PRG-w4a: ゲーム木

脇田建

2019.10.18

ゲーム木 (Game Tree)

ゲーム木とは

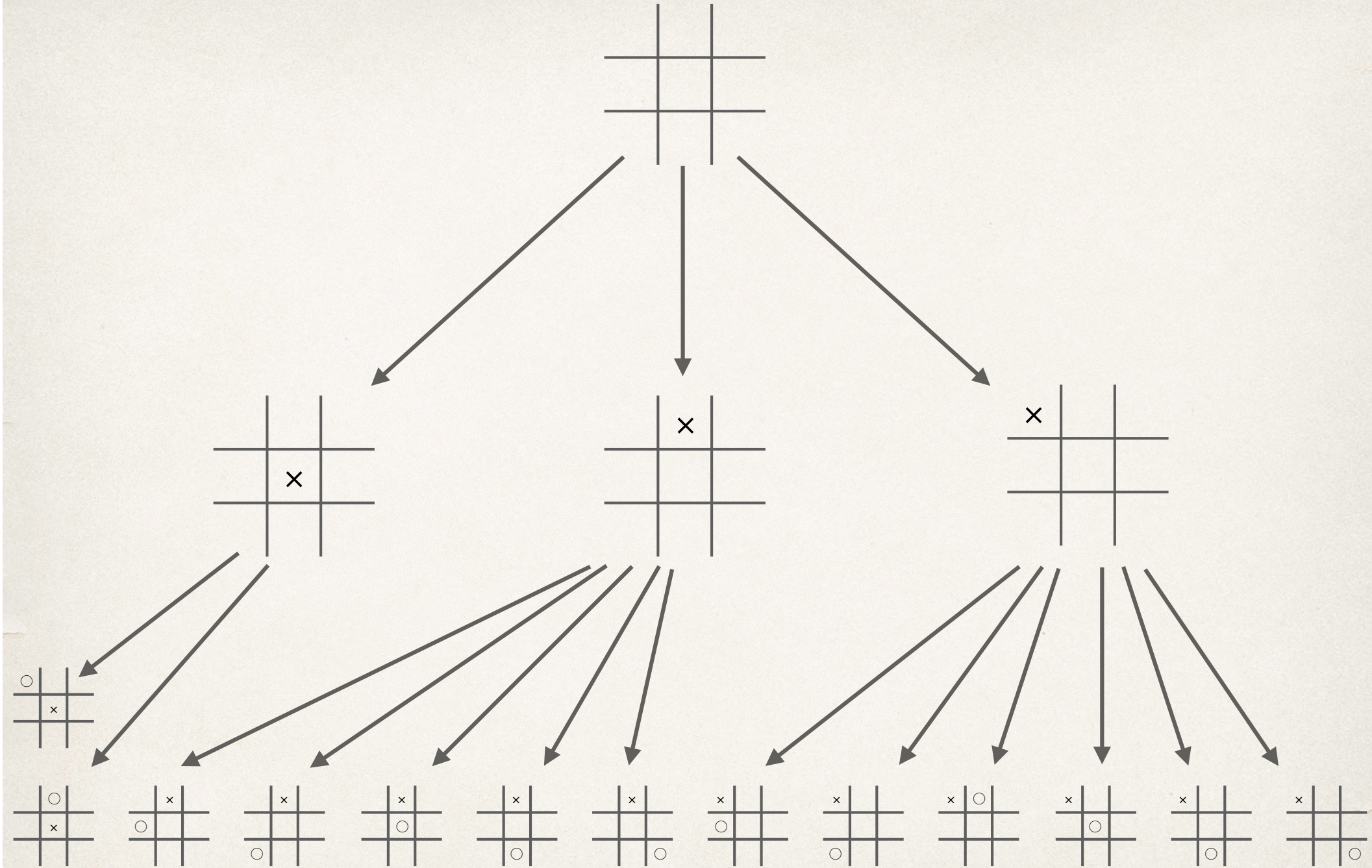
- 有向グラフ
  - 頂点：あるゲーム進行のある時点の状況
  - 辺：ゲームの状況の遷移

