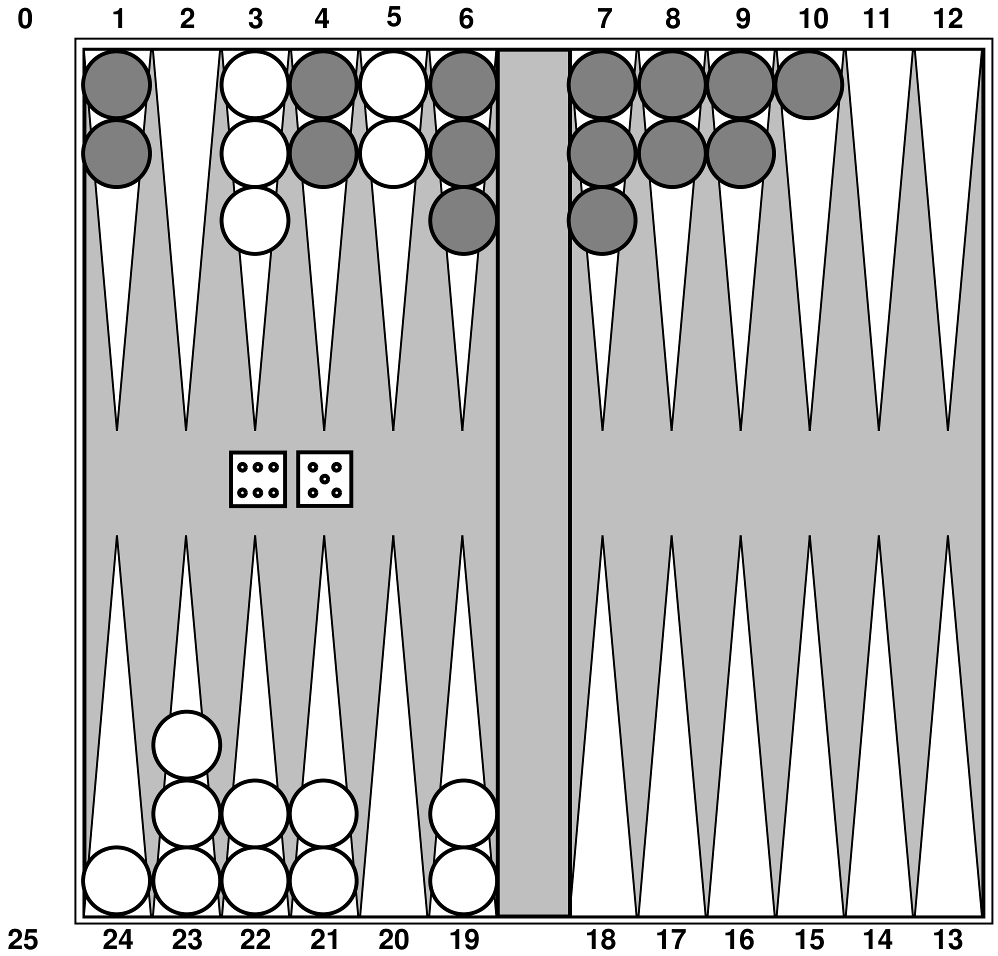
4.3.3. Sattuma ja strategia

Lisätään strategiapeliimme vielä satunnainen komponentti.

Esimerkiksi Backgammon-pelaaja aloittaa vuoronsa heittämällä 2 noppaa, joiden silmäluvut kertovat ne siirtomahdollisuudet mistä valita.



Kuva 51: Backgammon-pelitilanne [RN95, kuva 5.9].

Sovelletaan pelipuuajattelua tällaisten nopasta riippuvien strategisten valintojen tekemiseen.

Lisätään siis pelipuihimme *sattumasolmu (chance node)* (DICE) jokaiseen sattuma- eli nopanheittokohtaan:

Lapsina ovat kaikki vastaavan *satunnaistapahtuman (random event) mahdolliset vaihtoehdot*.

