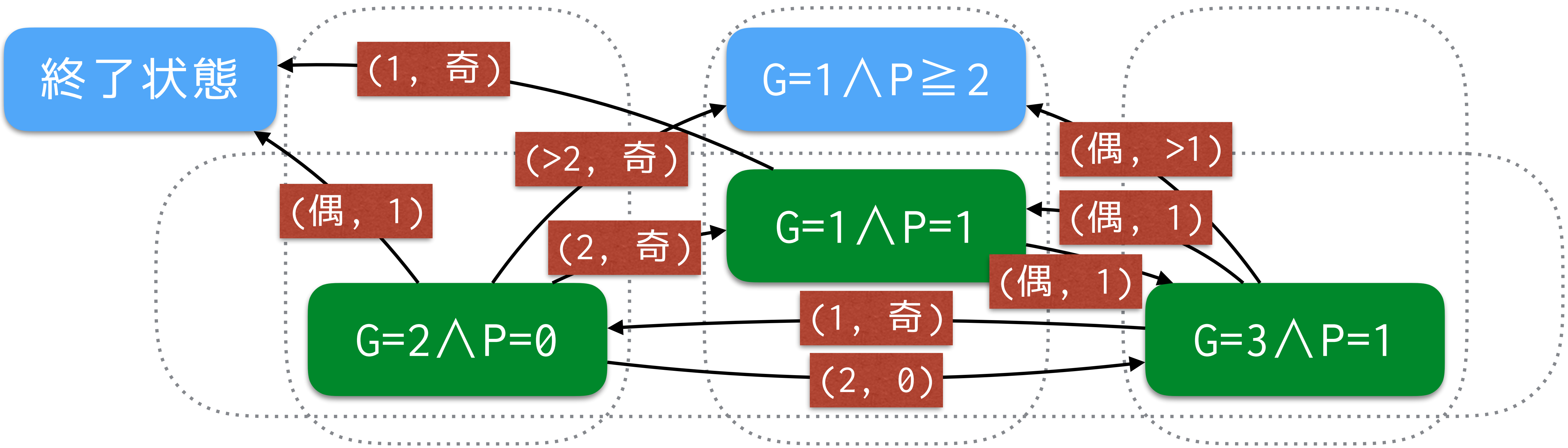


(偶，奇)	奇数個	偶数個	奇数個
(奇，奇)	なし	{(1，*)}	{(1，*)}

=>対偶）奇>1のとき先手ができる行動は次の3通りのみ

- EOから偶=2を選ぶ：(1，1)を選んでゲーム終了
- Eから(0，偶>2)を選ぶ：G=3∧P≧2
- Eから(0，2)を選ぶ：(1，1)を選ぶと同じ形

解法

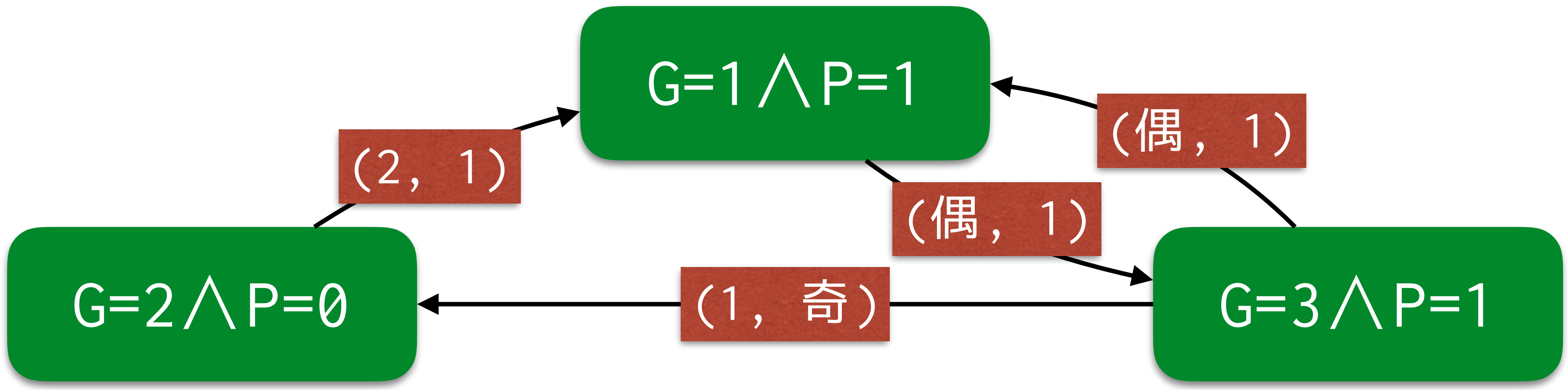


（偶，奇）	奇数個	偶数個	奇数個
