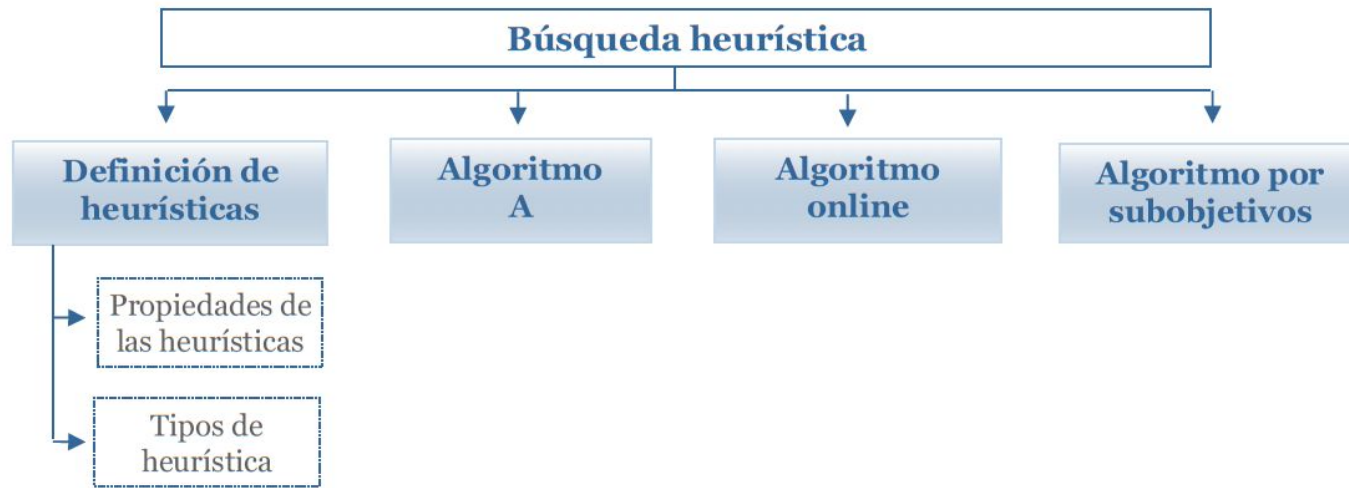


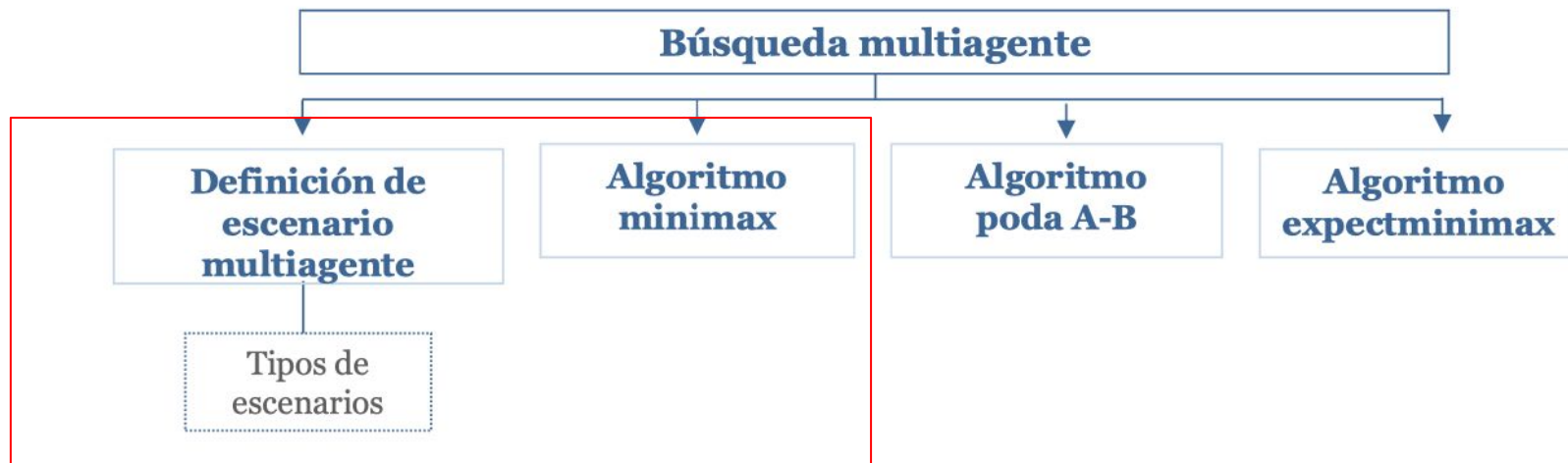
Tema 6: Búsqueda entre adversarios

En la clase pasada...