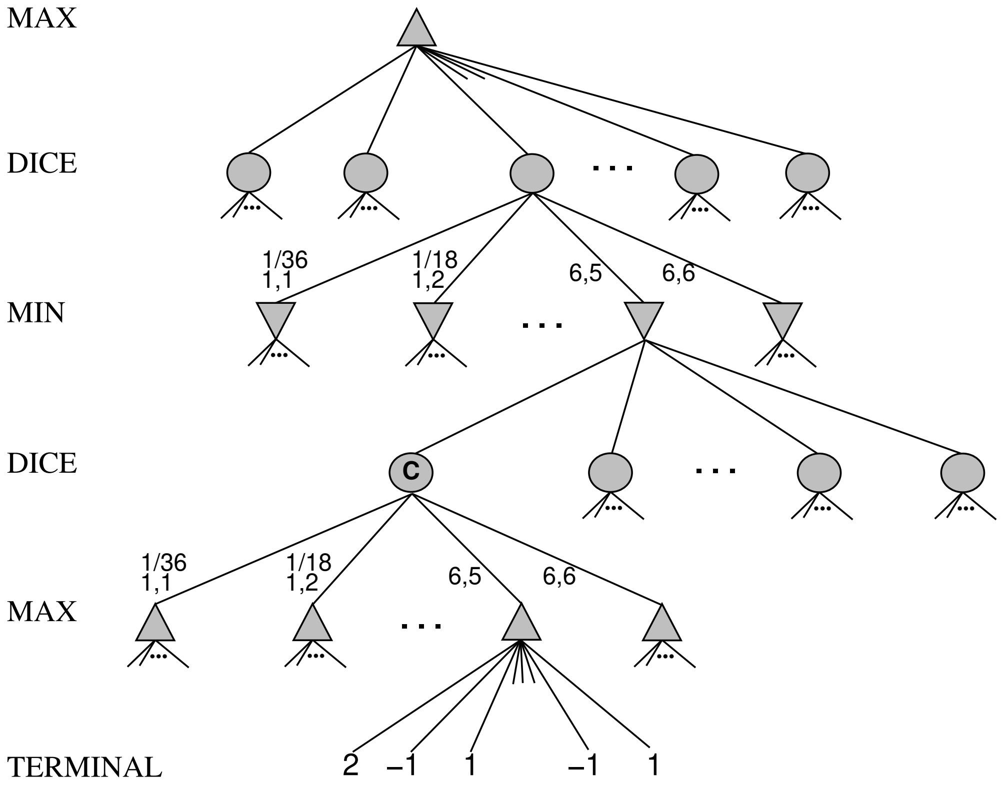
Backgammonissa kaikki 2 nopalla heitettävissä olevat (järjestämättömät) silmälukuparit.

Lapsesta alkaa pelipuu, jossa vaihtoehto on jo todellisuutta.

Haaralla on vastaavan vaihtoehdon *todennäköisyys*.

Backgammonissa kunkin silmälukuparin osuus kaikista silmälukupareista.

Sattumasolmut muodostavat *sattumatasoja* max- ja min-tasojen väliin.



Kuva 52: Backgammon-etsintäpuu [RN95, kuva 5.10].

Päästäksemme käyttämään luvun 4.3.1 minimax-periaatetta meidän on määriteltävä uudelleen se tapa, jolla tällaisen pelipuun solmut arvotetaan lapsista alkaen:

Tutut lehdet sekä $\max$- **ja** $\min$-**solmut** arvotetaan kuten aiemmin.

Sattumasolmu C **jonka lapset ovat** $\max$-**solmuja** ("max-sattumasolmu") arvotetaan *max-odotusarvoonsa*

$$\mathrm{expectimax}(C) = \sum_i P(d_i) c_i \tag{13}$$

missä
- indeksi i käy läpi solmun C lapset
- $P(d_i)$ on vaihtoehdon d_i todennäköisyys
- c_i on lapsesta i samalla menetelmällä laskettu arvo.

eli *tulosfunktion U odotusarvo*.

