ikdoeict.be

Programmeren4
Wetenschappelijke Toepassingen 2
2015 -2016

Katja Verbeeck
Joris Maervoet
Tim Vermeulen

Game AI

Outline

- **(light) Intro in AI**
- **Game AI :**
  - Zero-sum games
  - Minimax
  - Varianten van minimax
- **Opdracht**

Intro Artificial Intelligence

•**Kunstmatige intelligentie (KI) of artificiële intelligentie (AI) is de wetenschap die zich bezighoudt met het creëren van een artefact dat een vorm van intelligentie vertoont.**

•**Wanneer is iets intelligent?**

→ **Turing Test ,** ***Alan Turing,***
Computing Machinery and Intelligence, 1950

•**Computationele Intelligentie:**

–Computationele technieken gebruiken
om natuurlijke intelligente
phenomenen te modelleren
→ **machine leertechnieken**

–Intelligente software → **Software Agenten**

Turing Test

http://www.bbc.com/future/story/20150724-the-problem-with-the-turing-test

Scott: … Do you understand why I'm asking such basic questions? Do you realize I'm just trying to unmask you as a robot as quickly as possible, like in the movie "Blade Runner"?
Eugene: … wait
Scott: Do you think your ability to fool unsophisticated judges indicates a flaw with the Turing Test itself, or merely with the way people have interpreted the test?
Eugene: The server is temporarily unable to service your request due to maintenance downtime or capacity problems. Please try again later.

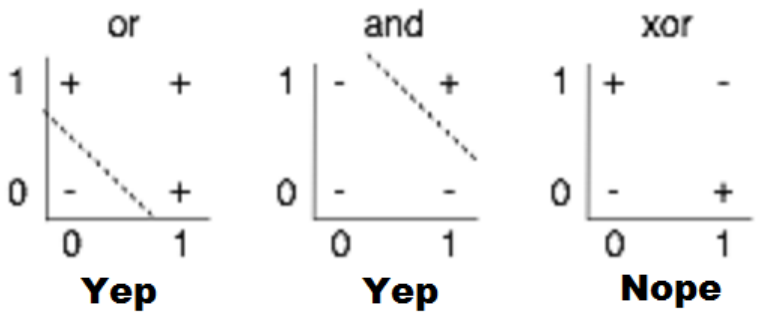
Can a computer detect meaningful relationships between the people it is watching? (Credit: Getty Images)

http://techland.time.com/2013/01/04/finally-a-computer-that-writes-contemporary-music-without-human-help/

Meet Iamus, a computer at the University of Malaga in Spain capable of composing contemporary classical music without human aid.

Beetje geschiedenis

- **1940 Programeerbare digitale computers**
- **1943 – 1956 : McCulloch & Pitts : het brein modelleren als een booleaans circuit**
- **Darthmouth meeting : ontstaan van de naam AI**
- **1956 – 1974 : de gouden jaren**
- **AI Winter : de XOR functie kan niet gemodelleerd worden**



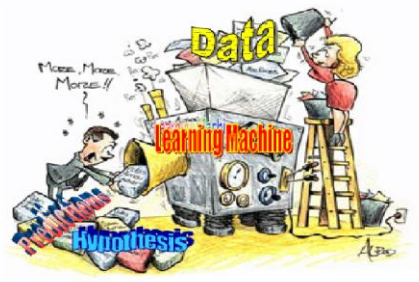
Visualization of the limitations of Perceptrons. Finding a linear function on the inputs X,Y to correctly ouput + or - is equivalent to drawing a line on this 2D graph separating all + cases from - cases; clearly, for the third case this is impossible.

http://www.andreykurenkov.com/writing/a-brief-history-of-neural-nets-and-deep-learning/

Beetje geschiedenis

- 1986 Ontstaan van tak machine learning
  - Neurale netwerken (met gebruik van hidden layers) worden weer populair
- 1995 AI wordt als aparte wetenschap gezien
  - Integratie van leren, redeneren, kennisrepresentatie
  - Toepassingen in taal, beelden, data mining
- 2006 gezichtsherkenning (beeldverwerking)
- 2003 – 2007 DARPA grand challenge
- 2011 IBM Watson

What do you mean? Machines can they learn?



(from Eric Xing lectures on ML)

Showing that it works ...

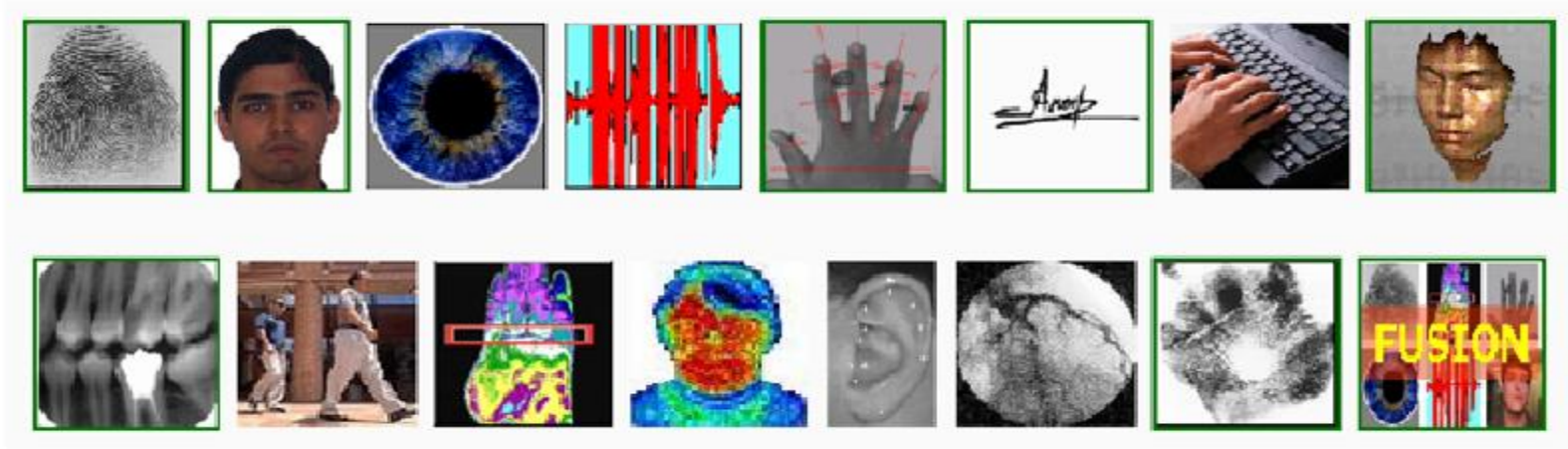


Figure 4: Biometric recognition.

Showing that it works ...

Figure 9: License plate recognition: US license plates.

Deep-Blue

1997 Deep Blue de eerste schaak computer die de toenmalige wereld kampioen Garry Kasparov versloeg. Deep Blue kon 200 miljoen posities doorzoeken per seconde, maakt hierbij gebruik van gesofisticeerde evaluatie functies en (undisclosed) zoekmethoden om ongeveer 40 zetten vooraf te 'denken'

Mars Rover

- Directe remote controle is niet mogelijk
  - Het duurt + / - 10 minuten om een bericht te sturen vanop aarde naar Mars
  - Het is niet mogelijk om met voldoende nauwkeurigheid te voorspellen wat de robot zal tegenkomen in zijn omgeving.
  - → De robot moet autonoom kunnen werken!

