1.1 ÄLYKKÄÄT AGENTIT

- Agentti havainnoi toimintaympäristöään *sensorein* ja vaikuttaa siihen *aktuaattorein*
  - Ihmisen sensoreita ovat mm. silmät, korvat ja nenä sekä aktuaattoreita esim. kädet ja jalat
  - Robotin sensoreita voivat puolestaan olla kamerat ja laser-etäisyysmittarit, aktuaattoreita ovat eri moottorit
  - Ohjelmistoagentti puolestaan havainnoi näppäinpainalluksia, tiedostojen sisältöjä ja tietoliikennepaketteja
  - Sen toimintoja puolestaan ovat näytölle tai tiedostoon kirjoittaminen ja tietoliikennepakettien lähettäminen
- Oletamme agentin havainnoivan oman toimintansa, muttei välttämättä niiden vaikutusta
- Yleisesti ottaen agentin toiminta kullakin hetkellä voi riippua koko sen havaintohistoriasta
- *Agenttifunktio* kuvaa havaintojonon toiminnaksi
- Funktion kaikkien mahdollisten syöte-vaste -parien taulukko on täydellinen ulkoinen kuvaus agentista
- Sisäisesti agentin määrittää sen kontrolliohjelma

Agentin toiminnan arviointi

- Tavoittelemamme rationaalisen agentin pitäisi menestyä suorittamassaan tehtävässä
- Menestyksen arvioimiseksi tarvitsemme *tuloksellisuusmitan*
- Toiminnan rationaalisuus riippuu
  - tuloksellisuusmitasta joka määrää menestyksen,
  - agentin maailmaa koskevista ennakkotiedoista,
  - sen mahdollisista toiminnoista ja
  - havaintohistoriasta

Jokaisella havaintojonolla *rationaalinen agentti* valitsee toiminnan, joka odotusarvoisesti maksimoi tuloksellisuusmitan havaintohistorian ja ennakkotietojen valossa

- Filtteröinnin pitää luokitella sähköpostiviestit roskapostiksi tai oikeiksi viesteiksi
- Lääkärin pitää valita leikatako potilas vaiko ei
- Pelkkä oikein luokittelu ei ole paras mahdollinen tuloksellisuusmitta, koska oikeilla ja väärillä valinnoilla on eri painoarvot
- Tärkeän viestin siirtäminen roskapostikansioon on haitallisempaa kuin satunnaisesti läpipääsevät roskaviestit

True Positive	False positive
False negative	True negative

Toimintaympäristöjen ominaisuuksia

Toimintaympäristöjä voidaan luokitella ainakin seuraavien ominaisuuksien perusteella

- **Täysin vai osittain havainnoitava?**
  - Täysin havainnoitavassa maailmassa agentti saa sensoreiltaan kaiken toiminnan valintaan vaikuttavan relevantin tiedon
  - Näin ollen se ei tarvitse ylläpitää omaa käsitystään maailman tilasta
  - Ympäristö voi olla vain osittain havainnoitavissa perusluonteensa takia tai sensoreiden epätarkkuuden vuoksi
- **Deterministinen vai stokastinen?**
  - Jos maailman nykyinen tila ja agentin suorittama toiminto määräävät seuraavan tilan, on maailma deterministinen. Muuten se on stokastinen.
  - Deterministinenkin maailma voi vaikuttaa stokastiselta, jos se ei ole täysin havainnoitavissa
- **Episodinen vai sekventiaalinen?**
  - Episodisessa maailmassa edellisten episodien toiminnat eivät vaikuta tulevissa episodeissa
  - Sekventiaalisessa maailmassa puolestaan tehtävät päätökset vaikuttavat myös tuleviin päätöksiin

Toimintaympäristöt /2

- **Staattinen vai dynaaminen?**
  - Staattisessa maailmassa agentti voi pysähtyä pohtimaan toimintaansa ilman pelkoa asetelman muuttumisesta
  - Dynaamisessa maailmassa agentin on jatkuvasti tarkkailtava maailman tilaa
  - *Semidynaamisessa* ympäristössä maailma sinänsä ei muutu ajan kuluessa, mutta agenttia rangaistaan toiminnan suunnitteluun kuluvasta ajasta
- **Diskreetti vai jatkuva-arvoinen?**
  - Jako kohdistuu maailman tilaan, aikaan ja agentin havaintoihin ja toimintoihin
- **Yksi vai monta agenttia?**
  - Monen agentin maailmassa voidaan kilpailla tai tehdä yhteistyötä