Backgammon-esimerkissä nopanheiton C hyvyys heittäjälleen $\max$ on se arvo, jonka hän voi *odottaa* saavansa
1. heittämällä nopat johonkin satunnaiseen silmälukupariin
2. pelaamalla mahdollisimman hyvin tästä saadusta parista alkaen.

Eli lasketaan lasten maksimin odotusarvoa.

$\min$-**sattumasolmu** arvotetaan sekin yhtälöllä (13) $\mathrm{expectimin}$-arvoonsa.

Nyt lapset i ovatkin $\min$-solmuja, ja niiden arvot c_i minimoituja.

Eli lasketaan lasten minimin odotusarvoa.

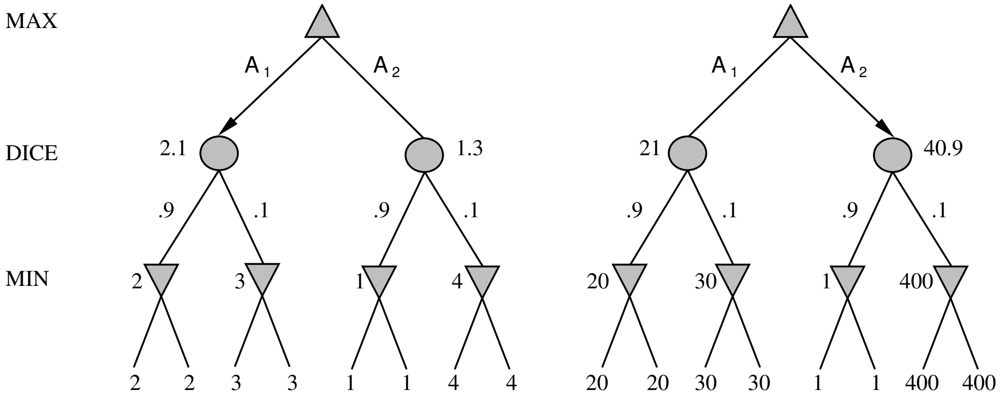
Kun minimax-algoritmia 10 muokataan näin, saadaan *expectiminimax*-algoritmi.

Tämän muokatun algoritmin nopeuttaminen katkaisuehdolla ja evaluointifunktiolla EVAL on vaikeampaa kuin luvussa 4.3.1:

- Yhtälön (13) laskemat odotusarvot riippuvat funktion EVAL *aidoista numeroarvoista*.

- Yhtälön (12) heikko järjestysehto ei siis enää riitä.

  Kuva 53 osoittaa tämän: kun lehtien arvoja muutetaan järjestys säilyttäen, niin valittu siirto muuttuu myös!



Kuva 53: Järjestys säilyy, suositus ei [RN95, kuva 5.11].

- Sen sijaan funktion EVAL tulisi olla nyt *positiivinen lineaarimuunnos* todellisesta tulosfunktiosta U.
- Sama vaikeus on kaikissa sellaisissa ongelmissa, jotka sisältävät epävarmuustekijöitä.

Olkoon n satunnaissolmujen erilaisten vaihtoehtojen lukumäärä. Backgammonissa $n = 21$.

Silloin expectiminimax-etsintä käy läpi sellaista puuta, jonka haarautumisaste on bn.

Etsinnän aikavaatimukseksi saadaan $O(b^m n^m)$ tutkittua solmua.

Jos etsintää nopeutetaan luvun 4.3.2 alfa-beta-karsinnalla, niin päädytään seuraavaan:

1. Milloin voimme karsia kuvan 52 solmulta C lapsia?
2. Sen isä on $\min$-solmu eli kiinnostunut solmun C arvon *ylärajasta*; sama ajatus näkyi jo kuvassa 49.