DARPA challenge

Robotic Control , DARPA (URBAN) Challenge

2004 : the Mojave Desert for 150 miles (240 km) Geen enkel van de automatische voertuigen was in staat om de finish te bereiken. CMU's Red Team kon als enige 11.78 km autonoom afleggen

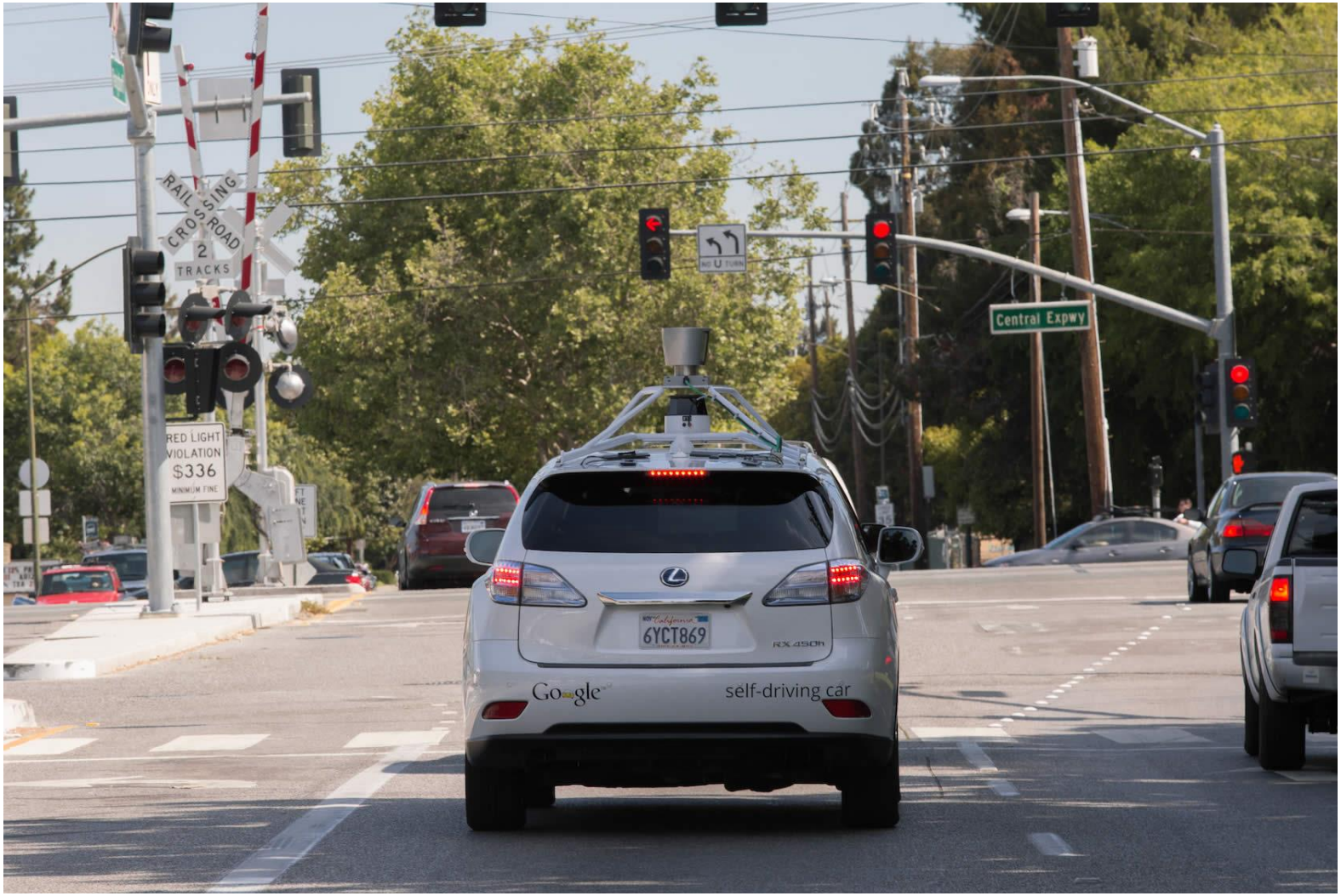
2005 212 km (132 mi) off-road course. Deze keer konden 22 van de 23 finalisten de afstand van 2004 evenaren, 5 wagens konden succesvol het volledige parcours afleggen.

2007 Deze keer moest er 96 kilometer afgelegd worden in een stedelijk gebied in minder dan 6u (60 mi) . De wagens moesten zich houden aan de verkeersregels en onderhandelen met andere voertuigen en obstakels.

Google CARS

https://www.google.com/selfdrivingcar/

2009 de autonome Toyota Prius op een snelweg in California.



https://www.google.com/selfdrivingcar/

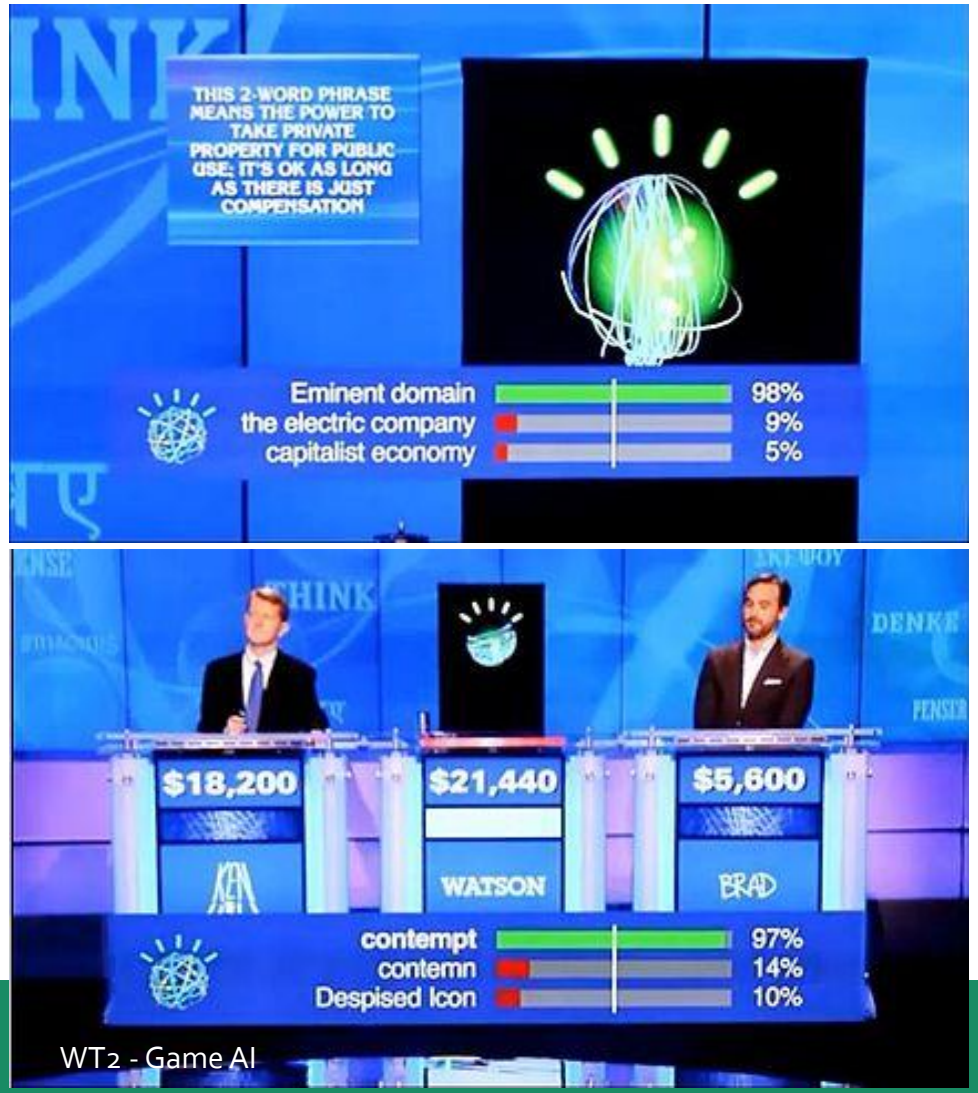
2013 de focus ligt nu op stadsverkeer, een veel complexere omgeving dan de snelwegen.

February 11, 2013 : Google wil autonome voertuigen in de straten op en termijn van 3 a 5 Jaren.

Vandaag hebben ze zelf een voertuig ontworpen dat proefprojecten draait in verschillende steden.

https://www.google.com/selfdrivingcar/

IBM WATSON



the Jeopardy!quiz show

Het antwoord wordt geprojecteerd, de vraagstelling moet door de kandidaten gevonden worden.

Game AI

Typisch gebruikt voor :

- Beweging
- Beslissen waarnaartoe te bewegen (path planning)
- Tactische en strategische beslissingen te nemen

"intelligente" simulatie van gedrag (i.e. een uitgedokterde strategie en vooral ook gepaste reacties)

Game AI

- **Ontwikkeld om suboptimaal te zijn**
  - Performantie is niet de hoofdzaak : het problem moet niet optimal opgelost worden; de heuristische aanpak is vaak beter en realistischer
- **Ontwikkeld om te entertainen**
  - Het is niet de bedoeling om de spelers te frustreren. Goeie Game AI moet ook kunnen verliezen, en vooral spelers het gevoel geven dat ze slim zijn en het spel in handen hebben.
- **Ontwikkeld om de illusie van intelligentie op te wekken**
  - Als een speller denkt dat een bot intelligent is, dan is hij dat ook.
  - Is soms simpel : laat een karakter de andere karakters volgen met zijn hoofd
  - Is soms moeilijk: karakters die tegen een muur botsen komen stom over – of karakters die vals spelen (gaan gewoon door de muur bv) geven een verkeerde indruk.

