

## TEKOÄLY

### syksy 2003

- \* 4 ov, jatko-opintokelpoinen
- \* Luennot ti 12–14 S1, to 12–14 TB109  
2. 9. – 11. 12. (14 – 1 viikkoa)  
prof. Tapio Elomaa
- \* Harjoitustyöt 3 kpl  
tekn.yo. Minna Ruuska
- \* Viikkoharjoitukset  
ti ja to 10–12, pe 12–14 sali TC163  
tekn.yo. Teppo Soininen
- \* Koe 17. joulukuuta klo 9–12
- \* Arvostelu:  $30+15(+15)=45(+15)+5$
- \* {elomaa, ruuska2, teppos}@cs.tut.fi

1

## Viikkoharjoitukset

- \* Viikkoharjoitukseen osallistuminen on erittäin suositeltavaa
- \* Osallistumista saa "merkinnän". Valmiudesta esittää taululla vastaus kysymykseen saa kustakin merkinnän
- \* Kussakin harjoituksissa on n. 5 tehtävää  
⇒ kaikkiaan merkintöjä voi kerätä n.  $6 \cdot 12 = 72$  kappaletta

Osuus merkinnöistä	Lisäpisteet kurssin arvostelussa
20%	1
40%	2
60%	3
75%	4
85%	5

3

## Luentoaikataulu

1. **Johdanto** (I, 1)
  - \* Taustaa
  - \* Tekoälyn historia
  - \* Älykkäät agentit
2. **Logiikka, tietämys ja päättely** (III, 2–3)
3. **Ongelmanratkaisu ja hakualgoritmit** (II, 5–6)
4. **Toiminnan suunnittelu** (IV, 7)
5. **Epävarma tietämys ja päättely** (V, 8–10)
6. **Koneoppiminen** (VI, 10–12)
7. **Probabilistinen kielen käsittely** (VII, 13)
8. **Robottiikka** (VII, 14)

5

## Määritelmä?

- \* Älykkäälle koneelle ainoa tuntemamme vertailukohta olemme me itse
- \* Toisaalta vertailu *inhimilliseen* älykkyteen rajaa pois muut (paremmat) vaihtoehdot
- \* Ideaalista älykkyyttä kutsutaan *rationalisuudeksi*
- \* Toisaalta älykkyyttä voidaan tarkastella *ajattelun* tai *toiminnan* näkökulmasta
- \* Yhdistelminä saadaan neljä toisistaan poikkeavaa näkökulmaa tekoälyyn

7

## Harjoitustyöt

- \* Kurssilla on kolme pakollista harjoitustyötä. Kurssin suorittaminen edellyttää kaikkien töiden tekemistä
- \* kukin työ arvostellaan asteikolla 0–5.  $\geq 1$  on hyväksytty
- \* Lisäksi kunkin harjoitustyön parhaalle ratkaisulle annetaan 5 ylimääräistä pistettä (yhteensä 10)
- \* Logiikkaohjelmointi-harjoitustyö: miiniharava. Palautuspäivä 28. 10.
- \* Trimok-peliä pelaava agentti. Palautuspäivä 9. 12.
- \* Essee. Vapaavalintainen aihe oppijakson aihealueelta. Palautuspäivä 16. 12.

2

## Materiaali

Kurssin oppikirja on:

- \* S. Russell, P. Norvig: *Artificial Intelligence, A Modern Approach*, Second ed., Prentice Hall, 2003

Valmista monistetta ei ole, luentokalvot pannaan verkkoon luentojen tahtiin.

[www.cs.tut.fi/kurssit/8101905/](http://www.cs.tut.fi/kurssit/8101905/)

[www.cs.tut.fi/~elomaa/opetus/AI03/](http://www.cs.tut.fi/~elomaa/opetus/AI03/)

4

## 1. Johdanto

- \* Tekoäly on käsitteenä laaja, korkealentoinen ja ajan myötä muuttuva
- \* Nykyisin ehkä enemmän filosofien ja kognitiotieteilijöiden kenttää
- \* Tietojenkäsittelyllisesti tekoäly käsittää joukon tavoitteiltaan rajatumpia tutkimusaloja, jotka ovat jo eriytyneet melko etäälle toisistaan
- \* Yhteistä eri osa-alueille on pyrkimys koneiden "älykkyyden" nostamiseen
  - ◊ (Toisin katsoen: ohjelmistojen käytön helpottamiseen)
- \* Tuloksena käyttökelpoisia ohjelmistoja ja mekaanisen laskennan rajoja kartoittavaa teoriaa

6

## Turingin testi

- \* Englantilaisen matemaatikon Alan Turingin 1950 esittämä kriteeri koneen älykkyydelle on se, ettei ihminen erota konetta ihmisestä keskustellessaan sen kanssa tekstiviestein
- \* Eräs inhimillisen toiminnan testi
- \* Ns. *totaalisessa* Turingin testissä koneen on pystyttävä myös havainnoimaan ja manipuloimaan fyysisitä ympäristöään
- \* Aikarajoitettuja kilpailuja järjestetään vuosittain
- \* Parhaiten knoppitietoisia kilpailuttajia vastaan pärjäävät huijausohjelmat
- \* Inhimilliset asiantuntijat arvioidaan suurimmalla tn:llä koneiksi