3. Tällainen yläraja voidaan selvittää laskematta kaikkia solmun C lapsia, jos tiedetään
   - solmun C lapsien todennäköisyydet $P(d_i)$ (kuten edellä)
   - rajat joiden sisällä funktion $\textsc{Eval}$ (tai U) arvot liikkuvat.

Todennäköisyyksien läsnäollessa karsinta ei kuitenkaan liene yhtä tehokasta kuin luvussa 4.3.2.

Yksityiskohdat sivuutetaan.

5. Tietämyksen esittäminen logiikoilla [RN95, osa III] [Nil98, osa III]

Kuka mitä häh? on filosofian peruskysymys. Nuori Nietzsche kiinnitti huomiota metafysiikan ja kielen yhtäläisyyksiin. Länsimaisen filosofian tai indoeurooppalaisen lauseen perusmuoto on subjekti-predikaatti-objekti. Tästä perustavasta 'kuka mitä häh' -rakenteesta syntyy se, mitä Nietzsche kutsui kansan metafysiikaksi – ajatus siitä, että todellisuus on luotu tämän saman kaavan mukaan. Toisin sanoen, jos ja kun kieli rakenteellisesti yllyttää meitä ilmaisemaan 'kuka mitä häh' -tyyppisiä lauseita, niin kohta olemme jo vakavalla naamalla tekemässä 'Jumala loi maailman' -tyyppistä kosmologiaa. Nietzsche iski kiilaan tähän kansan metafysiikkaan kuuluisalla esimerkillään salamasta. Kuka iskee? Salama. Mitä se iskee? Salaman.

Häh? Miksei riitä sanoa: salamoi!

Nietzschen huomautus synnyttää myös mielenkiintoisen kysymyksen siitä, onko suomi metafyysinen kieli. Vaikka suomen väitelauseet toki taipuvat luontevasti 'kuka mitä häh' -rakenteeseen, kielestämme puuttuu muodollinen subjekti. Emme sano 'it rains' tai 'Es regnet' vaan 'sataa'. On sääli, ettei Suomessa 1800-luvulla ollut yhtään kunnon romanttista kansallisuusmiestä, joka olisi kehittänyt tälle perustalle klassisen suomalaisen filosofian. *Tuomas Nevanlinna*

