

INF 4130

1. oktober 2009

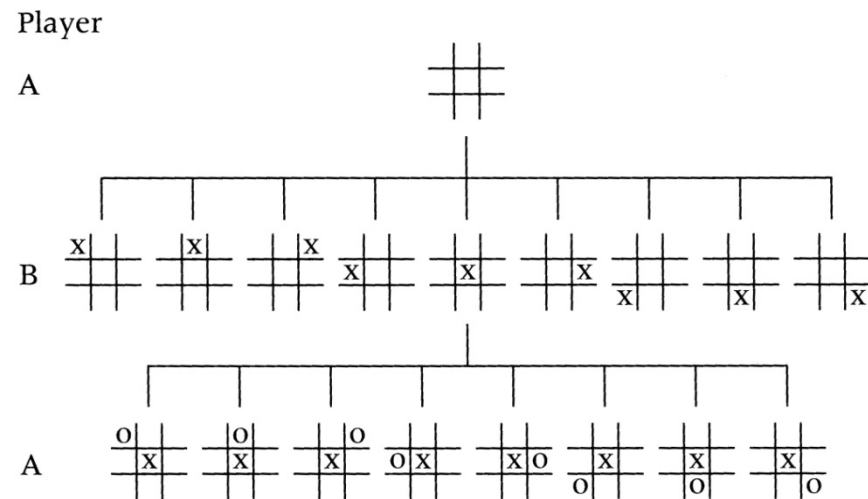
Stein Krogdahl

- Dagens program:
 - Første time:
 - Kap 23.5: Trær og strategier for spill med to spillere
 - Andre time, gjesteforelesning:
 - Rune Djurhuus: Om sjakkspillende programmer
 - (Ikke pensum, egne foiler legges ut)
- Neste uke:
 - Uavgjørbarhet

Kapittel 23.5: Spill, spilltrær og strategier

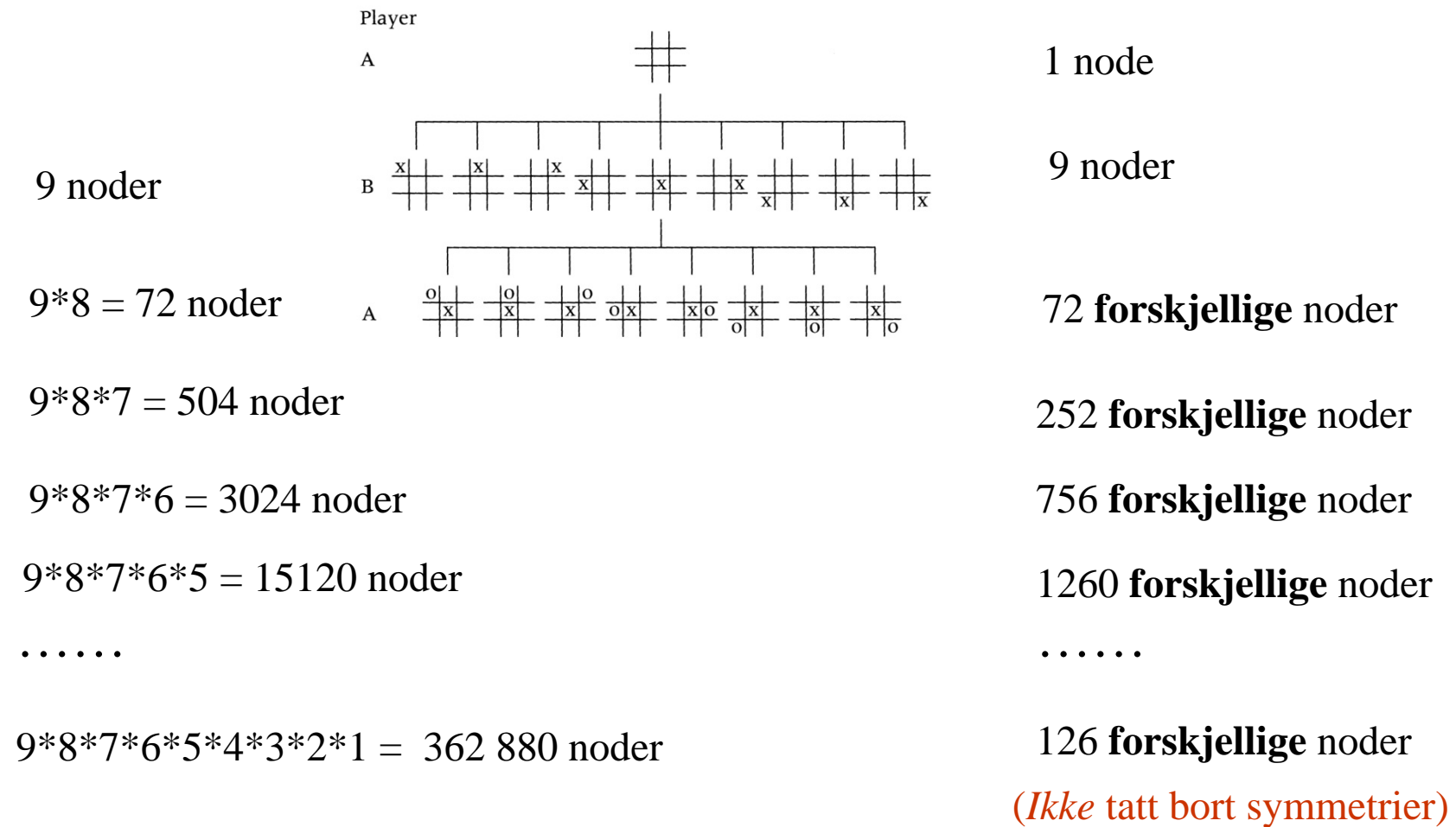
- Vi kunne lage et tre av avgjørelser for f.eks. "15-spillet" eller leting etter en beste rundtur for en "travelling salesman", og prøve å finne veien til en løsningsnode eller en beste vei.
 - Men da er det bare én "spiller", og man er ute etter en "nøytral" løsning
- Om det er to spillere som spiller mot hverandre blir saken annerledes. Det den ene ser som en bra situasjon ser den andre som dårlig.
 - Trærne av mulige spill blir ofte kjempestore. For sjakk estimert til 10^{100} noder.
 - Men selv for et enkelt spill som "tick-tack-toe" kan ved rett fram utlegging for $9! = 362\,880$ noder.