Índice

- ▶ Búsqueda entre adversarios (dos sesiones)
 - Introducción
 - El algoritmo de búsqueda minimax
 - La poda alfa-beta
 - El algoritmo de búsqueda expect-minimax



Problemas multiagente

“Aquellos en los que más de un agente especializado actúa de modo concurrente en un mismo entorno”.

→ Los agentes del entorno no pueden controlar las acciones que el resto va a realizar

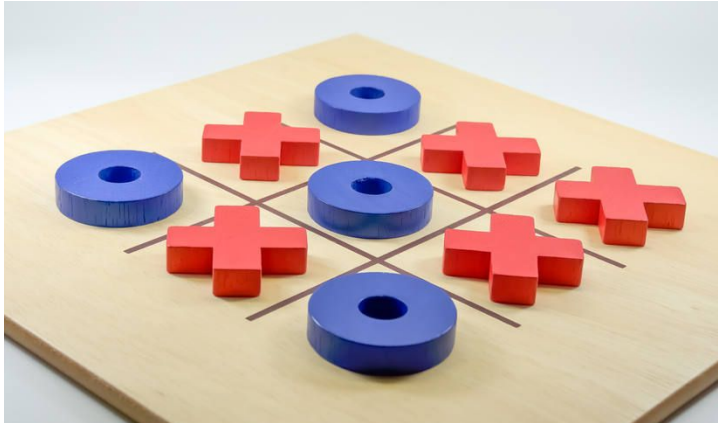
→ Sí que pueden intentar predecir el comportamiento del resto de agentes

Problemas multiagente

Distintos tipos de problemas entre varios agentes, según su comportamiento:

- **Escenarios cooperativos (tightly-coupled)** : los agentes tienen metas compartidas y, por tanto, las acciones de unos deberían facilitar los objetivos de todos los agentes.
- **Escenarios parcialmente cooperativos:** algunas metas son compartidas por varios agentes, pero otras pueden ser opuestas o independientes.
- **Escenarios colaborativos (loosely-coupled):** las metas de los agentes son independientes entre ellas pero resuelven un problema común
- **Escenarios antagónicos:** las metas de todos los agentes son opuestas.
 - El ejemplo «clásico» de escenarios antagónicos es el juego de la suma nula, donde un (grupo de) jugador(es) solo puede ganar algo a coste de otro (grupo de) jugador(es)

Juego de tablero



Vamos a asumir ciertas cosas:

- Entorno único
- Cada agente controla la ejecución de sus acciones
- Las acciones influyen en la decisión de los otros agentes
- Un agente “puede predecir” las acciones de los demás

Juegos bipersonales con información perfecta con y sin elementos de azar

Vamos a asumir ciertas cosas (II)

- **Suma nula:** lo que gana uno, lo pierde el otro
- **Dos agentes** (se puede generalizar, Max^n)
- **Información completa:** se conoce en cada momento el estado completo del juego
- **Deterministas o de información perfecta:** no entra en juego el azar (se puede generalizar)
- **Alternados:** las decisiones de cada agente se toman de forma alternada

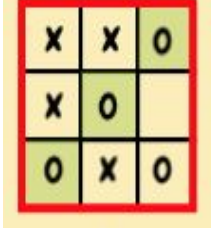
https://es.wikipedia.org/wiki/Juego_de_suma_cero

Qué conocimiento mínimo necesitamos

Los agentes máx. y min. disponen de un conocimiento mínimo *a priori* que se puede resumir en:

- s_0 Estado inicial
- $expandir(s) : \{s_1, \dots, s_n\}$ Conjunto finito de estados sucesores
- $esTerminal(s) : true \mid false$ Si es un nodo terminal
- $utilidad(s): k, k \in \mathbb{R}$ función parcial de utilidad del juego

Función de utilidad

Gana Max	Gana Min	Empate
		

Valor negativo:

-1

-10

-inf

Valor positivo:

+1

+10

+inf

Búsqueda minimax - algoritmo general

1. Generar el árbol de juego
2. Aplicar la función de utilidad en cada nodo terminal
3. Propagar las utilidades **hacia arriba**
 - a. Nodos max, se elige la utilidad máxima
 - b. Nodos min, se elige la utilidad mínima
4. Elegir la jugada óptima de max en el nodo raíz
 - a. El sucesor del raíz con utilidad máxima