Robotin kannalta "reaalimaailma" on …

- *Osittain havainnoitava*
  - Sensorit ovat epätäydellisiä ja havainnoivat vain lähiympäristöä (sama pätee myös ihmiselle)
- *Stokastinen*
  - Pyörät lipsuvat, akut tyhjenevät, osat hajoavat – koskaan ei ole varmaa, että aiottu toiminto toteutuu
- *Sekventiaalinen*
  - Toimintojen seuraukset muuttuvat ajan myötä ⇨ robotin on hallitava sekventiaalisia päätösongelmia ja kyettävä oppimaan
- *Dynaaminen*
  - Milloin pohtia, milloin toimia
- *Jatkuva/Ääretön*
  - Algoritmien on toimittava tässä ympäristössä, ei esimerkiksi äärellisessä diskreetissä hakuavaruudessa
- *Yhden/monen agentin maailma*
  - Riippuu siitä halutaanko muut oliot nähdä agentteina vai stokastisesti toimivina ympäristön osina

- Koska agentin havaintohistoria-vaste -parien taulukko kuvaa sen ulkoisen käyttäytymisen, niin periaatteessa agentin kontrolliohjelma voisi perustua tällaiseen taulukointiin
- Tietysti tämä on toimiva ratkaisu vain hyvin pienissä toimintaympäristöissä
- Realistisissa tapauksissa taulukointi ei ole käyttökelpoinen ratkaisu
- Agentin kontrolliohjelman on siis saatava aikaan haluttu toiminto kullakin havaintohistorialla ilman kaikkien vaihtoehtojen taulukointia

- Seuraavat perustyypit ovat yleisimmät ratkaisut tähän ongelmaan

Refleksiivinen agentti

- Yksinkertaisin kontrolliohjelma määrää agentin toimimaan tämänhetkisen havainnon perusteella jättäen aiemmat havainnot huomiotta
- Refleksit ovat käytössä hätäkeinoina niin ihmisillä kuin roboteillakin
- Refleksiivinen toiminta tuottaa oikeita päätöksiä vain jos toimintaympäristö on täysin havainnoitavissa
- Agentin sisäinen tilan perusteella voidaan valita kulloinkin voimassa oleva sääntöjoukko
- Täten pystytään ylläpitämään maailman mallia, joka kattaa osia siitä, mitä ei voida havainnoida

Tavoitehakuinen agentti

- Agentilla on havaintojensa lisäksi tiedossaan tavoite, johon se pyrkii
- Tavoite on jokin ympäristöä koskeva väittämä, joka tulisi toteuttaa
- Yhdistämällä tavoite ja tieto mahdollisten toimintojensa vaikutuksista voi agentti pyrkiä toteuttamaan tavoitteen
- Jos tavoitetta ei voida saavuttaa suoraan yhden toiminnon seurauksena, niin on suunniteltava sarja toimintoja sen saavuttamiseksi
- Voidaan käyttää joko hakualgoritmeja (luentoviikot 5–6) tai toiminnan suunnittelua (7)

Hyötyä tavoitteleva agentti

- Agentti voi saavuttaa tavoitteensa monella tapaa, ratkaisuilla voi olla laatueroja
- Tavoitteen asettaminen sellaisenaan ei riitä ilmaisemaan monimutkaisia asetelmia
- Jos maailman mahdollisille tiloille asetetaan järjestys *hyötyfunktiolla* (utility function), niin agentti voi pyrkiä parantamaan sen arvoa
- Hyötyfunktio kuvaa tilan (tai tilajonon) reaaliluvuksi
- Funktiota ei tarvitse olla eksplisiittisesti olemassa, jotta menetelmää voitaisiin käyttää

Oppivat agentit

- Agenttien koodaaminen käsin kaikkiin mahdollisiin tehtäviin vaikuttaa toivottamalta tehtävältä
- Jo Turing (1950) ehdotti oppimista menetelmäksi älykkäiden järjestelmien luomiseksi
- Agenttiteknologiassa oppimiskomponentin on oltava irrallinen varsinaisesta toimintakomponentista
- Palaamme oppimisen tekniikoihin yksityiskohtaisesti kurssin loppupuolella