- Starten av et tre for tic-tac-toe blir slik:



- Spilleren som starter er A
 - Vi skal også gjøre alle vurderinger ut fra A synspunkt (inntil videre)

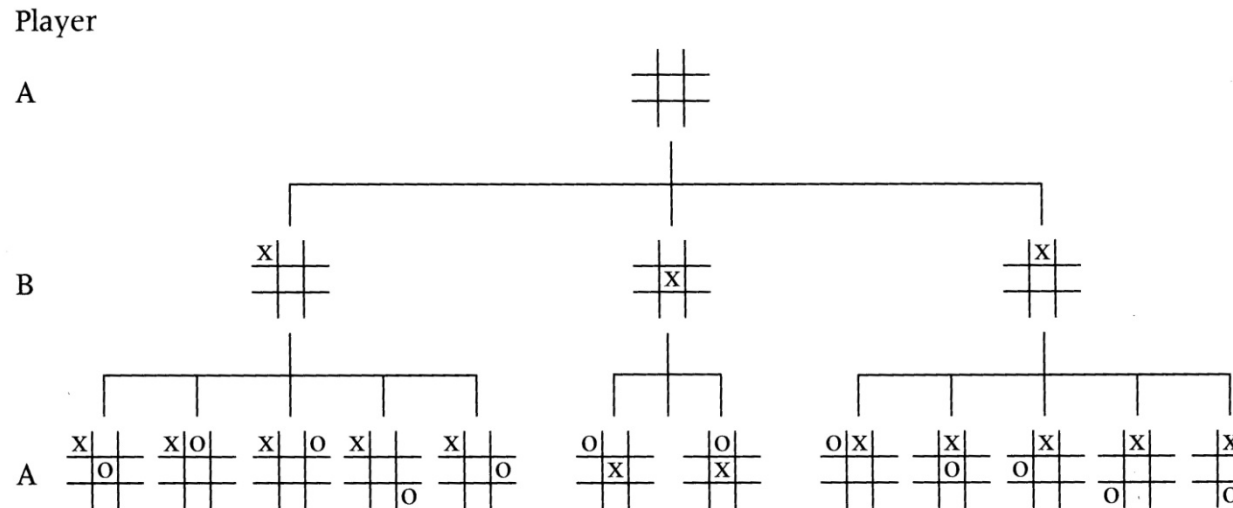
Treet blir fort stort



Altså, man kan i noen spill tjene masse på å kjenne igjen like noder, og ikke gjøre undersøkelsen om igjen (minner litt om dynamisk programmering).

Men dette krever som regel **MYE Plass!**

Ta bort symmetrier etc.



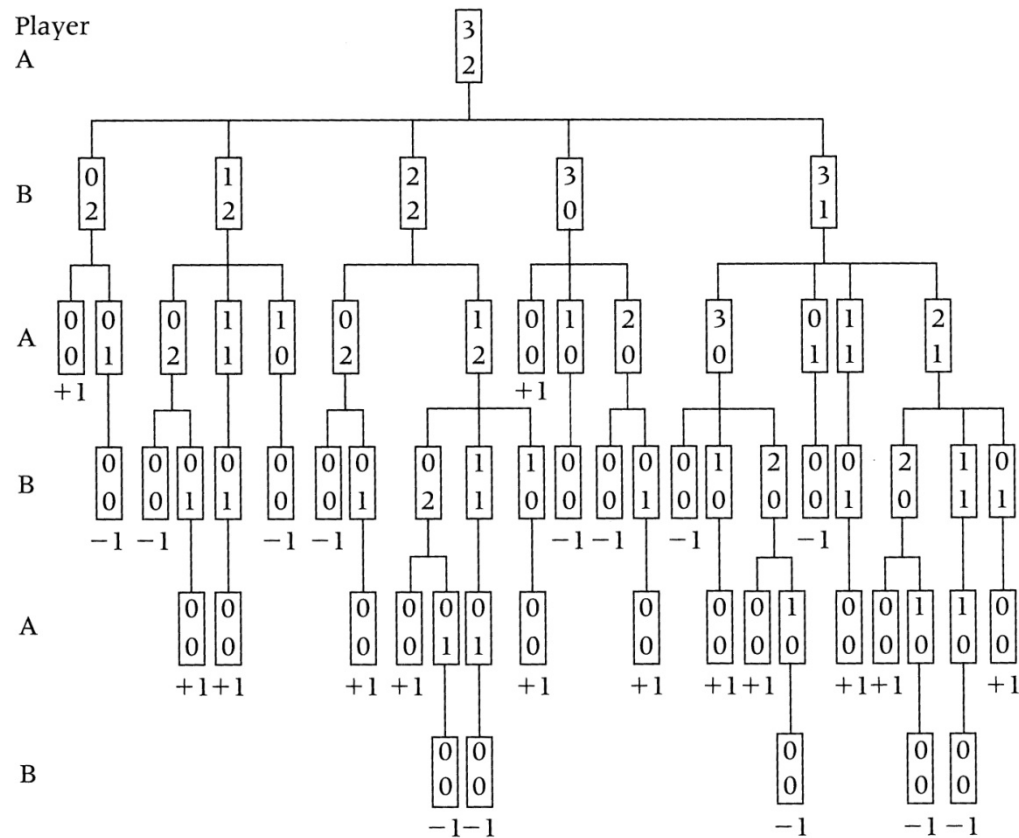
- Man kan også spare mye på symmetri-betraktninger:
 - Ta bare med de barna som er essensielt forskjellige
 - Altså her, slå sammen løsninger som er symmetriske av hverandre
 - Kan også legges inn letingen etter *samme situasjon*, på forrige foil. Reduserer lager plassen betraketlig i noen spill