Ojo a esto, que implica leer el árbol de juego de abajo a arriba

Reglas del juego

Tres en Raya:

- dos jugadores (*min* y *max*)
- los jugadores van poniendo fichas en las casillas de un tablero 3x3
 - *max* usa las fichas **X** / *min* usa las fichas **O**
 - una casilla puede contener como mucho una ficha
- Reglas:
 - Inicialmente el tablero está vacío
 - *max* empieza y los jugadores se van alternando en poner sus fichas
 - *max* gana si obtiene una raya de tres fichas **X**
 - *min* gana si obtiene una raya de tres fichas **O**
 - si todas las casillas están ocupadas sin que haya una raya de 3 fichas del mismo tipo, hay empate

X	O	X
O	X	O
	O	X

gana *max*
10

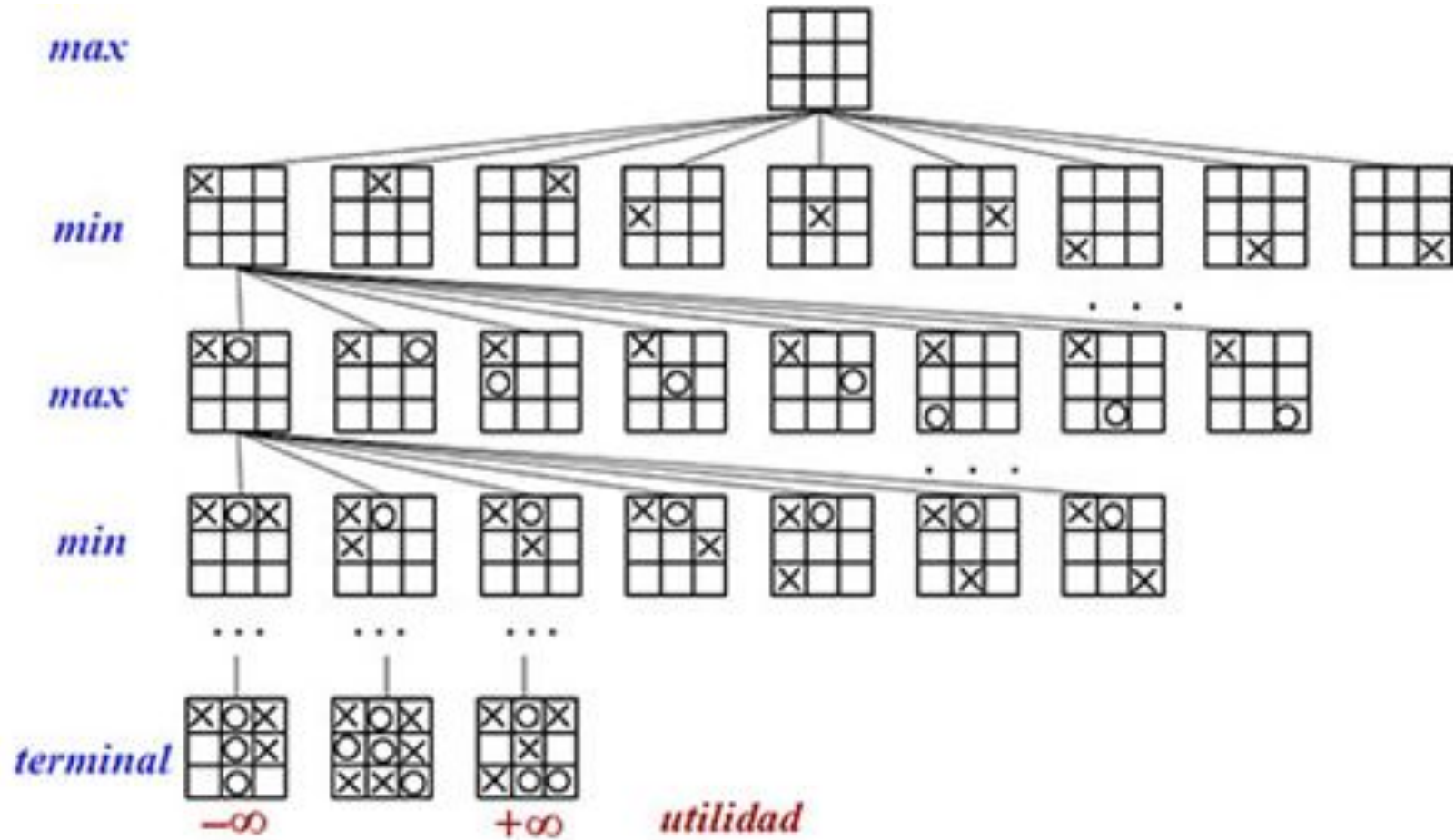
O	O	X
O	X	O
O	X	X

gana *min*
-10

X	O	X
O	X	O
O	X	O

empate
0

Ejemplo de Árbol de juego

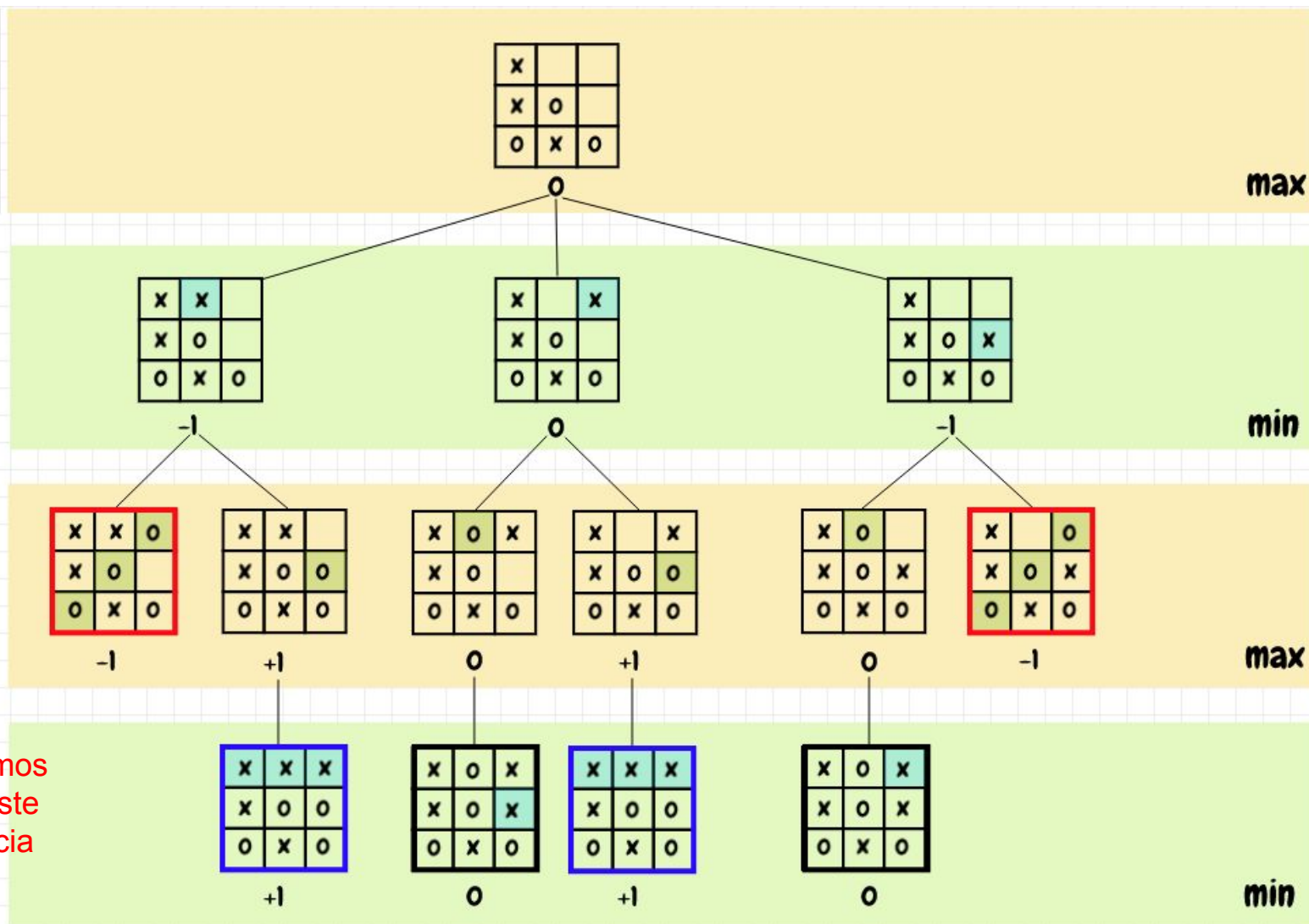


Tres en raya - Minimax (poco informado)