Welke Games bestuderen we hier?

Strategische spellen met volgende typische karakteristieken :

1. Er moet een sequentie van zetten gedaan
2. Zetten zijn onderhevig aan regels
3. Bepaalde zetten kunnen punten opleveren
4. Het doel is om het aantal punten te maximaliseren

Zero-sum Games

- Onderhandelingssituatie waarbij de belangen van de partijen recht tegenover elkaar staan. **De winst van de een is het verlies van de ander**.

- een spel waarbij de opbrengst een constante waarde heeft. Als een speler wint, moeten de andere spelers evenveel verliezen: **er is maar 1 buit te verdelen**. Een voorbeeld is de schaakpartij, deze kan op drie manieren eindigen: 1-0, ½-½, of 0-1; in alle gevallen is de som van de scores gelijk.

- Voor twee spelers bestaat steeds een *ideale strategie* - afwijken van deze strategie is voor geen van beide spelers voordelig. Dit kan wel een zogenaamde *gemixte strategie* zijn. Bijvoorbeeld bij het bekende spelletje steen, papier, schaar is de ideale strategie om elk van de 3 mogelijkheden met kans 1/3 te kiezen.

2spelers – zero-sum

- 2 spelers

- De zetten worden afwisselend geplaatst
  - MAX speler
  - MIN speler

- De beschikbare informative kan variëren :
  - Perfecten info : schaak, dammen, tic-tac-toe…

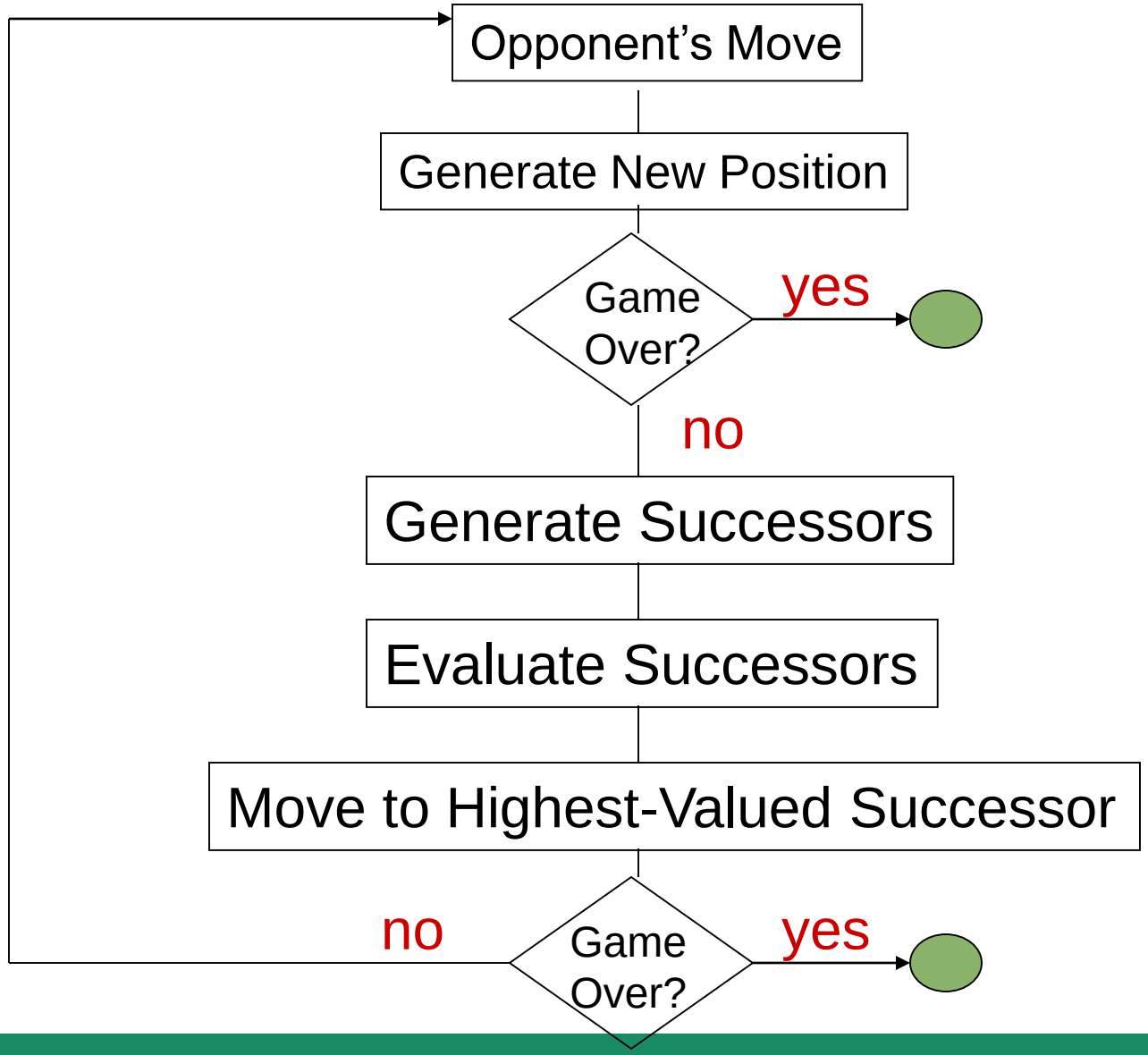
    typisch hier is dat er geen random factor in het spel is, beide spelers hebben ook exact dezelfde info
  - Imperfecte info: poker, Stratego, bridge…

Een game is een zoekprobleem maar …

Tegenstanders zijn onvoorspelbaar → je moet een zet verzinnen rekening houdend met elke mogelijk antwoord van de tegenstander.

Er is een tijdslimiet → wanneer het antwoord niet gevonden kan worden in die tijdspanne, moet je terugvallen op een weloverwogen gok, benadering of vuistregel (ook wel heuristiek genoemd)

2spelers – zero-sum



Minimax strategie

Von Neumann (1928)

Game Tree voor het spel tic-tac-toe (2-spelers, geen toeval, beurten)