2. LOGIIKKA, TIETÄMYS JA PÄÄTTELY

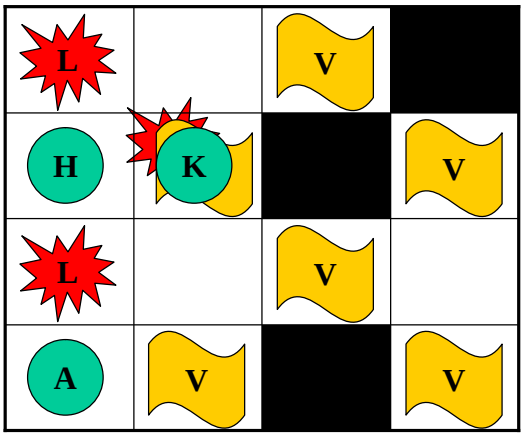
- Ryhdymme nyt tarkastelemaan *tietämyskannan* TK (knowledge base) omaavia agentteja
- TK:n avulla agentti pyrkii pitämään yllä tietoa vain osittain havainnoimastaan maailmasta ja tekemään päätelmiä sen tilasta
- Tietämyksen esittämiseen käytetään logiikkaa
- TK on joukko *lauseita* (sentence)
- Lähtötilanteessa agentin TK sisältää ennalta annetun *taustatietämyksen*
- Agentti tallettaa (Tell) TK:aan kaikki havaintonsa ja pyytää (Ask) TK:lta toimintaohjetta

- Toimintaohjeen valitseminen TK:n tietojen perusteella voi edellyttää laajamittaistakin päättelyä
- Myös tieto valitusta toiminnosta talletetaan TK:aan
- TK:n käyttö mahdollistaa agentin toiminnan määrittelemisen *tietämyksen tasolla* sen sijaan että antaisimme suoraan toteutusmääritelmän
- Agentin toiminnan määräämiseksi riittää antaa sen tiedot ja tavoite
- Tietämyksen tasolla toimiminen on deklaratiivinen lähestymistapa
- Proseduraalisessa tavassa puolestaan toiminnot koodataan suoraan

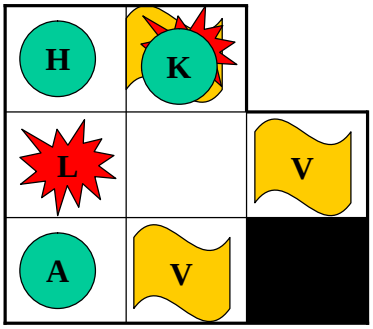
Esimerkkipeli

- Agentti toimii 4 × 4 ruudukkomaailmassa lähtien aina liikkeelle ruudusta [1,1] rintamasuunta oikealle
- Maailmassa on myös yksi (staattinen) hirviö ja kulta-aarre, lisäksi ruuduissa voi olla kuoppia
- Agentin mahdollisia toimintoja ovat
  - eteenpäin liikkuminen ruutu kerrallaan,
  - rintamasuunnan kääntäminen 90° oikealle tai vasemmalle,
  - kullan poimiminen ja
  - yhden ainoan nuolen ampuminen rintamasuuntaansa
- Agentti kuolee osuessaan samaan ruutuun (elävän) hirviön kanssa tai pudotessaan kuoppaan
- Peli päättyy agentin kuolemaan tai aarteen poimimiseen

- Hirviön ja kullan sijainti arvotaan tasaisen jakauman mukaan muiden ruutujen kuin [1,1] keskuudesta
- Muut ruudut kuin [1,1] sisältävät kuopan todennäköisyydellä 0.2
- Koordinaattiakselien suuntaisesti hirviön (H) viereisissä ruuduissa agentti (A) havainnoi löyhkän (L) ja kuopan (K) viereisissä ruuduissa viiman (V)
- Kulta havaitaan vasta samassa ruudussa,
  seinään törmääminen havaitaan ja
  hirviön kuolinhuuto kaikuu kaikkialle ruudukkomaailmaan