X wins: +1

O wins: -1

Draw: 0



<https://nestedsoftware.com/2019/06/15/tic-tac-toe-with-the-minimax-algorithm-5988.123625.html>

Demo

<https://tictactoe-api-server.herokuapp.com/>

<https://simo.sh/projects/tic-tac-toe-ai>

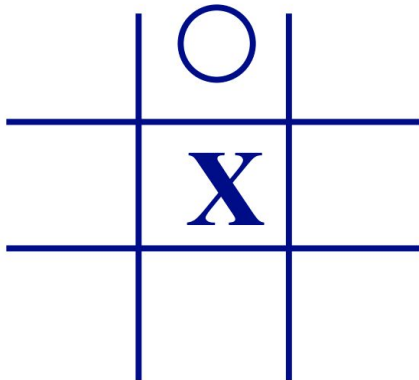
<https://mostafa-samir.github.io/Tic-Tac-Toe-AI/>

<https://github.com/simpleai-team/simpleai>

Función de utilidad


Se pueden utilizar a modo de ‘heurística’


Función de evaluación $f(n)$ para “Tres en raya”: número de posibilidades de hacer tres en raya del jugador menos número de posibilidades de hacer tres en raya del oponente

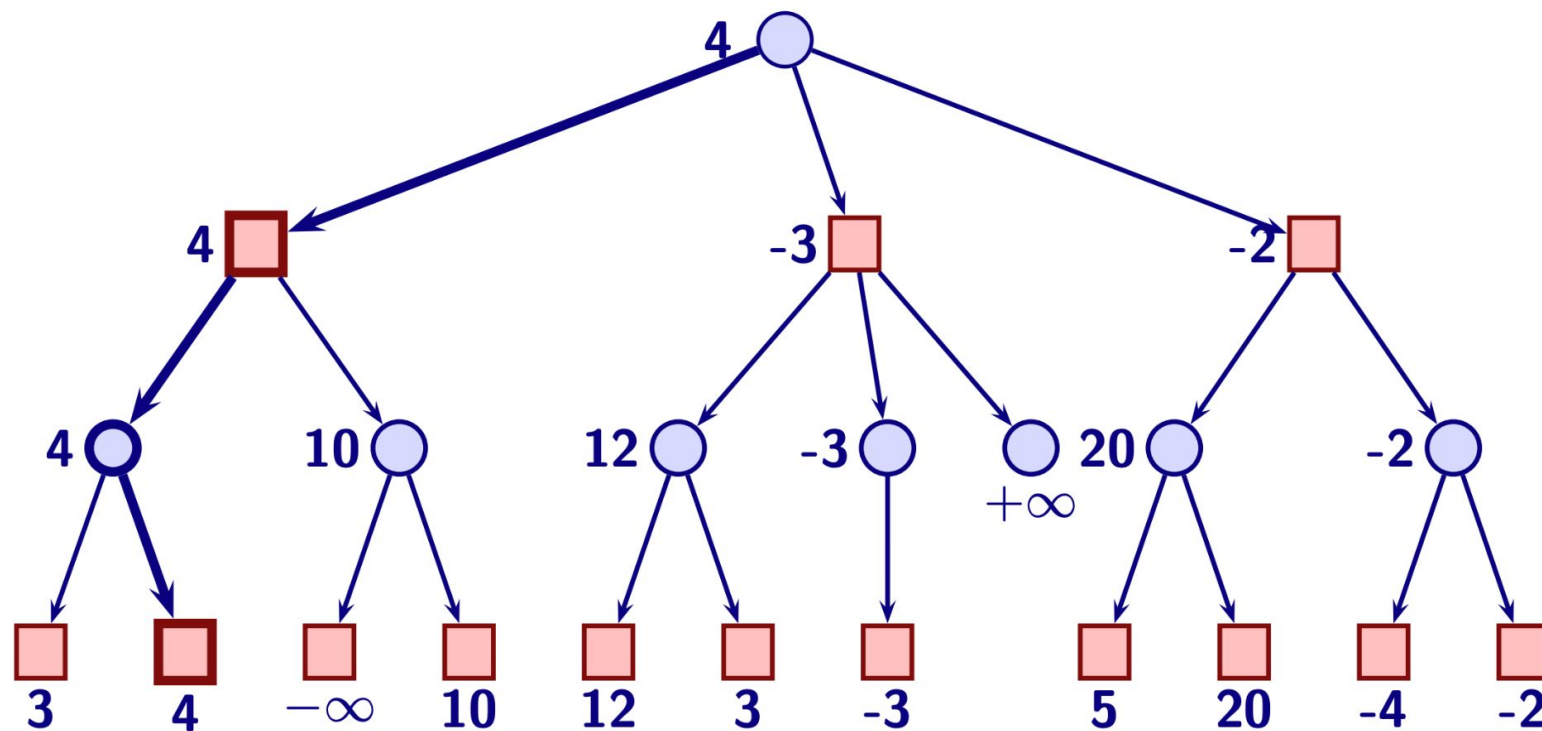


- Si colocamos la primera ficha en:
 - el centro: 4 posibilidades para hacer 3 en raya ($f(n) = 4 - 4 = 0$)
 - en una esquina: 3 posibilidades ($f(n) = 3 - 5 = -2$)
 - en el centro de un lateral: 2 posibilidades ($f(n) = 2 - 6 = -4$)

Tres en raya - Minimax (mejor informado)

v_i  Nodos MAX (juega el Minimax)

v_i  Nodos MIN (juega el oponente)



<http://ocw.uc3m.es/cursos-archivados/inteligencia-artificial-2/material-de-clase-1/minimax.pdf>

Tres en raya - Minimax (mejor informado)

Minimax

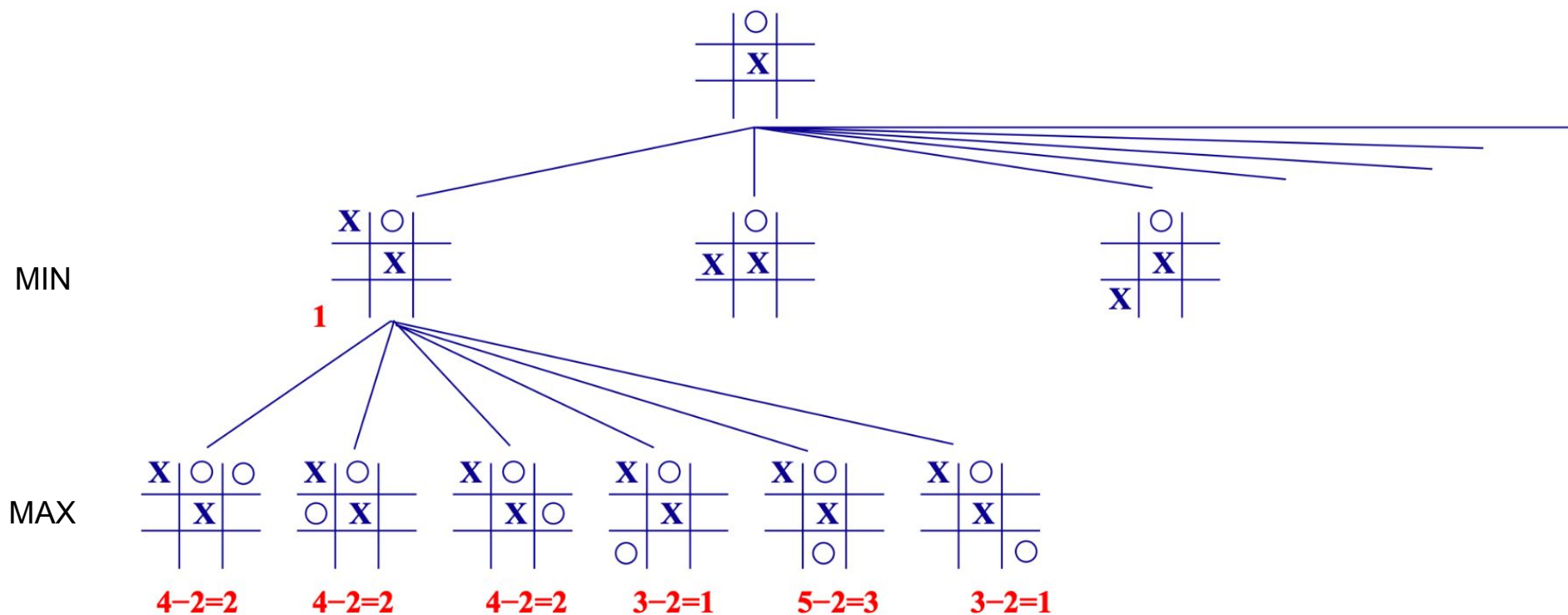
Valores en caso de **estado meta o empate**