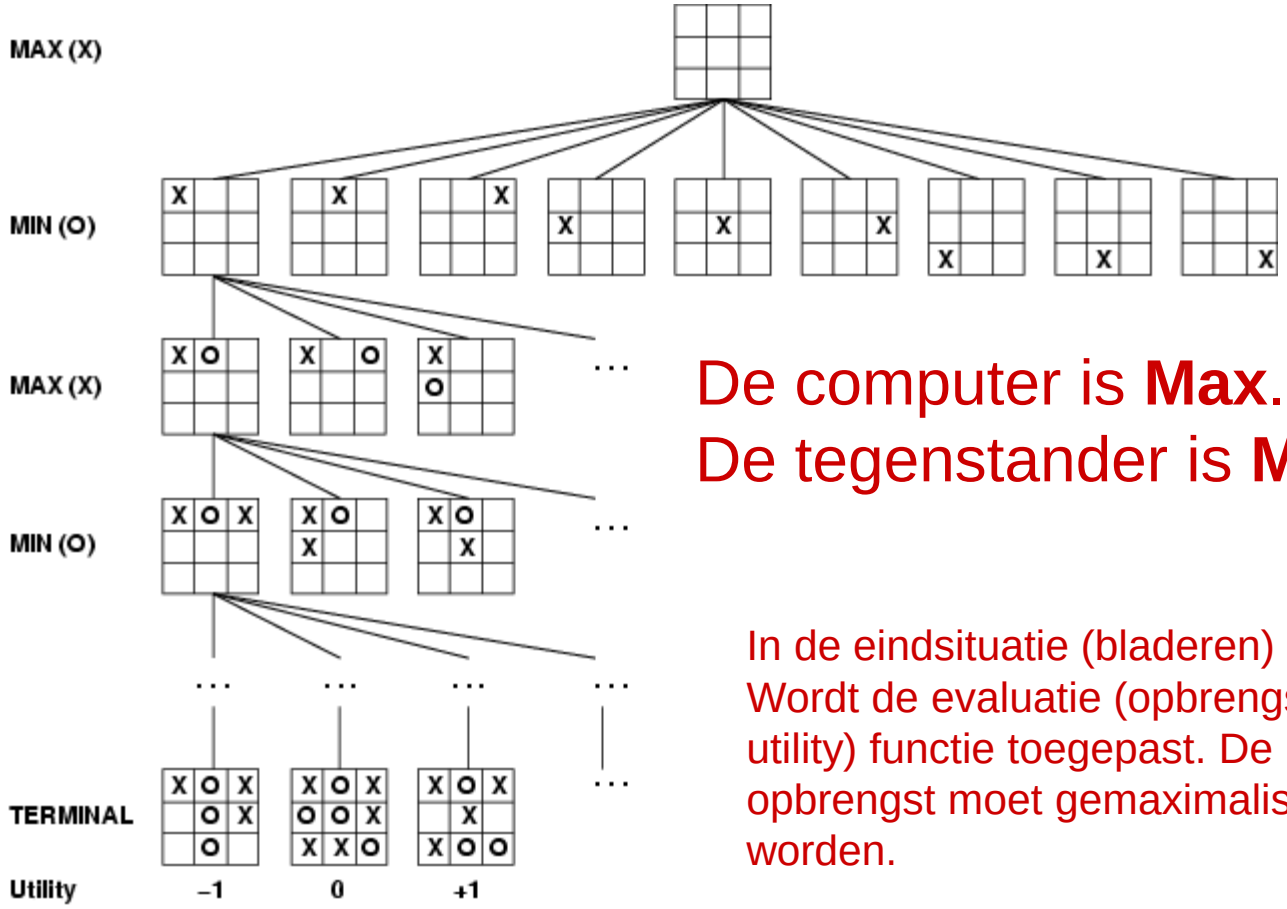
computer's beurt

tegenstander

computer's beurt

tegenstander

De eindknopen worden geëvalueerd



De computer is **Max**.
De tegenstander is **Min**.

In de eindsituatie (bladeren) Wordt de evaluatie (opbrengst / utility) functie toegepast. De opbrengst moet gemaximaliseerd worden.

Mini-Max Terminology

Utility / evaluatie functie: de functie die de punten verdeelt – deze weet je als je in een eindknoop van de game tree belandt.

backed-up value

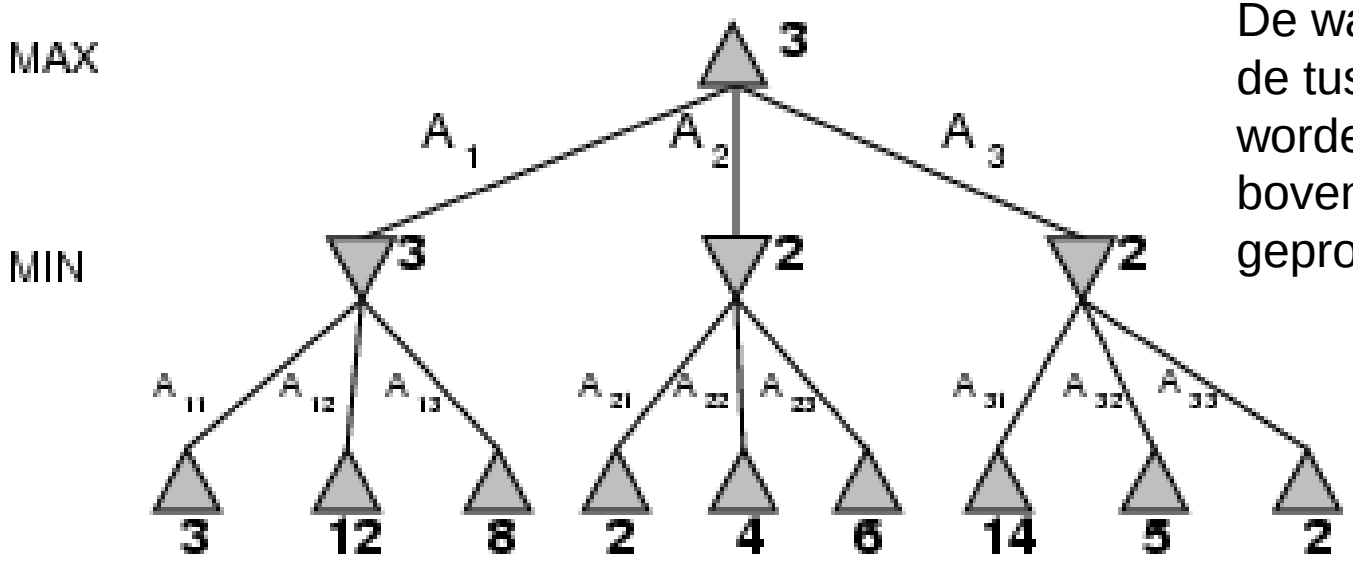
- van een max-positie: de waarde van de beste opvolgersknoop (successor)
- van een min-positie: de waarde van de kleinste opvolgersknoop

minimax procedure: zoek enkele levels diep in de boom tot je de eindknopen bereikt; daar kan je de evaluatie functie uitvoeren en de waarden naar boven propageren volgens het max of min principe. Selecteer dan in de bovenste (root) knop de beste zet.

Minimax : voorbeeld met slechts 2 beurten

Kies de zet met de hoogste minimax waarde

= dit geeft de beste opbrengst als de tegenstander optimaal speelt



De waarden van de tussenknopen worden naar boven gepropageerd

Algemeen : Minimax algoritme

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Het minimax algoritme doorloopt de volledige game tree via depth-first exploratie.

Recursie

Het minimax-algoritme is "recursief": neem in bladeren de evaluatie-functie, in MAX-knopen het maximum van de kinderen, in MIN-knopen het minimum van de kinderen. MAX- en MIN-knopen wisselen elkaar af.

Het afdalen in de boom gebeurt op een diepte-eerst manier.

Complexiteit

Wanneer de game tree maximal diepte ***m*** heeft en er zijn op elk ogenblik in het spel maximaal ***b*** zetten mogelijk, dan heeft het minimax algoritme een tijdscomplexiteit van $O(b^m)$

Exponentieel dus !

(Via de techniek van prunen (takken wegknippen, zie verder) kan de exponent echter wel gehalveerd worden …)

Cut-off : minder zetten vooruit denken

Bij cut-off search vervang je in het minimax-algoritme de test op eindtoestanden door een “cut-off test” — een evaluatie-functie voor de tussenknopen dus.

Nadelen :

- horizon-probleem soms wordt een noodzakelijke slechte zet uit beeld geduwd door extra (minder slechte) zetten.
- Voorbeeld schaken : b = 35 en $b^m \approx 10^6$, krijg je m = 4. En 4-ply vooruit kijken is amateur niveau. 8-ply ≈ PC of schaakmeester, en 12-ply is wereldtop-niveau (computer of mens).

Cut-off Tic-tac-toe

Bijvoorbeeld de evaluatiefunctie kan voor elke tussenknoop als volgt gekozen worden

E =

(het aantal beschikbare rijen, kolommen en diagonalen die nog mogelijk zijn voor MAX)

-

(het aantal beschikbare rijen, kolommen en diagonalen die nog mogelijk zijn voor MIN)

MAX zal deze functie maximaliseren

MIN zal deze functie minimaliseren

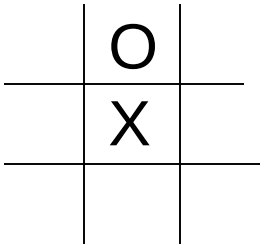
Deze evaluatiefunctie bevat symmetrieën.

Voorbeeld gebruikt diepte 2

Deze evaluatiefunctie is een heuristiek : met de logica van deze functie hoopt men goede resultaten te bereiken (maar deze is zeker niet bewezen!)