- Oheisessa lähtötilanteessa agentti voi päätellä molempien mahdollisten siirtymäruutujen ([1,2] ja [2,1]) olevan turvallisia, koska [1,1]:ssä ei ole löyhkää eikä viimaa
- Siirtyminen [2,1]:een saa agentin havainnoimaan viimaa, joten ruutu [2,2] tai [3,1] (tai molemmat) on kuoppa
- Tämän jälkeen vain ruutu [1,2] on turvallinen vierailematon ruutu
- Havainto ruudussa [1,2] on löyhkä, näin ollen
  - Hirviö ei oletuksen perusteella voi olla ruudussa [1,1]. Toisaalta se ei voi olla ruudussa [2,2] (koska muuten ruudussa [2,1] olisi havaittu löyhkä), joten hirviön on oltava ruudussa [1,3]
  - Ruudussa [2,2] ei voi olla kuoppaa, joten sen on oltava ruudussa [3,1]



2.1 LOGIIKKA

- TK:n muodostavien lauseiden syntaksi määräytyy valitun tietämyksenesityskielen perusteella
- Logiikassa kielen semantiikka määrää lauseiden *totuuden* suhteessa *malleihin* (mahdollisiin maailmoihin)
- Lause β *seuraa loogisesti* lauseesta α, $\alpha \models \beta$, joss kaikissa malleissa, joissa α on tosi, myös β on tosi
- Toisin sanoen: jos α on tosi, niin myös β:n on oltava tosi
- Tarkastellaan esimerkkipelin ruutuja [1,2], [2,2] ja [3,1] sekä sitä sisältävätkö ne kuopan
- Tieto on binäärinen, joten tälle tilanteelle on $2^3 = 8$ mahdollista mallia

Logiikka /2

- Koska ruudussa [1,1] ei tehty havaintoja ja ruudussa [2,1] havaittiin viimaa, niin TK:n malleja ovat ne joissa on kuoppa ruudussa [2,2] tai [3,1] tai molemmissa
- Nämä kolme mallia yhdessä sen kanssa, jossa kummassakaan mainitussa ruudussa ei ole kuoppaa ovat johtopäätöksen
  α_1 = "Ruudussa [1,2] ei ole kuoppaa" mallit
- Kaikissa malleissa, joissa TK on tosi myös α_1 on tosi, joten $TK \models \alpha_1$
- Sen sijaan johtopäätös α_2 = "Ruudussa [2,2] ei ole kuoppaa" on epätosi joissain malleissa, joissa TK on tosi
- Näin ollen $TK \not\models \alpha_2$

Logiikka /3

- Edellä kuvatulla tavalla toimiva looginen päättelyalgoritmi on mallintarkistus, koska kaikki mahdolliset mallit käydään läpi sen selvittämiseksi onko α tosi kaikissa malleissa, joissa TK on
- Jos päättelyalgoritmi i voi johtaa α:n TK:sta, niin merkitään $TK \vdash_i \alpha$
- Päättelyalgoritmi, joka johtaa vain loogisia seurauksia on eheä (sound, myös totuudensäilyttävä)
- Toinen päättelyalgoritmilta kaivattu ominaisuus on täydellisyys (completeness): se kykenee johtamaan kaikki loogiset seuraukset

2.2 PROPOSITIOLOGIIKKA

- Atomilauseita ovat propositiosymbolit P, Q, R, ...
- Kukin propositiosymboli vastaa väittämää, joka voi olla tosi tai epätosi
- Erikoisasemassa olevat symboli **T** on aina tosi ja **E** on aina epätosi
- Loogisilla konnektiiveilla muodostetaan monimutkaisempia propositioita yksinkertaisemmista
  - ¬ Negaatio. *Literaali* on joko (positiivinen) atomilause tai negatoitu atomilause
  - ∧ (ja) Konjunktion tekijöitä ovat *konjunktit*
  - ∨ (tai) Disjunktion osat ovat *disjunkteja*
  - ⇨ Implikaatiolla on etu- (premissi, ehto) ja takajäsen (johtopäätös)
  - ⇔ (joss) Ekvivalenssi

- BNF-esityksenä:

Lause → Atomilause | P-lause
Atomilause → **T** | **E** | Symboli
Symboli → P | Q | R | ...
P-lause → ¬ Lause | (Lause ∧ Lause) | (Lause ∨ Lause)
| (Lause ⇨ Lause) | (Lause ⇔ Lause)

