UNIVERSITAT POLITÈCNICA DE VALÈNCIA

# Búsqueda con adversario: algoritmo minimax y poda alfa-beta [1]

Alfons Juan
Albert Sanchis
Jorge Civera

DSIC
Departamento de Sistemas Informáticos y Computación

[1] Para una correcta visualización, se requiere Acrobat Reader v. 7.0 o superior

# Objetivos formativos

- Conocer la búsqueda con adversario básica.
- Aplicar el algoritmo ***minimax*** y poda ***alfa-beta***.

# Índice

# 1. Búsqueda con adversario

La búsqueda con adversario consiste en elegir jugada en juegos:

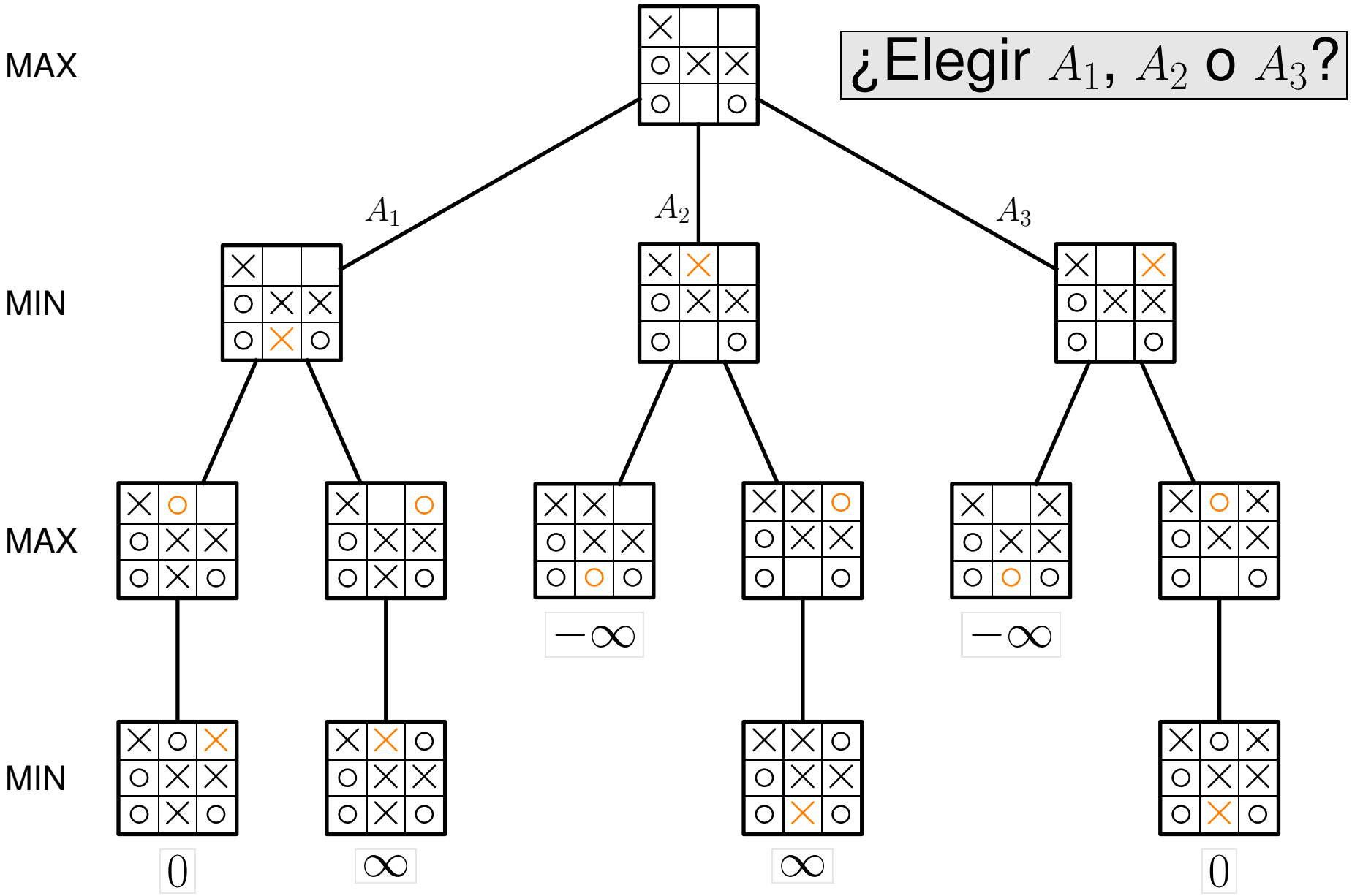
- ***deterministas*** i.e. la suerte no interviene
- ***de 2 jugadores*** MAX (el sistema) y MIN (el adversario)
- ***por turnos*** empieza MAX y tiene que elegir jugada
- ***info. perfecta*** sabemos estados y reglas del juego (ej. ajedrez)
- ***suma zero*** utilidades MAX/MIN al final del juego opuestas