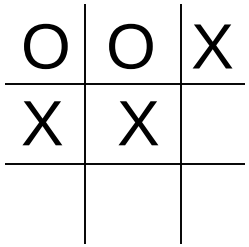
Voorbeeld

X = Computer; O = Opponent



	X	O
rows	2	2
cols	2	2
diags	2	0

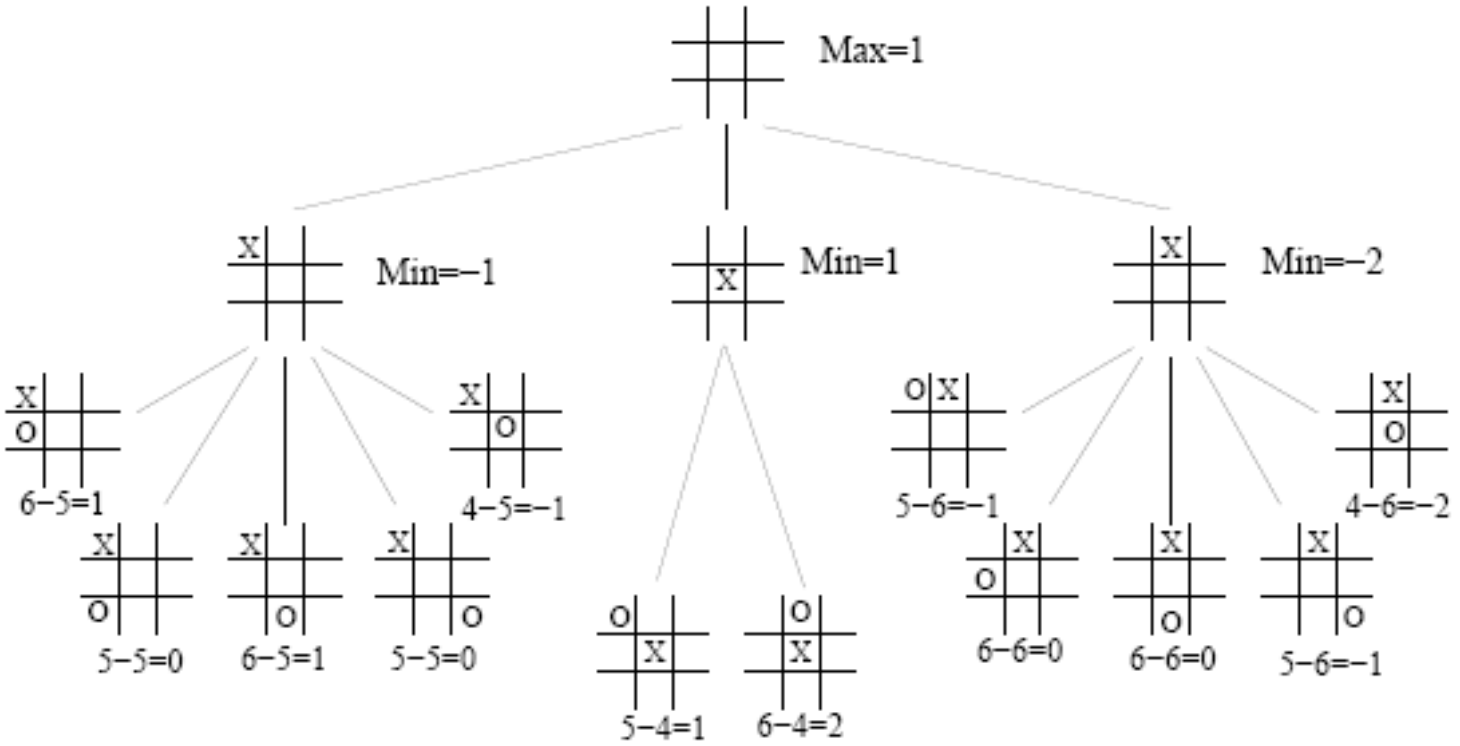
E = 6 – 4 = 2



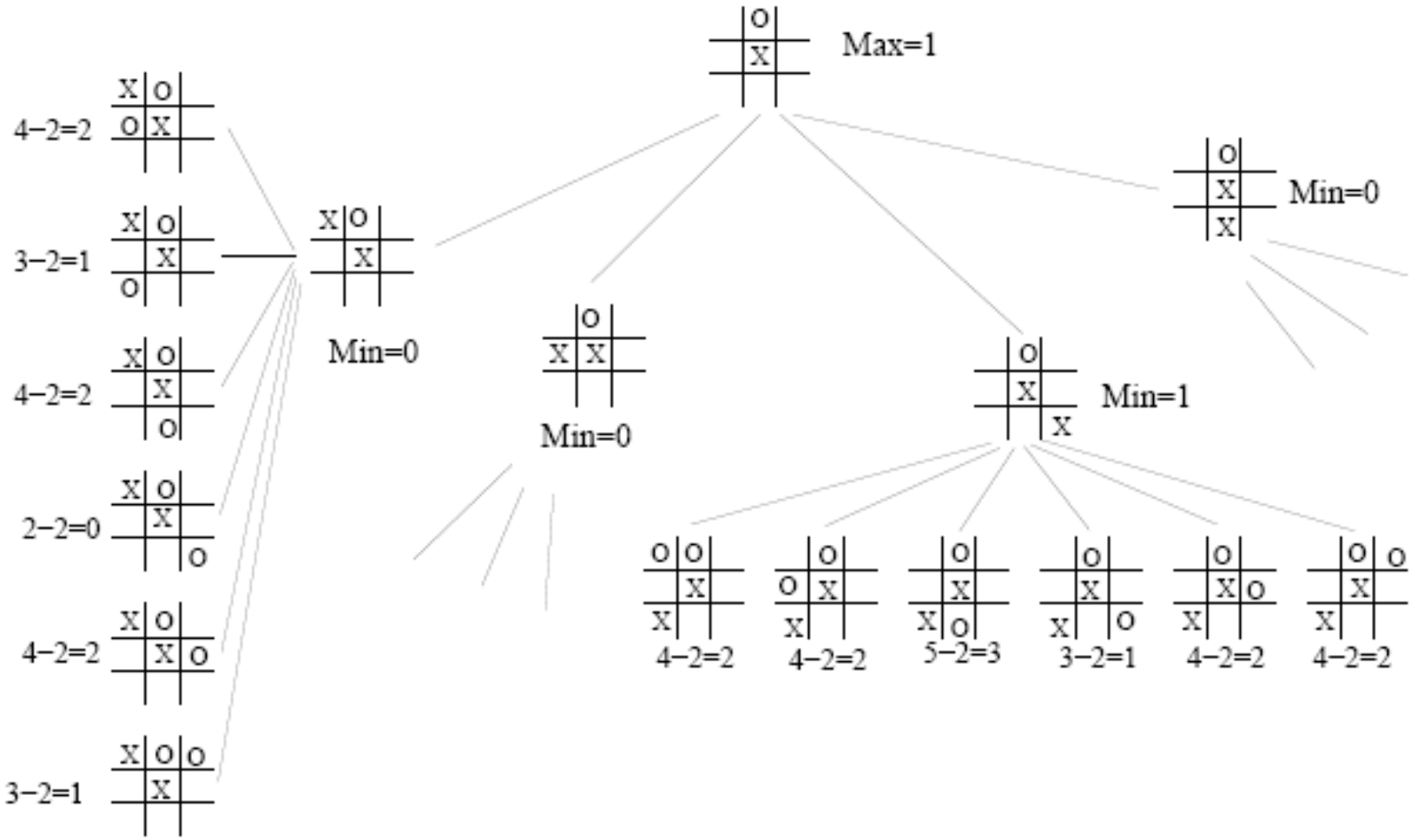
	X	O
Rows	2	1
Cols	1	0
Diags	1	0

E = 4 – 1 = 3

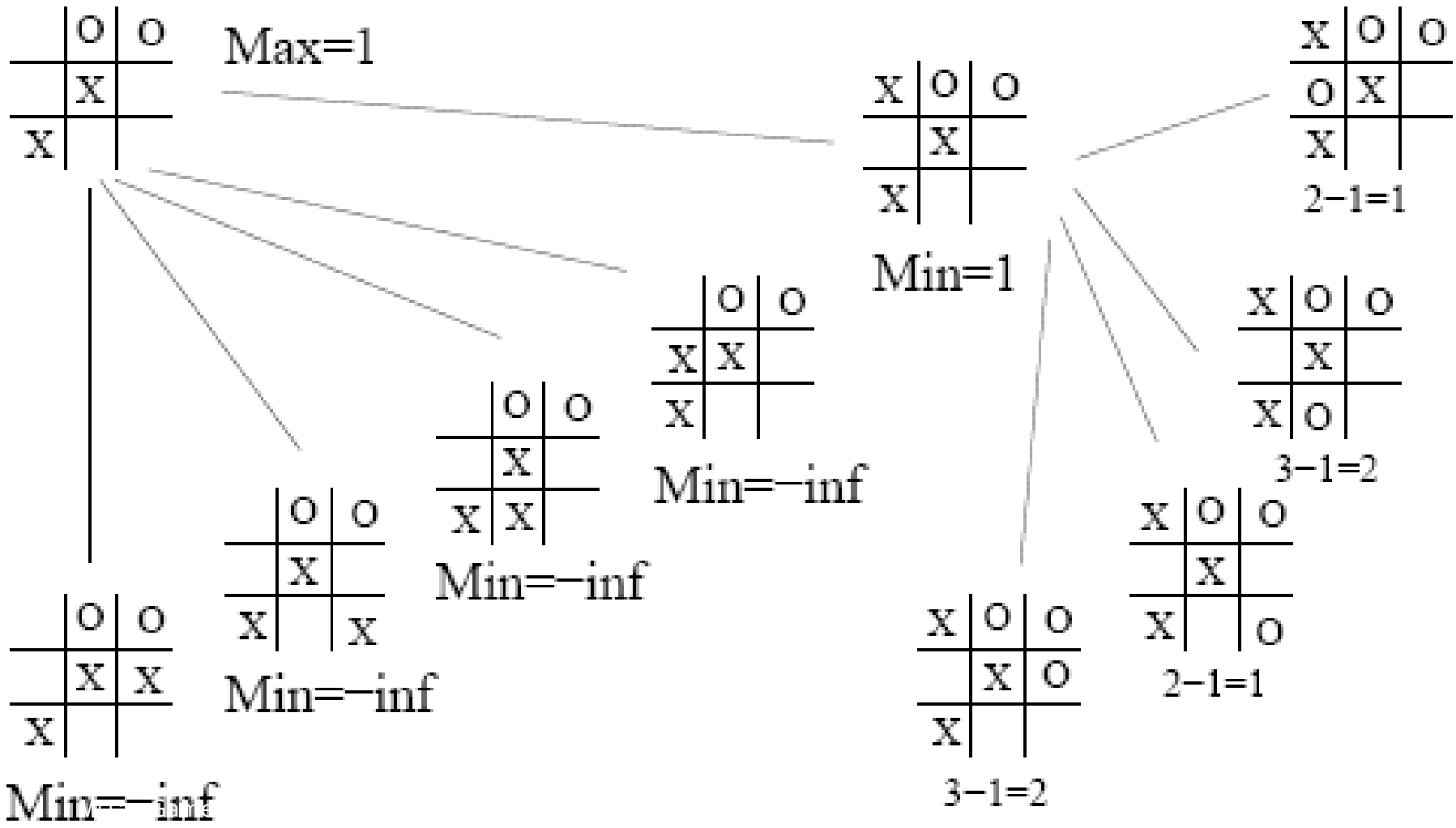
Voorbeeld : tic-tac-toe



Voorbeeld : tic-tac-toe



Example: tic-tac-toe



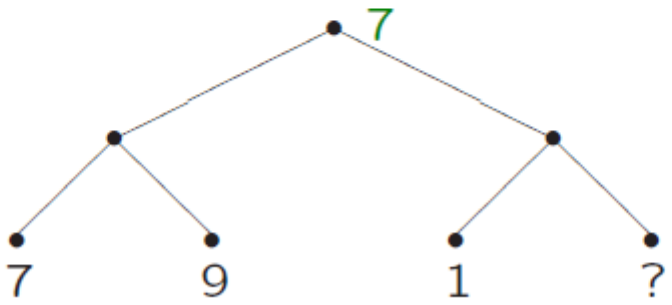
Varianten op minimax

alpha – beta pruning

Negamax

Het idee : snoeien in de game tree = pruning