Jatketaan luvussa 2 aloitettua erilaisten toimijasuunnitelmien tarkastelua *tietämyspohjaisella toimijalla (knowledge-based agent)*:

```
function KB-AGENT(percept) returns an action
  static: KB, a knowledge base
          t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```

Algoritmi 12: Tietämyspohjainen toimija [RN95, kuva 6.1].

Muistina on nyt *tietämyskanta (knowledge base)* KB. Operaatiot:

Tell lisää kantaan uuden vastaavalla *tietämyksenesityskielellä (knowledge representation language)* kirjoitetun *lauseen (sentence)*.

Keskitytään luvussa 1.2 mainittuihin formaalin logiikan kieliin:

- lause- ja
- predikaattilogiikkaan.

Ask kysyy kannalta *seuraako (follows)* annettu tietämyksenesityskielen lause kannan nykyisestä sisällöstä.

Tietämyskannan on kyettävä omaehtoiseen *päättelyyn (inference)* vastatessaan annettuun **Ask**-kyselyyn.

Tietokantaan (database) on ohjelmoija viritellyt valmiiksi saantipolut tietojen hakuun ja yhdistelyyn.

Toimijan "syntyessä" sen tietämyskannan alkuarvona on suunnittelijan etukäteen antama (sovellusalueen) *taustatietämys (background knowledge)*.

Tavoitteena on *kuvaileva (deklaratiivinen)* toimijoiden ja järjestelmien suunnittelu- ja toteutustapa:

- Suunnittelija kertoo vain *mitä* päämäärää toimijan tulee tavoitella.
- Toimija päättää itse *miten* se ryhtyy sitä tavoittelemaan.

Tietämyspohjaista toimijaa voidaan tarkastella 3 eri abstraktiotasolla:

Tietämys- eli epistemologinen taso puhuu niistä *merkityksistä*, joita kantaan talletetuilla ja niistä johdetuilla seurauslauseilla meille toimijan ulkopuolisille tarkkailijoille on.

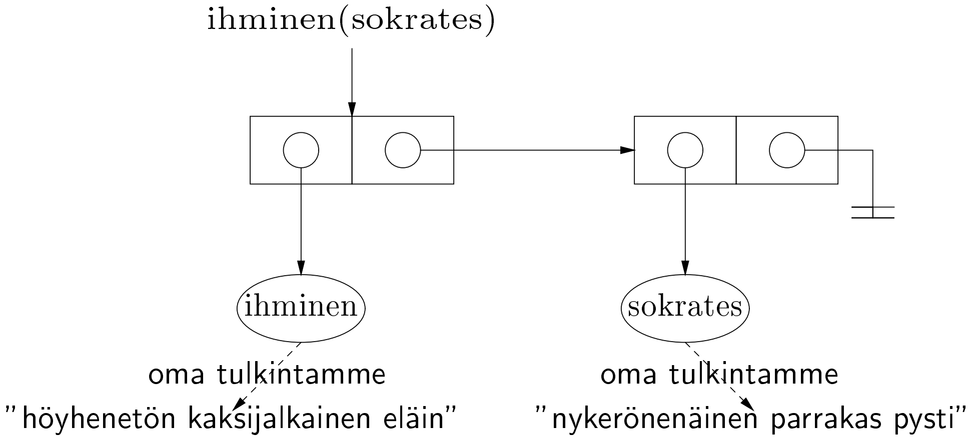
"Tuo ohjelma tietää, että Sokrates on ihminen."

Looginen taso puhuu niistä tietämyksenesityskielellä kirjoitetuista *lauseista*, jotka kantavat näitä merkityksiä.

"Tuon ohjelman kannassa on lause $\mathrm{ihminen}(\mathrm{sokrates})$."

Toteutustaso puhuu siitä, miten näitä lauseita sisältävä tietämyskanta on toteutettu.

"Tuota ohjelmaa suorittavan tietokoneen muistissa on kuvan 54 tietorakenne."



Kuva 54: Scheme-lista `(ihminen sokrates)`.

Tietämyksenesityskielessäkin erotellaan 3 puolta:

Kielioppi eli syntaksi (syntax) antaa ne säännöt, joiden mukaan kielen lauseet rakentuvat.

Esimerkiksi *kielioppisääntö (grammar rule)*

$$\text{Väitelause} \longrightarrow \text{Subjekti Predikaatti} \tag{14}$$

antaa (varsin puutteelliset) "rakennusohjeet" yksinkertaisten englanninkielisten väitelauseiden rakentamiseen.

Esimerkiksi

$$\texttt{computer thinks} \tag{15}$$

on säännön (14) mukainen lause: `"computer"` käy subjektiksi, `"thinks"` predikaatiksi.

Luvussa 1.4 mainittu Chomskyn teoria tarkasteli juuri tällaisia sääntöjärjestelmiä. Ne ovat tulleet myös tietojenkäsittelytieteen vakiotyökaluiksi.

Merkitysoppi eli semantiikka (semantics) kertoo, miten nämä syntaksin mukaiset lauseet voidaan ymmärtää.

Esimerkiksi sääntöön (14) liittyvä *tulkinta (interpretation)* sanoo:

- subjektin tulkinnan antama olio eli tietokone
- suorittaa predikaatin tulkinnan antamaa toimintoa eli ajattelee".

Näin teksti (15). . .

1. ensin *jäsentyy (parse)* kieliopin sääntöjä noudattavaksi lauseeksi
2. joka sitten tulkitaan vastaavaksi "ideaksi", "ajatukseksi", "merkitykseksi", "mielteeksi",. . .

eli vastaavan väitteen sisäiseksi esitykseksi.

Muistamme tämän sisäisen esityksen, ei sen ilmaisseen lauseen sanatarkkaa muotoa.

Monivaihemalli selittää Chomskyn tunnetun esimerkkilauseen