Elementos básicos:

- ***Estado inicial*** $s_0$***:*** desde donde MAX tiene que elegir jugada.
- ***Acciones(***$s$***):*** jugadas legales desde el estado $s$.
- ***Terminal(***$s$***):*** indica si $s$ es estado terminal del juego o no.
- ***Utilidad(***$s$***):*** utilidad para MAX del estado terminal $s$.

***Objetivo: elegir jugada (que lleve a un estado) de máxima utilidad***

# Ejemplo: elegir jugada en el tres en raya



# 2. Algoritmo minimax y poda alfa-beta

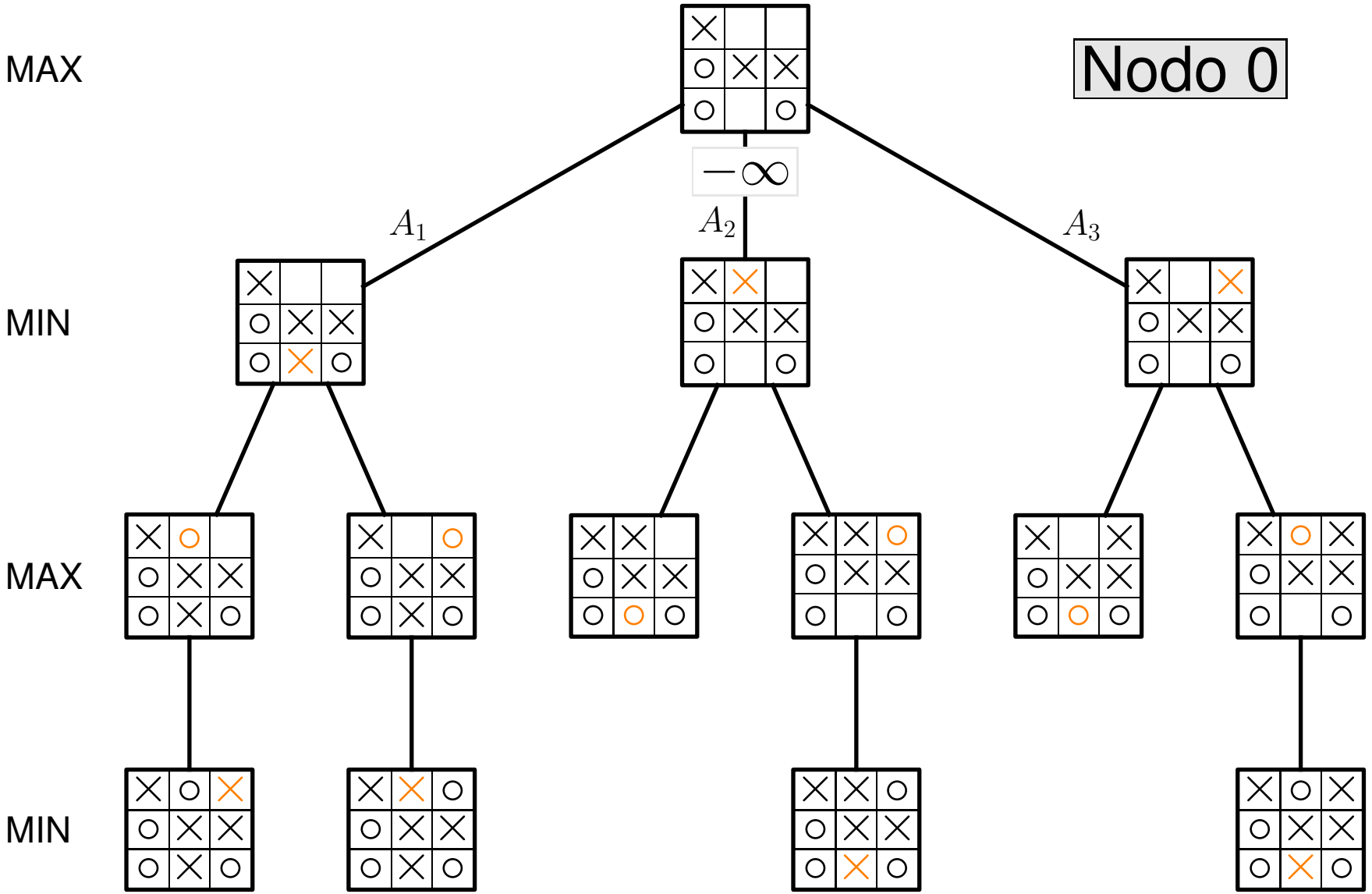
Valor, decisión y algoritmo minimax:

- ***Valor minimax*** de un estado/nodo: utilidad (MAX) del nodo terminal al cual llegamos si ambos jugadores juegan óptimamente
- ***Decisión minimax:*** elegir la jugada de mayor valor minimax
- ***Algoritmo minimax:*** cálculo de la decisión minimax mediante búsqueda con adversario por profundidad (limitada)

## *Algoritmo minimax básico*

**mm(**$n$**,** $p$**,** $max$**)** // nodo, profundidad, $max$="¿juega max?"
  **si** $n$ es terminal **devuelve** utilidad de $n$
  **si** $p = 0$ **devuelve** valor heurístico de $n$
  *// si* $max$ *devuelve el máximo de valores minimax de los hijos*
  **si** $max\ v = -\infty$; $\forall\ s \in$ succ$(n)$: $v = \text{máx}(v,$ mm$(s, p-1,$ FALSE$))$
  *// si no devuelve el mínimo de valores minimax de los hijos*
  **si no** $v = \infty$; $\forall\ s \in$ succ$(n)$: $v = \text{mín}(v,$ mm$(s, p-1,$ TRUE$))$
  **devuelve** $v$

# Ejemplo resuelto con minimax

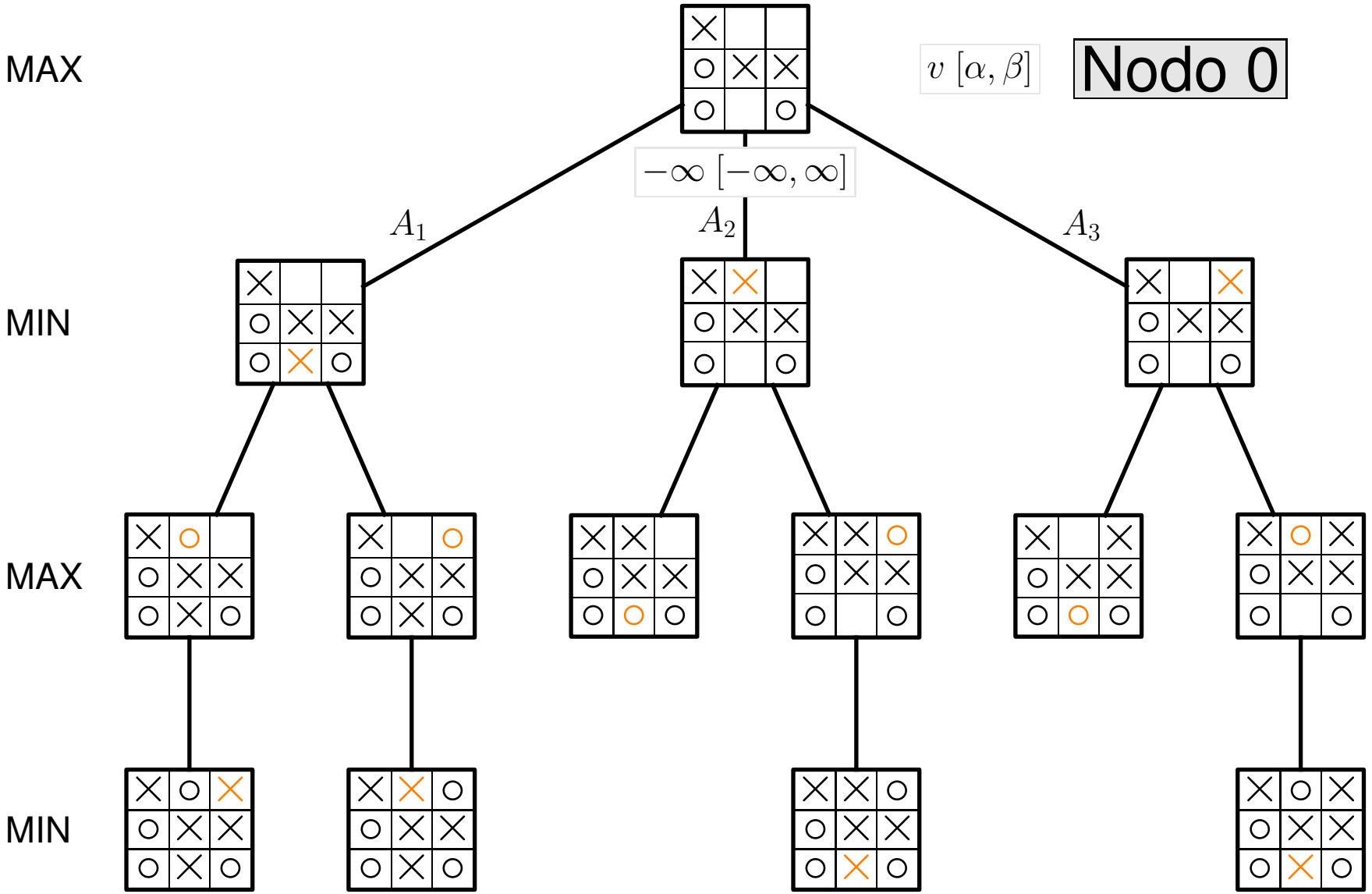