Het basisidee van het α-β-algoritme is het volgende: om de minimax-waarde in de wortel van onderstaande boom te berekenen is de waarde rechts onderin niet van belang — en die subboom kun je prunen (snoeien). Immers, het linker kind van de wortel is 7, en het rechterkind moet dus hoger dan 7 zijn wil het nog invloed uitoefenen. Nu is het (dankzij de 1) hoogstens 1, en zijn we klaar: waarde 7.

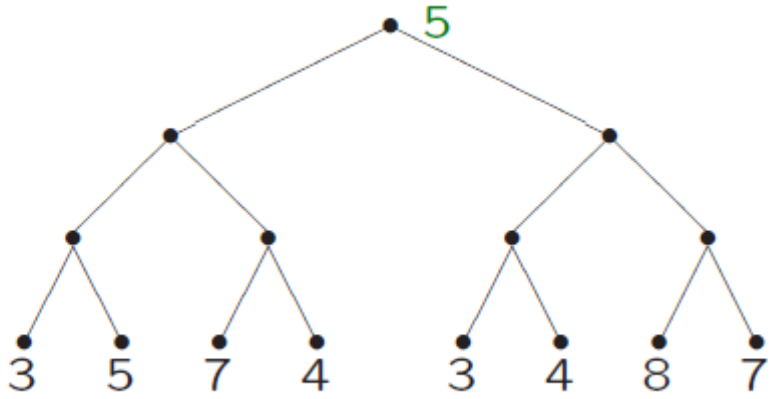


MAX aan zet

MIN aan zet

De waarde van ? Heeft geen belang meer, de min speler kiest een waarde van 1 of lager, terwijl de max speler 7 of hoger zal prefereren