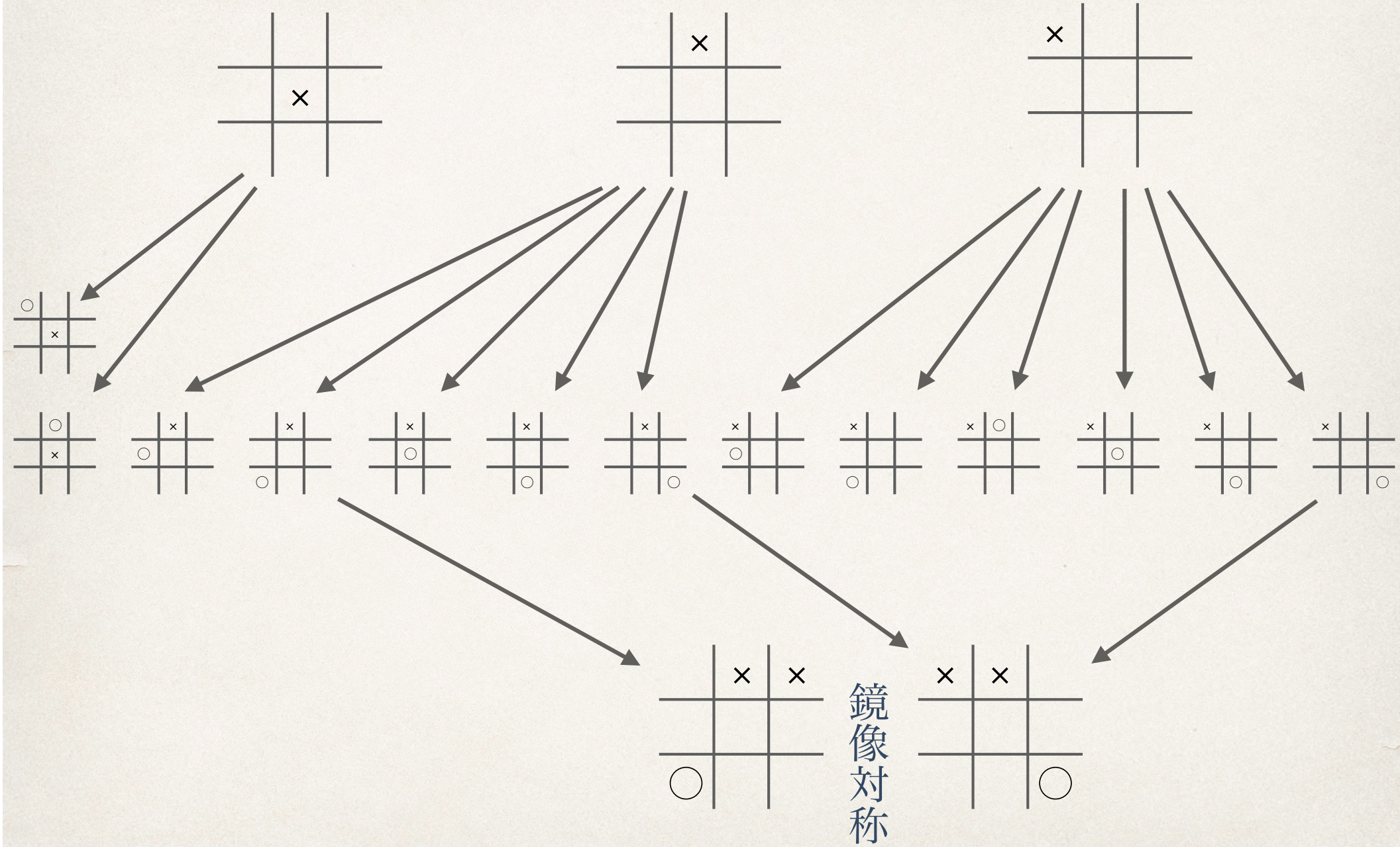
ゲーム木とは

- 有向グラフ ← なぜ木と言わない？

  Directed Acyclic Graph（有向非巡回グラフ）

  - 頂点：あるゲーム進行のある時点の状況
  - 辺：ゲームの状況の遷移




鏡像対称

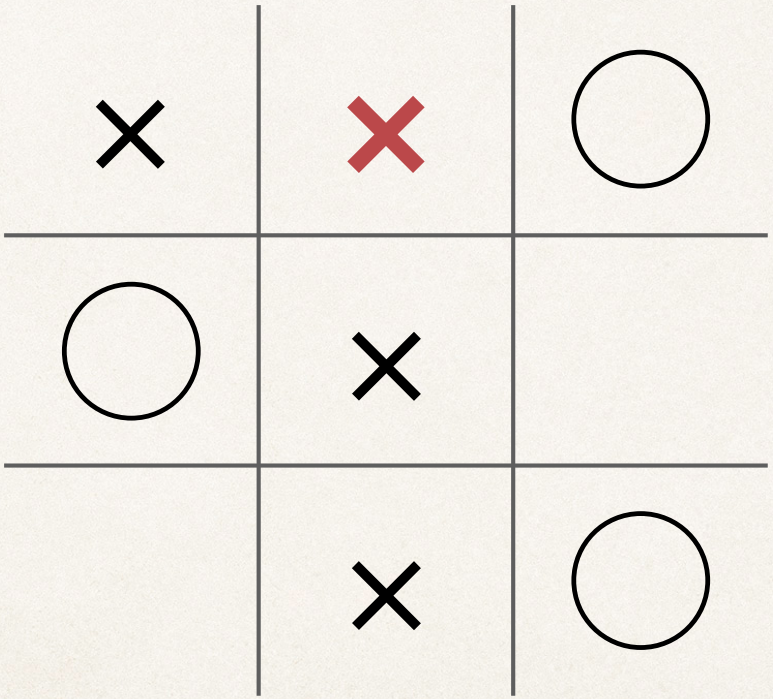
完全ゲーム木