- Sulkeiden vähentämiseksi konnektiivien presedenssiksi eli sitovuudeksi sovitaan järjestys $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$
- Täten esimerkiksi lause $\neg P \vee Q \wedge R \Rightarrow S$
  on ekvivalentti seuraavan lauseen kanssa
  $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

- Propositiologiikan semantiikka antaa säännöt lauseiden totuusarvon määräämiseksi mallien suhteen
- Propositiologiikassa malli kiinnittää kaikkien propositiosymbolien totuusarvon
- $\mathcal{M}_1 = \{ K_{1,2} = \mathbf{E}, K_{2,2} = \mathbf{E}, K_{3,1} = \mathbf{T} \}$

•Mv. lauseen totuusarvo voidaan laskea rekursiivisesti
- **E** on epätosi ja **T** tosi missä tahansa mallissa
- Mallin on asetettava propositiosymbolien totuusarvo
- P-lauseen arvo määräytyy *totuustaulun* mukaisesti

P	Q	¬ P	P ∧ Q	P ∨ Q	P ⇨ Q	P ⇔ Q
E	**E**	**T**	**E**	**E**	**T**	**T**
E	**T**	**T**	**E**	**T**	**T**	**E**
T	**E**	**E**	**E**	**T**	**E**	**E**
T	**T**	**E**	**T**	**T**	**T**	**T**

- Esim. lauseen $\neg K_{1,2} \wedge (K_{2,2} \vee K_{3,1})$ totuusarvo mallissa $\mathcal{M}_1$ on $\mathbf{T} \wedge (\mathbf{E} \vee \mathbf{T}) = \mathbf{T}$
- Looginen tietämyskanta TK, joka on muodostettu alkaen tyhjästä operaatioin $\mathrm{Tell}(TK, S_1)$, …, $\mathrm{Tell}(TK, S_n)$ on lauseiden konjunktio $TK = S_1 \wedge \ldots \wedge S_n$
- TK:taa voidaan siis käsitellä loogisena lauseena
- Seuraavassa propositiosymbolien tulkinta on:
  - $K_{i,j}$ on tosi jos ruudussa [i, j] on kuoppa
  - $V_{i,j}$ on tosi jos ruudussa [i, j] on viimaa

Tietämyskanta (1)

R_1: $\neg K_{1,1}$
R_2: $V_{1,1} \Leftrightarrow (K_{1,2} \vee K_{2,1})$
R_3: $V_{2,1} \Leftrightarrow (K_{1,1} \vee K_{2,2} \vee K_{3,1})$
R_4: $\neg V_{1,1}$
R_5: $V_{2,1}$

- Varsinainen tavoitteemme on päätellä päteekö $TK \models \alpha$ jollekin α
- Onko esim. $K_{2,2}$ looginen seuraus tietämyksestämme
- Esimerkissämme kahden ensimmäisen ruudun vierailun aikana relevantteja propositiosymboleja on 7 kpl, joten mahdollisia malleja on $2^7 = 128$ kpl
- Vain kolmessa näistä $TK = R_1 \wedge \ldots \wedge R_5$ on tosi
- Mallintarkistusalgoritmilla voimme todeta, että $\neg K_{1,2}$ on tosi kaikissa näissä kolmessa mallissa, joten ruudussa [1,2] ei ole kuoppaa
- Sen sijaan $K_{2,2}$ on tosi kahdessa ja epätosi yhdessä TK:n mallista, joten emme vielä kykene päättelemään ruudun [2,2] kuoppaisuutta

- Mallintarkistukseen perustuva algoritmi on eheä, koska se toteuttaa suoraan loogisen seuraamisen määritelmän
- Se on myös täydellinen, koska se toimii mille tahansa TK:lle ja α:lle aina pysähtyen — mahdollisia malleja on "vain" äärellinen määrä
- Jos TK ja α sisältävät yhteensä n symbolia, niin malleja on 2^n kpl
- Itse asiassa kaikkien tunnettujen propositiologiikan päättelyalgoritmien pahimman tapauksen aikavaativuus on eksponentiaalinen syötteen koon suhteen
- Propositiologiikan loogisen seuraamisen päätösongelma on co-NP-täydellinen