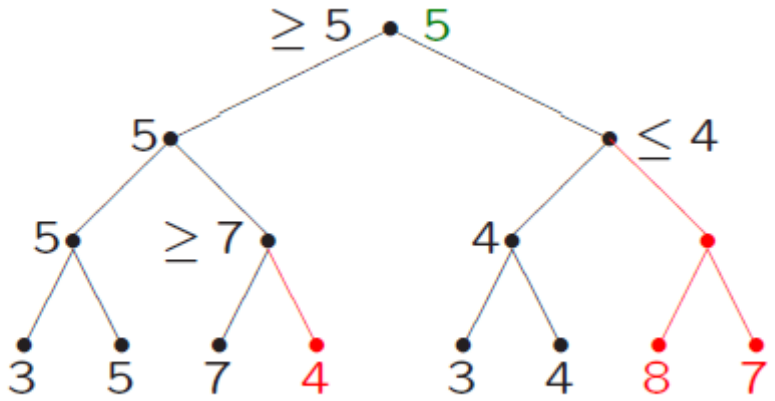
Voorbeeld



MAX aan zet

MIN aan zet

MAX aan zet



rood wordt

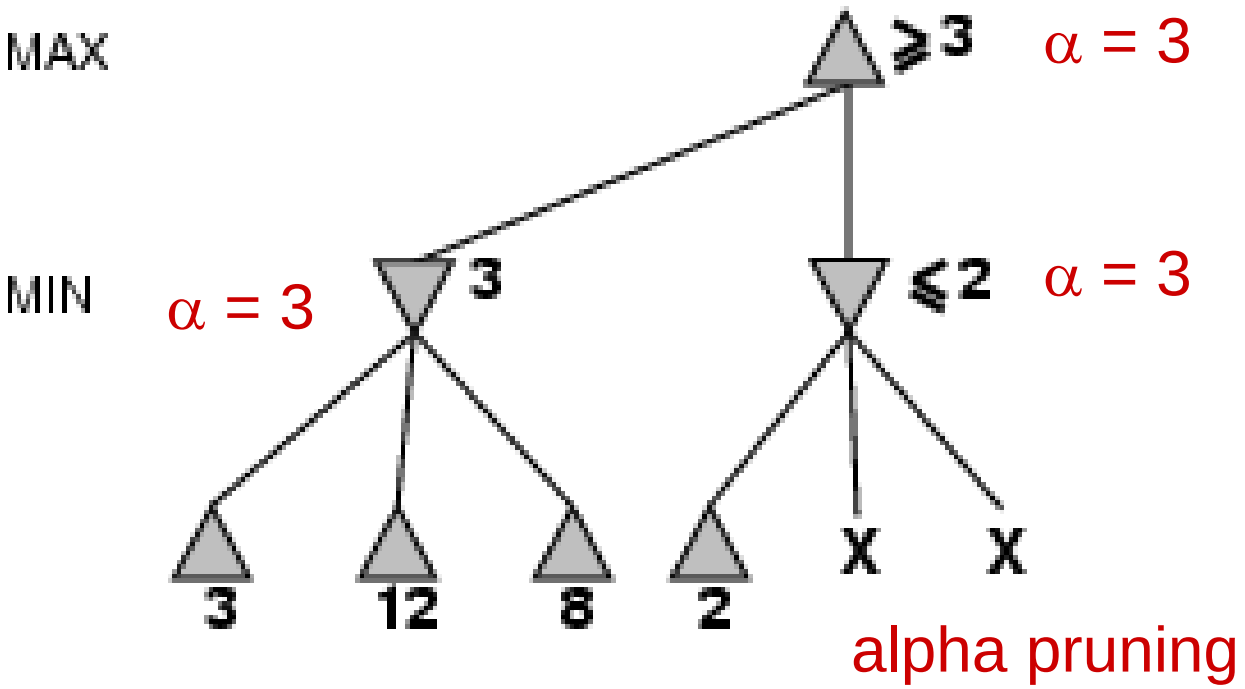
gepruned

Alpha-beta pruning

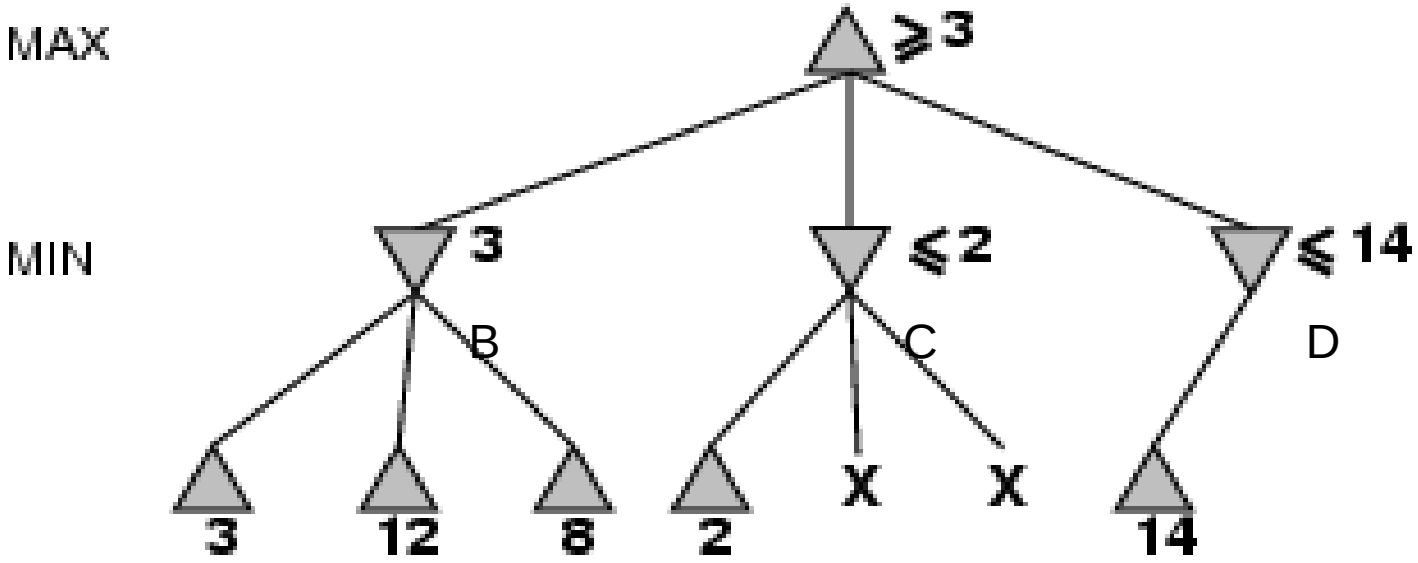
Alpha-beta pruning werkt aan de hand van 2 parameters : α en β

- Normaal geldt : α < β
- α houdt de beste waarde voor MAX op het huidige pad bij
- β houdt de beste waarde voor MIN op het huidige pad bij
- Initieel is de waarde voor α = - ∞ , β = + ∞
- Een tak kan weggeknipt worden wanneer de waarde van de huidige knoop kleiner is dan de huidige waarde van *α* voor MAX (*α pruning*)
- Een tak kan weggeknipt worden wanneer de waarde van de huidige knoop groter is dan de huidige waarde van *β* value voor MIN (*β* pruning)

α-pruning

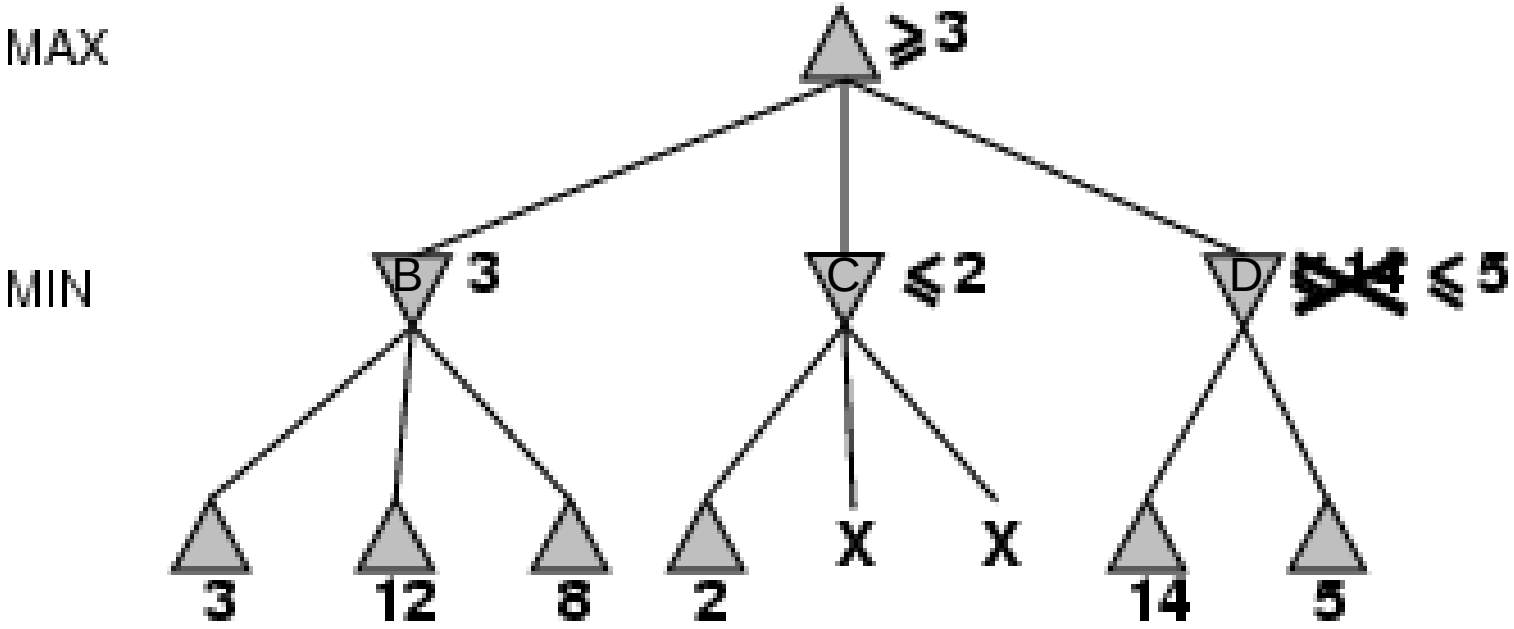


Alpha-beta pruning



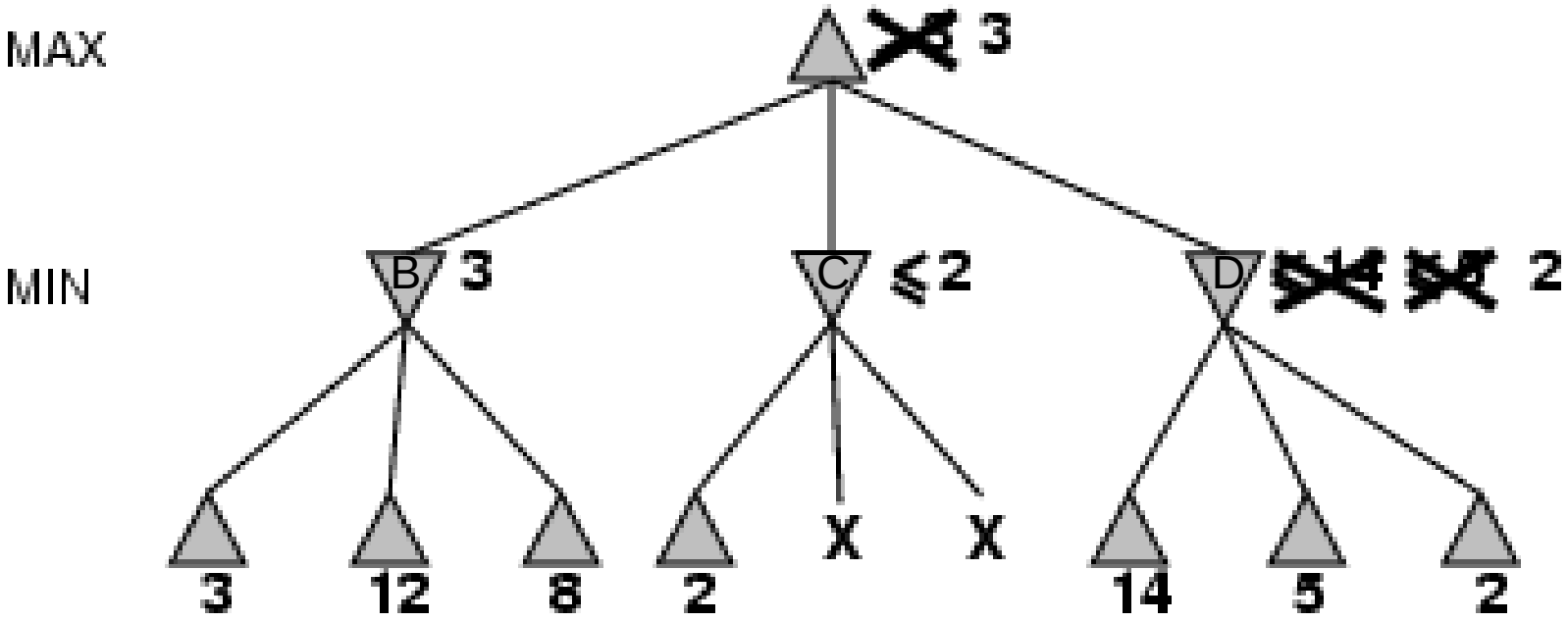
D, is een MIN knoop, dus de waarde zal maximaal 14 worden. Dit is op dit moment hoger dan MAX's beste waarde tot nogtoe (i.e., 3), vandaar dat de opvolgers van D wel bezocht moeten worden.