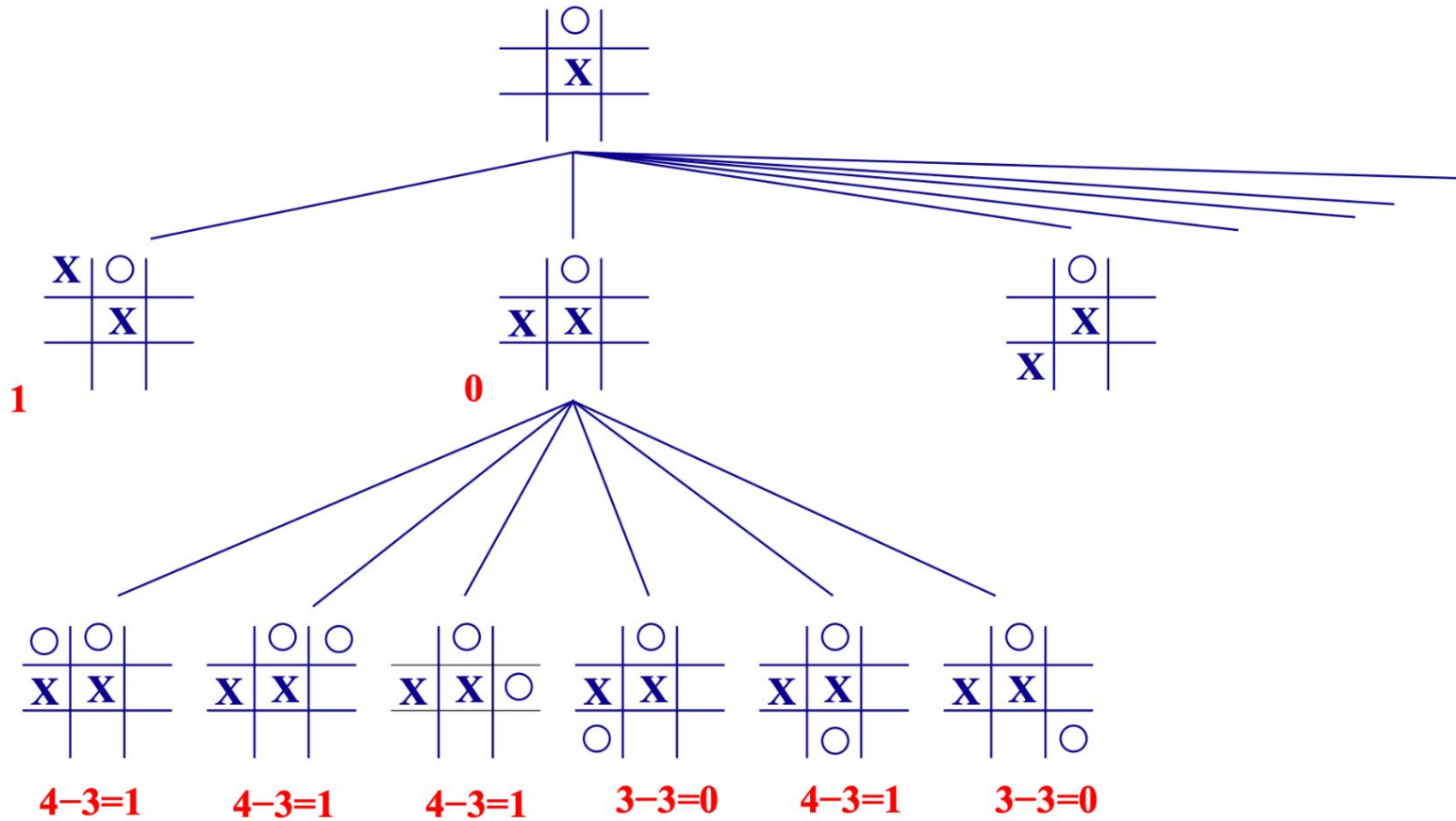
$$f(n) = \begin{cases} +\infty & \text{si } n \text{ es una situación ganadora} \\ -\infty & \text{si } n \text{ es una situación perdedora} \\ 0 & \text{si } n \text{ es una situación de empate} \\ f_{\text{ev}}(n) & \text{si } p = \text{Profundidad-m}{\acute{a}}\text{xima} \\ \max_{S_i \in S(n)} f(S_i) & \text{si } n \text{ es nodo MAX y } p < p_{\text{max}} \\ \min_{S_i \in S(n)} f(S_i) & \text{si } n \text{ es nodo MIN y } p < p_{\text{max}} \end{cases}$$