```
colourless green ideas sleep furiously.
```

1. Jäsennysvaihe 1 onnistuu, koska se on kieliopillisesti oikein.
2. Tulkintavaiheessa 2 muodostetaan vastine
   värittömät vihreät ideat nukkuvat raivokkaasti.
3. Lopuksi tämä vastine yritetään liittää aiempaan tietämykseen:
   - Subjektiin idea ei voi liittää väriin liittyviä määreitä — ikään kuin ideat eivät ole sellaista "tyyppiä" jolla olisi ominaisuus nimeltä 'väri'.
     Lisäksi värimääreet ovat keskenäänkin ristiriitaisia.
   - Predikaattiin nukkua ei kuulu raivokkuus.
   - Subjekti ei osaa tehdä sitä mitä predikaatti sanoo.

   Lauseen mielettömyys selviää siis vasta kun sen vastineen käsitteitä yrittää yhdistellä toisiinsa.

Kieli on *kompositionaalinen* jos sen lauseiden merkitys selviää yhdistelemällä lauseen osien merkityksiä.

Ainakin loogiset kielet ovat kompositionaalisia.

Kieliä tarkasteltaessa erotetaan toisistaan

objektikieli eli se kieli jota tarkastellaan
— Chomskyn esimerkissä englanti, jatkossa eri logiikat

metakieli eli se kieli jossa tarkastelu suoritetaan
— näissä luennoissa suomi.

Näin kierretään "eikö kielen tarkastelu edellytä sen ymmärtämistä":

Oletetaan että *objekti*kielen tarkastelija ymmärtää *meta*kielen.

Luvussa 1.2 mainittu loogikko Alfred Tarski esitti *semanttisen avoimmuuden vaatimuksen:*

> Objektikieli älköön puhuko metakieleen kuuluvista käsitteistä!

Tämä erityisesti siksi, että jos looginen objektikieli L pystyisi puhumaan sillä itsellään ilmaistujen lauseiden *totuudesta* — joka on metakielen käsite — niin objektikielessä L voitaisiin silloin ilmaista luvun 1.2 valehtelijan paradoksi "tämä lause on epätotta". . .

Silloin kaatuu yritys määritellä tyhjentävästi mitä objektikielen L lauseiden "totuus" oikein tarkoittaakaan.

Totuus onkin se keskeinen metakäsite, johon tässä esiteltävät *tavallisimmat* logiikat ja niiden pohjalta luodut tietämyksenesityskielet keskittyvät.

Muita mahdollisuuksia voisivat olla

- uskomukset ja niiden asteet
- tietämys ja sen kehittyminen
- todennäköisyydet
- päättelyn relevanssi
- . . .

Todistusteoria (proof theory) ratkoo seuraavaa *loogisten* kielten ongelmaa.

Syntaksi ja semantiikka kertovat, miten oikein muodostetut lauseet tunnistetaan ja ymmärretään vastaaviksi väittämiksi, joiden totuus tai epätotuus "nähdään suoraan".

Tietokone taas pystyy ainoastaan rakentelemaan muistiinsa kuvan 54 kaltaisia tietorakenteita; se ei näe mitään suoraan.

Miten tietokone voisi siis laskea mekaanisesti sille syötteenä annetun lauseen totuuteen liittyviä asioita, vaikkei se itse asiassa ymmärräkään syötettään?

Siten, että tietokoneeseen ohjelmoidaan sellaisia syötelauseiden *syntaktisia* laskutoimituksia, jotka eivät riko niiden *semantiikkaa*!