- ゲーム木のうち、
- 最初の状況から始めて、
- 有効な遷移がすべて尽くされているもの

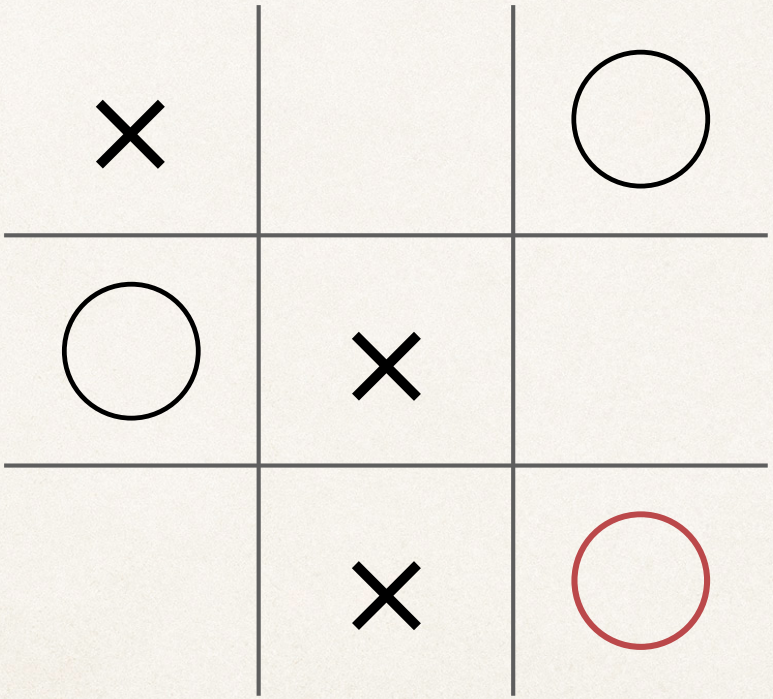
完全ゲーム木の葉

- 葉：完全ゲーム木の末端頂点
- ◯×の完全ゲーム木の葉の数は255,168個あるらしい。
  （鏡像、回転合同については問わない場合）
- 末端の頂点数がゲームの科学においてゲームの複雑性の評価値として扱われる。