Fystikk-spillet Nim

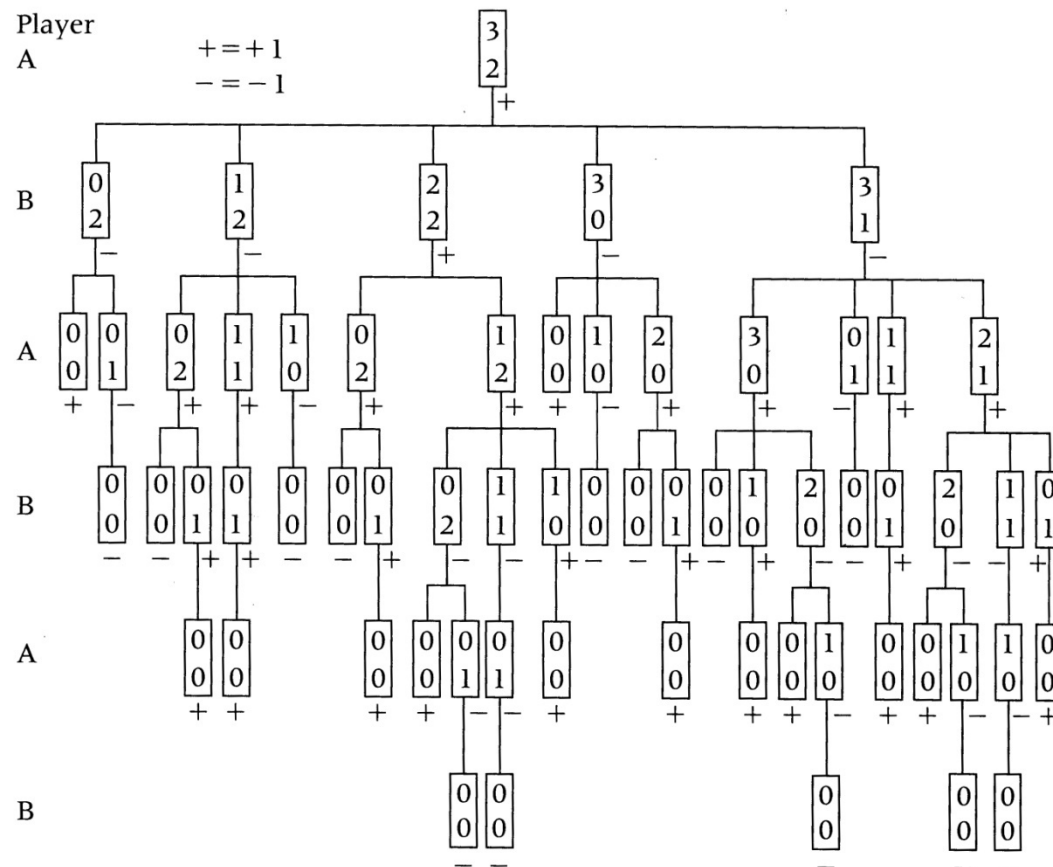
- Vi ser bare på "null-sum-spill". Altså, når den ene spilleren vinner noe så taper den andre tilsvarende.
- Enkleste variant: Det er to utfall, A vinner (+1) og B taper (-1), eller omvendt
- En del spill har også "uavgjort", for eksempel tic-tac-toe. Gies verdien 0.
- Spillet Nim:

- Start med to bunker med fyrstikker.
- Antall: m og n
- En spiller kan ta så mange man vil fra én bunke
- Den som tar siste fyrstikken har tapt.

- Her blir det aldri uavgjort
- Om vi starter med m=3 og n=2 er hele spilltreet tegnet ut her:
- Verdi for A satt på hver slutt-node