（奇，奇）	なし	{(1，*)}	{(1，*)}

（偶，奇>1）があるケースは全て考えたので

今後EOは（偶，1）のみとする

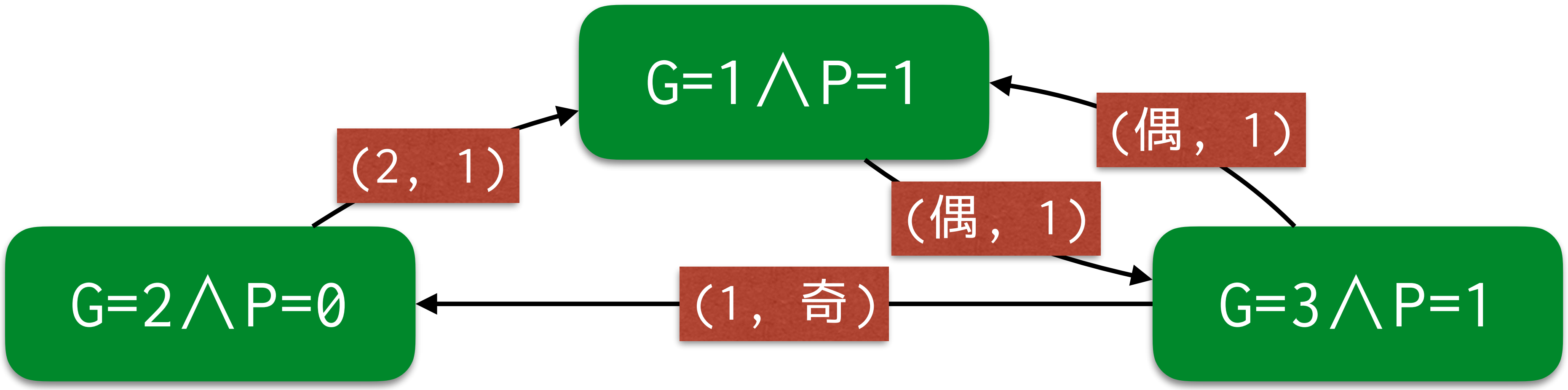
解法



（偶，奇）	奇数個	偶数個	奇数個
（奇，奇）	なし	{（1，*）}	{（1，*）}

無駄な遷移や即座に勝利できる遷移を省略

解法



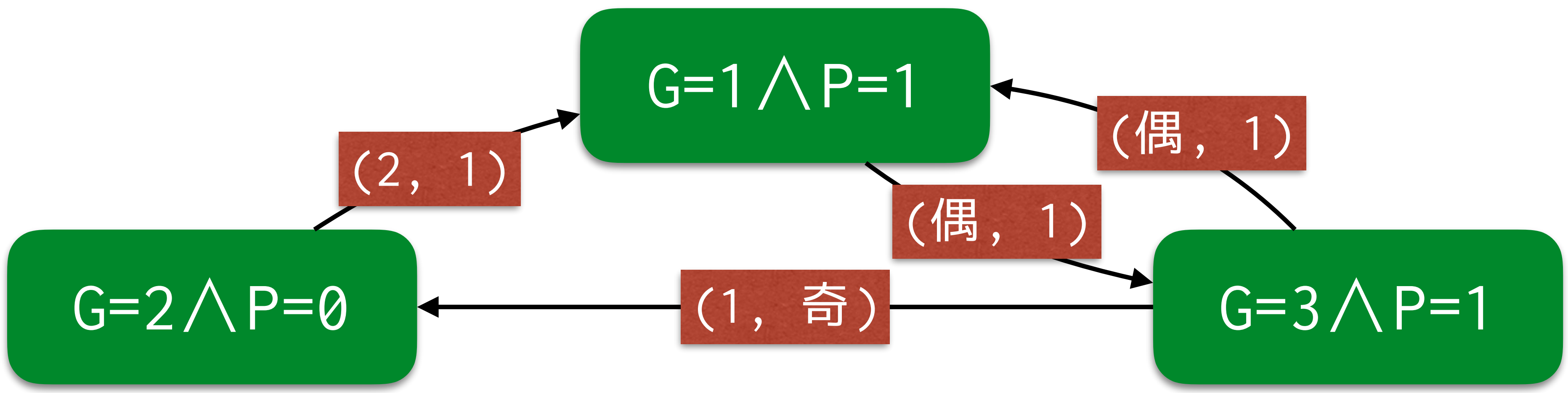
（偶，奇）	奇数個	偶数個	奇数個
（奇，奇）	なし	{(1, *)}	{(1, *)}