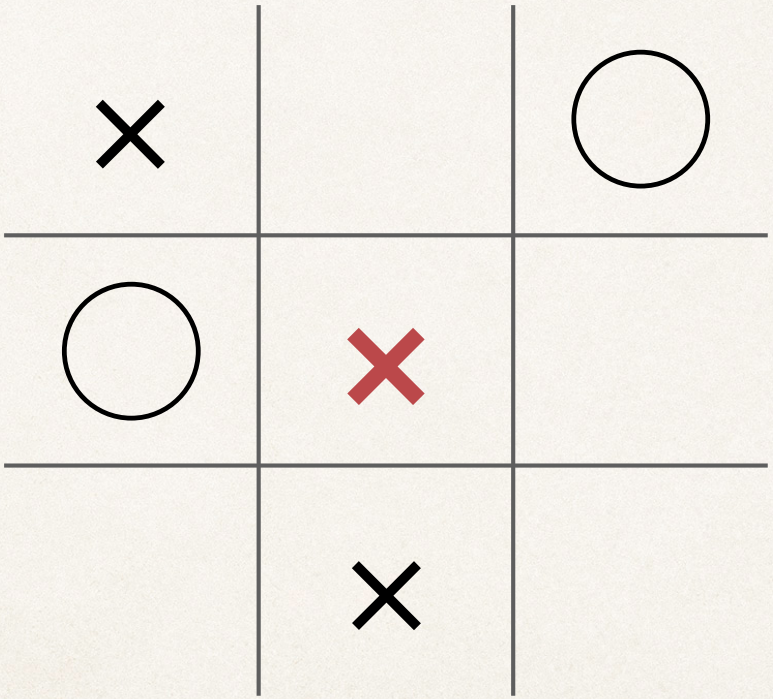
どちらの勝ち？



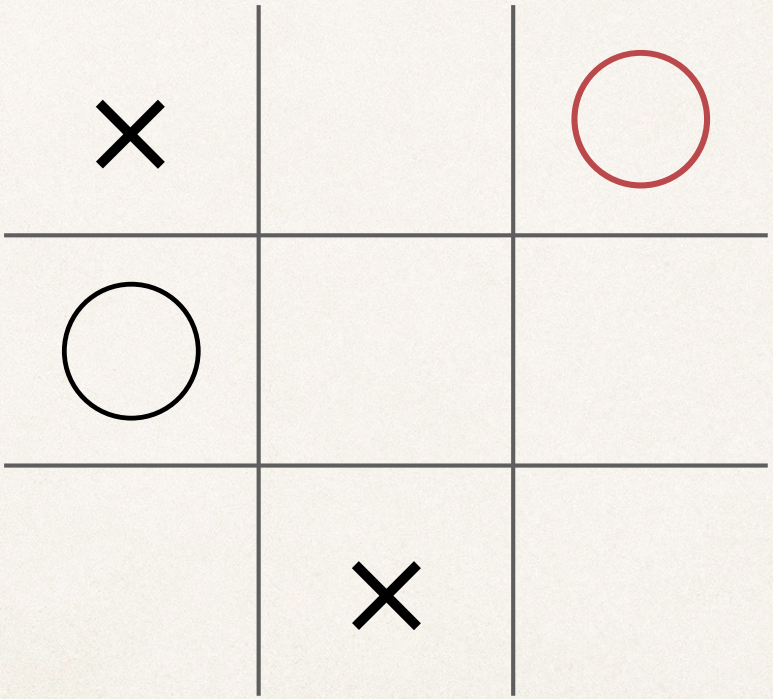
どちらの勝ち？



どちらの勝ち



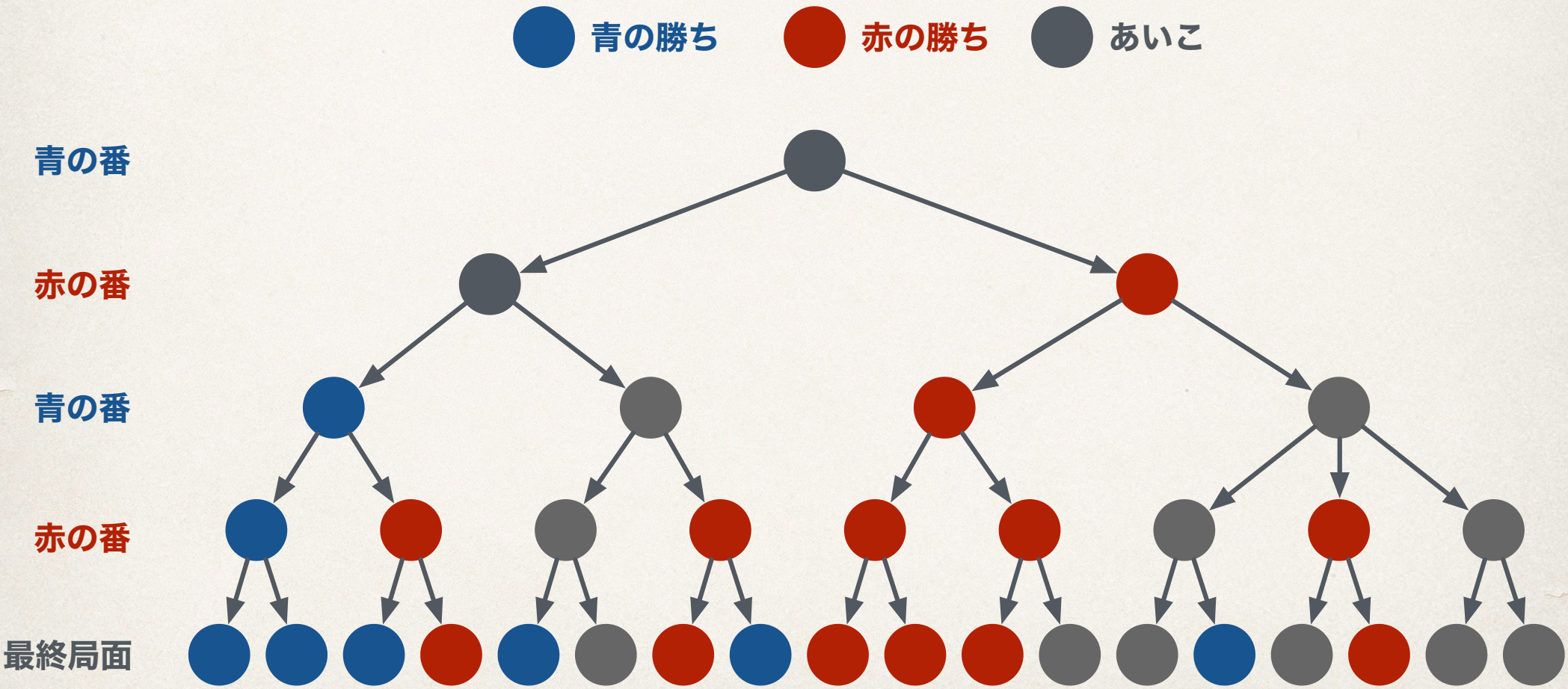
どちらの勝ち



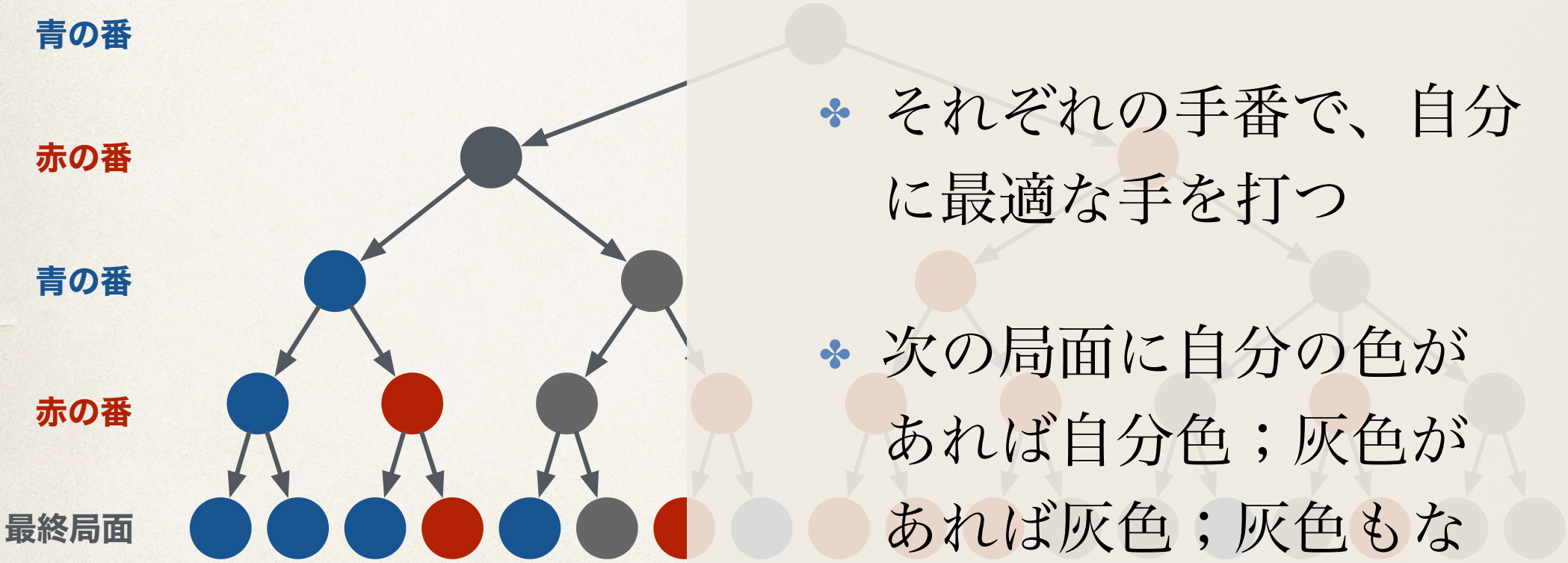
先手が勝つとは

- 先手の選択 {a1, a2, …}
- 後手の選択
  - a1 → {b11, b12, …}
  - a2 → {b21, b22, …}
  - …
- 後手のいずれの選択についても、先手が勝てる

ある完全ゲーム木における
ゲームの解



ある完全ゲーム木における
ゲームの解



- ボトムアップ
- それぞれの手番で、自分に最適な手を打つ
- 次の局面に自分の色があれば自分色；灰色があれば灰色；灰色もなければ相手の色

Minimax戦略