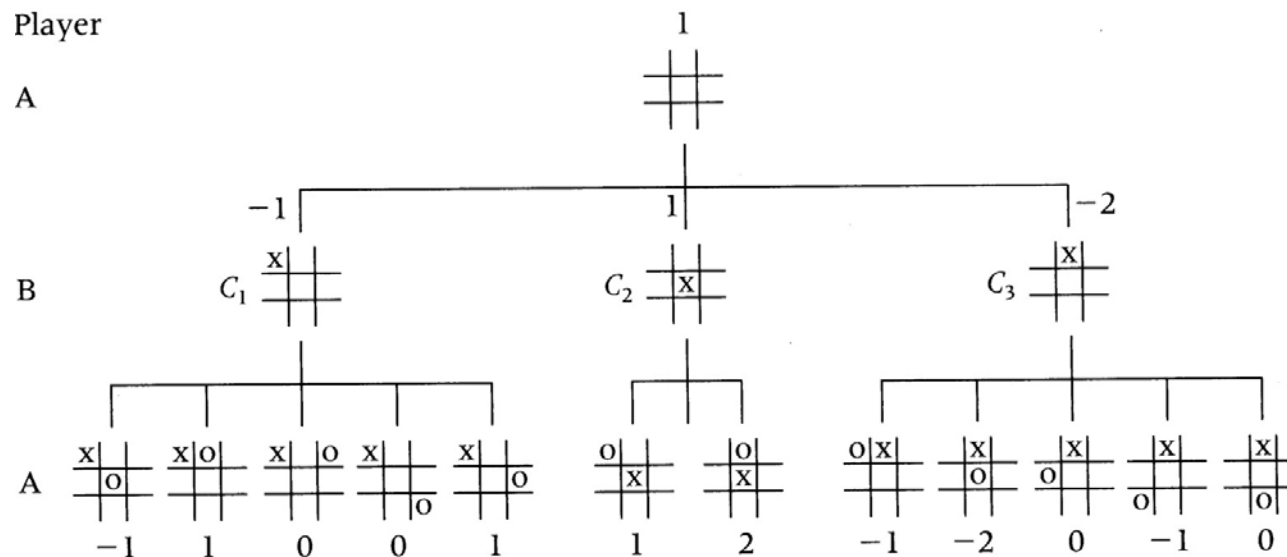


Hvordan finne strategi for å vinne, sett fra A?



- Vi angir hele tiden verdien av en node sett fra A
- A vil gjøre optimale trekk sett fra seg: Vil velge å gå til node med størst verdi
- A må regne med at B gjør optimale trekk sett fra seg.
- Siden verdiene noteres sett fra A, vil da B trekke til den *minste* subnoden
- For å få vite verdien av en node må vi derfor vite verdiene i alle subnodene
- Dette kan vi gjøre ved *dybde først* gjennomgang, og å beregne verdiene postfiks (på tilbakevei).

Oftest kan man ikke gå gjennom hele spilltreet



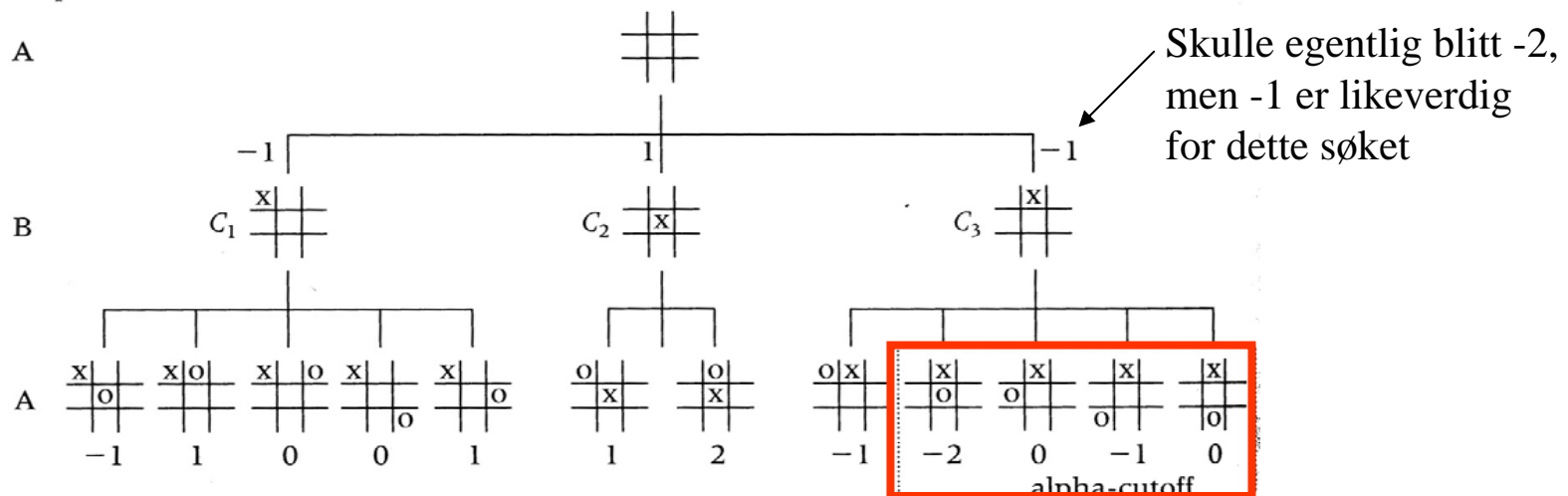
- Man må da f.eks. gå til en viss dybde, og her komme med et estimat (ut fra en gitt heuristikk-funksjon) for hvor god situasjonen er (sett fra A).
- I figuren over går vi til dybde 3 (eller 2, avhengig av hvordan man definerer dybde)
- Heuristikk her: Antall "vinnende" linjer for A minus det tilsvarende for B.
- A vil derfor her trekke til C_2 , som ser bra ut
- Men denne heuristikken er ikke lur når vi har gjort noen trekk. Tar ikke hensyn til at å vinne er bedre enn alle heuristikker. Gir derfor vinst-node verdi $+\infty = 9$
- Da får vi en god strategi for A, et stykke på vei.
- For: Om begge spiller perfekt, så blir det uavgjort i tic-tac-toe.
- F.eks. må slutt-situasjonen til høyre (uten vinner, uavgjort) få verdien 0:
- **NB: Vanskelig avveining mellom dybde og tidkrevende heuristikk!**