・ 命題4. G=3 ∧ ∀(e, o)∈EO [(e, o) = (2, 1)] のとき
先手必勝 ⇔ |EO|≧3

=>対偶) |EO|は奇数なので|EO|=1

(2, 1)を選ぶと(1, 奇)で敗北
(1, 奇)を選ぶと(2, 1)で敗北

解法

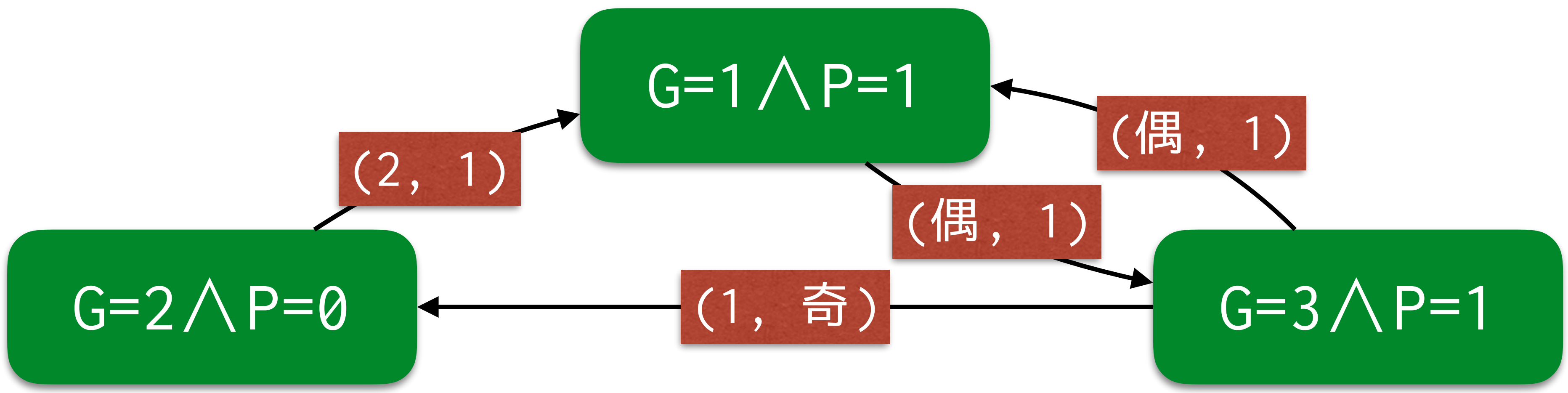


(偶，奇)	奇数個	偶数個	奇数個
(奇，奇)	なし	{(1，*)}	{(1，*)}

・ 補題1．G=3 ∧ ∀(e，o)∈EO [(e，o) = (2，1)] のとき
先手必勝 ⇔ |EO|≧3

<=) まず|EO|=3のとき
(1，奇)を選べば(2，1)→(2，1)となり，
このときG=3∧EO={(2，1)}なので先手勝ち
|EO|≧3のとき，|EO|は奇数なので
(2，1)→(2，1)を繰り返して|EO|=3にできる

解法

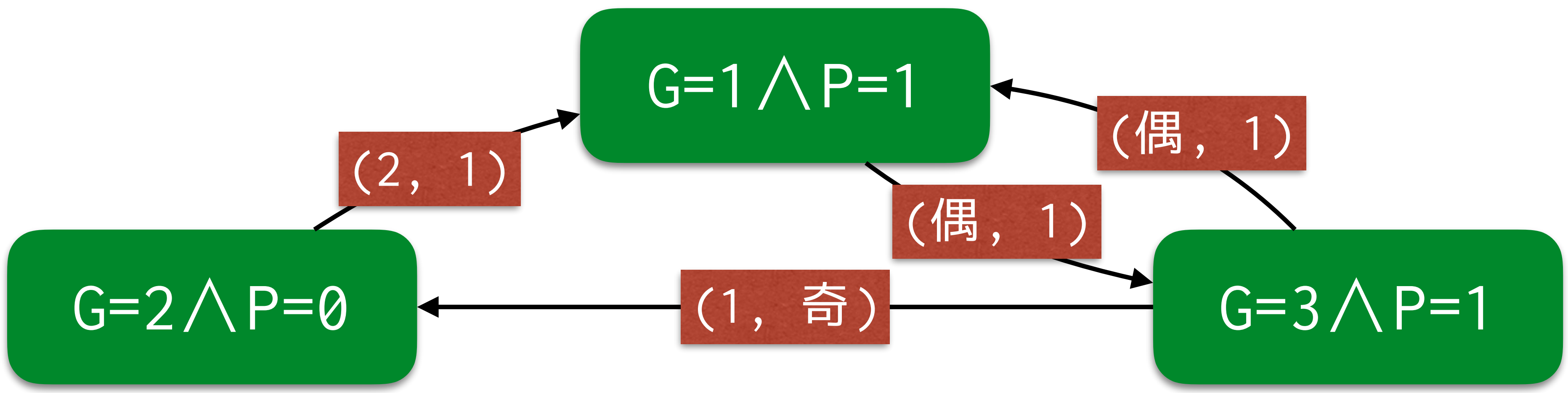


(偶，奇)	奇数個	偶数個	奇数個
(奇，奇)	なし	{(1, *)}	{(1, *)}

- Large =(偶>2, 1)の数, Small = (2, 1)の数 とする
- 命題4. G=3∧P=1∧∀(偶, 1) のとき
  先手必勝 ⇔ Small ≧ Large+3

証) Large>0である間(偶, 1)→(偶, 1)と選び続ける
補題1よりLarge=0となったときのSmallを最大化したい
(逆に後手は最小化したい)
従って(偶>2, 1)→(2, 1)と選び続ける

解法



(偶，奇)	奇数個	偶数個	奇数個
(奇，奇)	なし	{(1, *)}	{(1, *)}

- G=1∧P=1∧∀(2, 1)のときや G=2∧P=0∧∀(偶, 1)のときの結果は命題4からただちに得られる
- SmallとLargeの動きをシミュレートしても良い

お詫び

部分点の設定に不備があり、申し訳ございませんでした。