- （○×とは異なり）一般のゲームでは、完全ゲーム木を構成することはできない。そこで、手筋を数段だけ先読みした部分ゲーム木にもとづいて戦略を立てる。
- Minimax: 最悪時に想定される被害の最小化
- （1997年にチェスのグランドマスターを**IBM Deep Blue**が破ったときは12手の先読みをしていたらしい）

Minimaxアルゴリズム (1/3)

- 二人が交互に手を打つゲームの場合
- 評価関数: v(ゲームの状況)
  - 自分にとって有利・不利 → v が大きい・小さい
- 自分は v が大きくなる状況を導きたいが、相手は v が小さい状況を導きたい。

Minimaxアルゴリズム (2/3)

- def minimax(頂点, 深さ, プレーヤ): Double = {

```
    if (深さ == 0) v(頂点)
    プレーヤ match {
        case 自分 => { 評価関数を最大化しようとする }
        case 相手 => { 評価関数を最小化しようとする }
    }
}
```

Minimaxアルゴリズム(3/3)

- 自分にとって最高な状況 = 被害を最小化する手を選択
  - 頂点.子頂点たち.foldLeft(-∞).((v, 子頂点) =>
    max(v, minimax(子頂点, 深さ-1, 相手))
- 自分にとって最悪な状況 = 被害を最大化する手を選択
  - 頂点.子頂点たち.foldLeft(∞).((v, 小頂点) =>
    min(v, minimax(小頂点, 深さ-1, 自分))

Minmaxアルゴリズムの鍵は

- 評価関数
- Campbell 2002, Deep Blue,
  Artificial Intelligence 134(1-2), pp. 57-83,
  doi: 10.1016/S0004-3702(01)00129-1
- 7節: Deep Blue が採用した評価関数
- 4節、5節: 検索戦略（1997年のときよりも先読みが強化された）
- 6節: 並列実行手法

東工大オセロを目指せ！