X	X	O
O	O	X
X	X	O

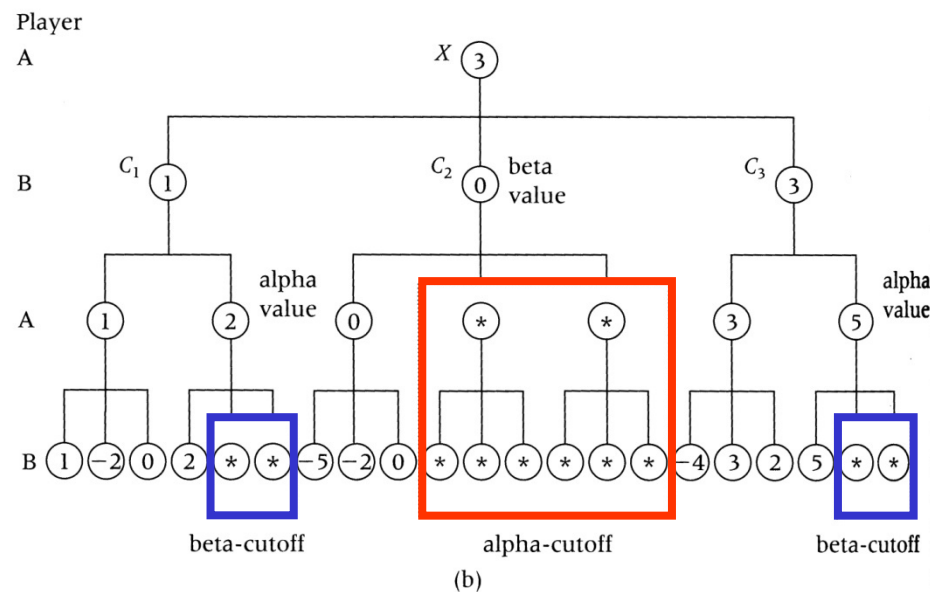
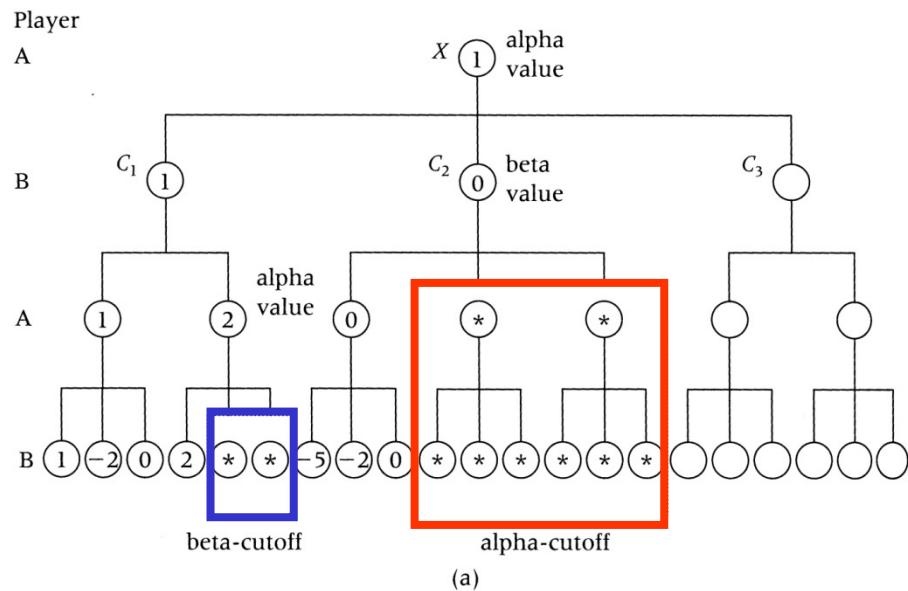
Alfa-beta-avskjæring