# Algoritmo minimax y poda alfa-beta

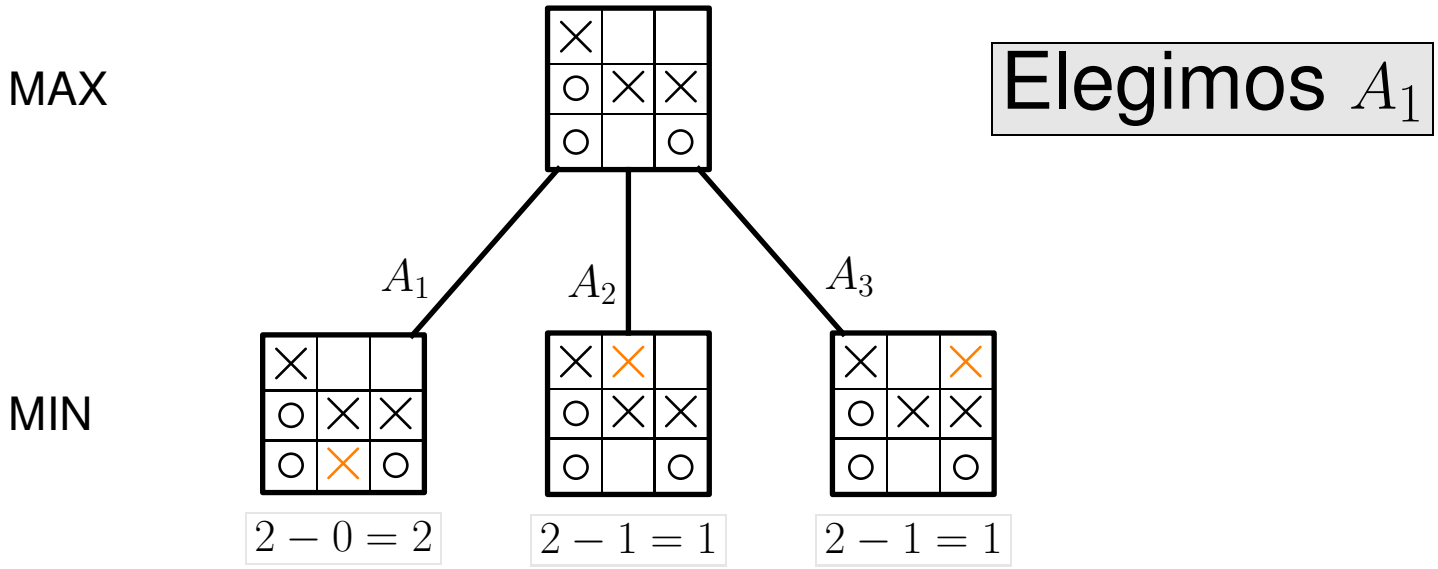
**mm**($n$, $p$, $max$) // nodo, profundidad, $max$="¿juega max?"
  **si** $n$ es terminal **devuelve** utilidad de $n$
  **si** $p = 0$ **devuelve** valor heurístico de $n$
  **si** $max$ $v = -\infty$; $\forall\, s \in$ succ$(n)$: $v = \text{máx}(v, \text{mm}(s, p-1, \text{FALSE}))$
  **si no** $v = \infty$; $\forall\, s \in$ succ$(n)$: $v = \text{mín}(v, \text{mm}(s, p-1, \text{TRUE}))$
  **devuelve** $v$

$\alpha$-$\beta$($n$, $p$, $\alpha$, $\beta$, $max$)
  **si** $n$ es terminal **devuelve** utilidad de $n$
  **si** $p = 0$ **devuelve** valor heurístico de $n$
  **si** $max$ $v = -\infty$
    $\forall\, s \in$ succ$(n)$
      $v = \text{máx}(v, \alpha\text{-}\beta(s, p-1, \alpha, \beta, \text{FALSE}))$
      $\alpha = \text{máx}(\alpha, v)$; **si** $\beta \leq \alpha$: **break** // corte $\beta$
  **si no** $v = \infty$
    $\forall\, s \in$ succ$(n)$
      $v = \text{mín}(v, \alpha\text{-}\beta(s, p-1, \alpha, \beta, \text{TRUE}))$
      $\beta = \text{mín}(\beta, v)$; **si** $\beta \leq \alpha$: **break** // corte $\alpha$
  **devuelve** $v$

# Ejemplo resuelto con poda alfa-beta



# Ejemplo resuelto con $p = 1$ y heurística




Elegimos $A_1$

Función heurística:

$$h(n) = \text{abiertas(MAX)} - \text{abiertas(MIN)}$$

donde

abiertas($j$)="# de filas, columnas y diagonales abiertas para $j$"

# Conclusiones

- Hemos visto en que consiste la búsqueda con adversario.
- Hemos aplicado el algoritmo ***minimax*** y poda ***alfa-beta***.
- Consultad [1, Cap. 5] para más detalles.

# Referencias

[1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.