Alpha-beta pruning



De tweede opvolger van *D* geeft waarde 5, wat nog steeds beter is dan de huidige MAX waarde, dus ook de derde opvolger wordt bezocht.

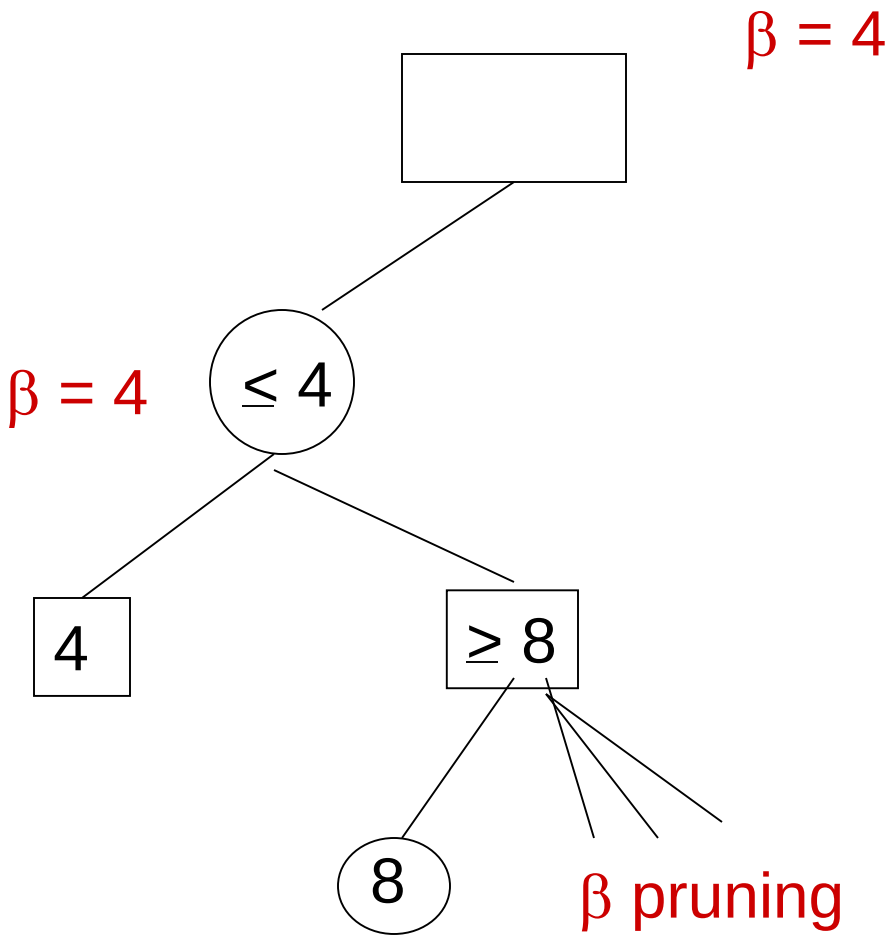
Alpha-beta pruning



De derde opvolger is slechts 2 waard, dus de MIN speler zal deze tak verkiezen en dus krijgt D de waarde 2.

MAX's beslissing zal dus zijn om zet B te verkiezen!

beta pruning



Eigenschappen alpha-beta pruning

Er geldt: α-β-pruning levert exact hetzelfde eindresultaat in de wortel als "gewoon" minimax.

De effectiviteit hangt sterk af van de volgorde waarin de kinderen (zetten) bekeken worden.

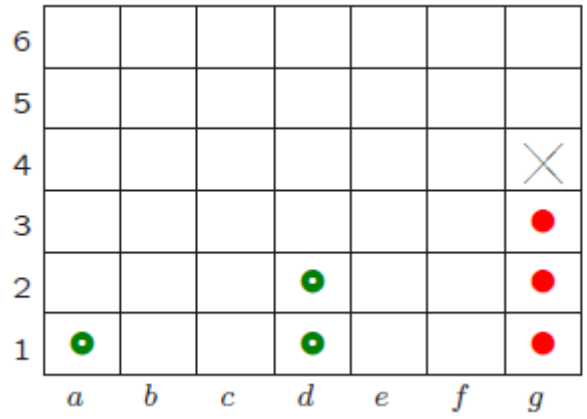
Met "perfecte ordening" bereik je tijdscomplexiteit $O(b^{m/2})$ (m is de diepte van de boom), dus je kunt effectief de zoekdiepte verdubbelen.

Opdracht : 4-op-een-rij

Implementeer aan de hand van een (afgeleide) minimax strategie een intelligente 4-op-een-rij speler. Toon via experimenten aan dat je AI speler zich effectief intelligent gedraagt. Zorg ervoor dat zowel een echte speler als de computer spelletjes kan spelen. (speler vs computer en computer vs computer) In deze opdracht verwachten we vooral veel experimenten en de bespreking ervan!

Denk na over een goede heuristiek en de symmetrie

- Met een goede heuristiek kan je je zetten ordenen waardoor pruning kan toegepast worden - de meest belovende kind knoop (volgens je heuristiek) eerst evalueren
- Vermijd om equivalente knopen opnieuw te evalueren
- Of onthoud slechte zetten
- Hoe doet je strategie het in **self-play ? Of tegen een random speler? of tegen nog een andere strategie? …. -> doe genoeg experimenten !!**



Vier-op-een-rij

null-move bekijk eerst voor de tegenstander goede zetten (sla je eigen zet in gedachten even over)

Reminder

Deel 1 - Het verslag -

+ moet een weergave van je onderzoek-stuk zijn

(hier : hoe interpreteer je het algoritme – welke extra keuzes (heuristieken) heb je meegenomen?)

+ inzoomen op het uiteindelijke resultaat : d.i.

  bespreek je experimenten en de resultaten ervan

+ event. kritische reflectie (future work)

+ Bibliografie

+ verwijzen vanuit de tekst naar de bibilio

+ maak je tekst leesvriendelijk : werk met extra figuren, tekeningen, een goede structuur, een inhoudstafel, …

Reminder

Deel 2 – De Code -

+ Maak je applicatie in c#

+ Je mag vertrekken van bestaande code maar niet zondermeer! (verwijzen + begrijpen + je eigen heuristieken erin verwerken)

+ Denk aan de gebruiksvriendelijkheid van je eindresultaat

+ Denk aan de opbouw en structuur van je code (uitbreidbaarheid, herbruikbaarheid, …)

Reminder

Deel 3 – De demo –

+ De demo vindt plaats tijdens het examen.

+ Je wordt in groepen ingedeeld en je kiest via tolinto zelf het uur op de juiste dag (alleen je eigen groep)

+ Bereid je demo voor ! Denk na over hoe je op een efficiënte manier ons kan overtuigen van je product !

+ Deadline : Vrijdag 27 mei (ook voor je update van opdracht 2)