- Ved siden av å forsøke å redusere treet ved å se på symmetrier og/eller kjenne igjen noder som allerede er analysert, kan man gjøre "alfa-beta-avskjæring"
- Dette er som følger:
 - Jeg, A, vurderer trekk ett og ett trekk fra en gitt A-situasjon (øverst)
 - Jeg har allerede sett så gode trekk at jeg kan oppnå verdien u (av C_1 og C_2 , $u = 1$)
 - Jeg ser så på neste potensielle trekk som fører til situasjonen C_3 , og **ser at B herfra kan gjøre et trekk som fører til en god situasjon for B** (men dårlig for A), og som sett fra A har verdien $v = -1$. Da kan ikke verdien i C_3 bli større en v (husk at B minimaliserer!)
 - Om da $v < u$ så "gidder" ikke A gå videre å studere om B skulle ha et enda bedre trekk. Det kan ikke gjøre A mer lysten på trekket til C_3 . C_2 er helt sikkert bedre.

Player



Eksempler på alfa-beta-avskjæring



- Avskjæring som gjøres ut fra A's betrakning kalles alfa-avskjæring
- Samme betrakning som gjøres fra B-tilstander, men med minimalisering. Disse kalles beta-avskjæring
- Ved siden av ser vi eksempler på både alfa- og beta-avskjæring
- Når vi skal implementere alfa-beta-avskjæring kan det være greit å skifte synspunkt for hvert nivå.
 - Da skal det *alltid* maksimaliseres
 - Men man må skifte tegn på verdiene når man skifter nivå.
- En slik implementasjon står på neste foil

Alfa-beta-søk

```
real function ABNodeValue(  
  X, // Noden vi vil ha alfa/beta-verdien for. Barn: C[1], C[2], ... , C[k]  
  numLev, // Antall nivåer som står igjen  
  parentVal) // Alfa/beta-verdien fra forelder-noden (-LB fra foreldren)  
// returverdi: Den endelige alfa/beta-verdien for noden X  
{  
  real LB; // Løpende LowerBound for alfa/beta-verdi for denne noden  
  
  if <X er terminal-situasjon> then {return <verdien av denne sett fra X>;}  
  else if numLev = 0 then {return <estimat av kvaliteten av denne, sett fra X>;}  
  else {  
    LB := - ABNodeValue(C[1], NumLev-1,  $\infty$ );  
    for i := 2 to k do {  
      if LB >= parentValue then {return LB;}  
      else { LB := max(LB, -ABNodeVal(C[i], Numlev-1, -LB) ); }  
    }  
  }  
  return LB;  
}
```

Startkall: **situasjonsKvalitet := ABNodeValue(rotnoden, 10, - ∞)**

Trykkfeil i algoritmen

- Det er noen enkle trykkfeil i programmet på side 741 i boka (*ikke* rettet i ny utgave):
 - "AB" manglet i navnet på prosedyren, der den blir kalt rekursivt
 - Det mangler en sluttparentes på enden av linja der **max** kalles
- De *er* rettet i skissen på forrige foil