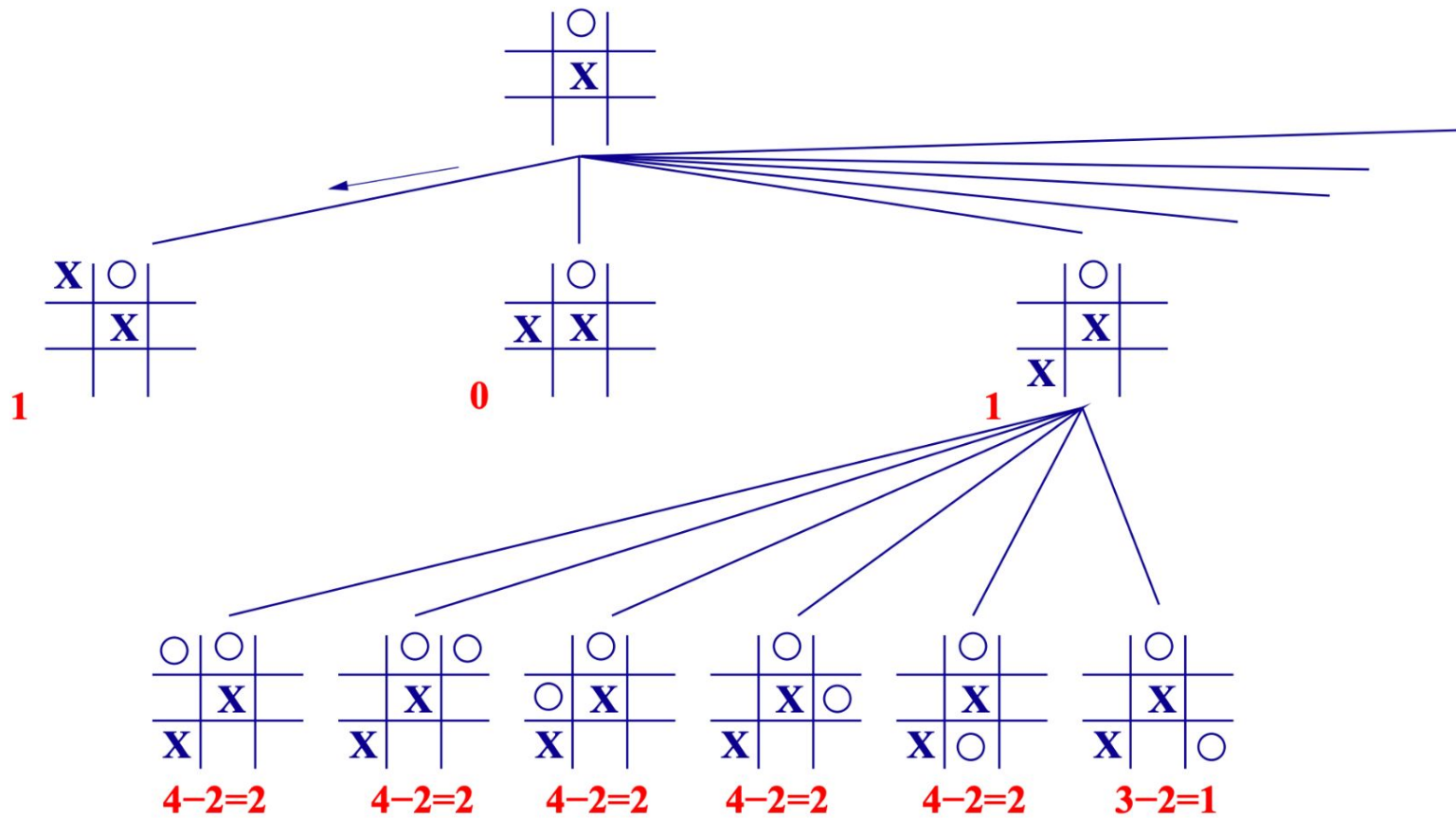
Valores en caso de **estados intermedios**

nº posibilidades de 3 en raya del jugador - nº posibilidades de 3 en raya del oponente

Rojo = función de evaluación en los nodos intermedios







El problema de búsqueda es intratable; no se puede realizar en un tiempo razonable

Veremos cómo resolverlo en la próxima clase :)

¿Preguntas?

- ▶ Contacto: nerea.luis@unir.es



www.unir.net