8

## Kognitiivinen mallintaminen

- ★ Inhimillisen ajattelun tutkimus kuuluu psykologian ja kognitiotieteen kenttään
- ★ Myös tietysti neurofysiologia jne.
- ★ Vahva yhteys tekoälytutkimukseen, jos näkökulma on inhimillinen ajattelu
- ★ Ajattelun/mielen algoritmiikka ja tietämysrakenteet
- ★ Allen Newell ja Herbert Simon (1961): General Problem Solver (GPS)–päätelyaskelten ja inhimillisen päätelyn vertailu

9

## Rationaalisuus

- ★ Rationaalinen ajattelu on oleellisesti formaalin logiikan ja loogisen päätelyn tutkimista
- ★ (Yksinomaan) logiikkaan perustuvat menetelmät kärsivät laskennan vaativuudesta sekä epävarman tiedon esittämisen vaikeudesta
- ★ Rationaalisen toiminnan mallissa tarkastellaan *agentteja*
- ★ Agentti on toimija
- ★ Ohjelmistoagentin erottaa ohjelmasta esim. sen autonomisuus, ympäristön havainnointikyky, adaptiivisuus, pitkä elinikä, kyky omaksua muiden tavoitteita, tjms.

10

- ★ Rationaalinen agentti toimii parhaan mahdollisen lopputuloksen saavuttamiseksi havaintojensa ja tietämyksensä puitteissa
- ★ Epävarmuuden vallitessa pyritään maksimoimaan odotusarvoa
- ★ Rationaalinen toiminta pitkällä tähtäyksellä voi edellyttää epärationaalista vaikuttavaa toimintaa lyhyellä tähtäyksellä
- ★ Reaalimaailmassa täydellinen rationaalisuus ei useimmiten ole mahdollista (ajanpuutteen takia)

11

## Taustatieteitä

- ★ Filosofia ja matematiikka
  - ◊ Formaali päättely, mielen ja tiedon olemukset, jne.
  - ◊ Päättelysäännöt, laskettavuus ja sen vaativuus, epävarma päättely (todennäköisyyslaskenta), jne.
- ★ Taloustiede
  - ◊ päätös- ja peliteoria, operaatiotutkimus, jne.
- ★ Neurotieteet
- ★ Psykologia ja kognitiotiede
- ★ Tietotekniikka
- ★ Kontrolliteoria ja kybernetiikka
- ★ Lingvistiikka

12

## Tekoälyn historiaa

- ★ Ensimmäiseksi tekoälyjulkaisuksi voidaan katsoa McCulloch ja Pitts (1943)
  - ◊ Esitetään kuinka yksinkertaisten laskentaelementtien, *neuronien*, muodostamalla verkolla voidaan laskea loogiset konnektiivit (**and**, **or**, **not**, jne.)
  - ◊ Osoitetaan, että kaikki laskettavat funktiot voidaan laskea neuroverkolla
  - ◊ Vihjattiin verkkojen voivan oppia
- ★ Hebb (1949) antaa yksinkertaisen päivityssäännön neuroverkkojen opettamiseen

13

- ★ Turing (1950) esittelee testinsä, koneoppimisen, geneettiset algoritmit ja palauteoppimisen
- ★ 1956 John McCarthy järjesti aihealueesta kiinnostuneiden tutkijoiden tapaamisen, missä tutkimusalue sai nykyisen nimensä
- ★ Heti alusta lähtien keskeisiä yliopistoja olivat nykyisinkin tekoälytutkimuksen huipulla olevat CMU, MIT ja Stanford
- ★ McCarthy (1958) Lisp-ohjelmointikieli
- ★ 1950- ja 1960-luvuilla saavutettiin huomattavia edistysaskelia mikromaailmoissa (esim. palikkamaailma) tapahtuvassa toiminnassa

14

- ★ Robottiakin eteni: esim. SRL:n Shakey (1969)
- ★ Samoin neuroverkko tutkimus (Widrow & Hoff, Rosenblatting perceptron)
- ★ Vähitellen kuitenkin kävi selväksi, että mikromaailmojen menestys ei sellaisenaan skaalautunut ylöspäin. Se oli saavutettu ilman syvempää ymmärtämystä kohdeongelmasta ja laskentaintensiivisen menetelmin
- ★ Neuroverkko tutkimuksen tietojenkäsittelystä tappoi yli vuosikymmeneksi Minskyn ja Papertin todistus perceptronin heikosta esitysvoimasta (xor-funktio)
- ★ 1970-luvulla kehitettiin asiantuntijajärjestelmiä, joihin kerättiin yhden sovellusalueen syvää tietämystä

15

- ★ Asiantuntijajärjestelmillä saavutettiin monilla aloilla ihmisasiantuntijoita parempi suorituskyky ja niistä tuli tekoälytutkimuksen ensimmäinen kaupallinen menestystarina
- ★ Asiantuntijajärjestelmien kehittämisen osoitautui kuitenkin vaativaksi käsityöksi
- ★ Logiikkaohjelmointi eli kirkkaimman kukoistuksensa 1980-luvun puolessa välissä
- ★ Neuroverkko tutkimus nousi taas tutkimusalaaksi tietojenkäsittelystä 1980-luvun puolivälistä lähtien
- ★ Myös koneoppimistutkimuksen nousukausi ajoittuu 1980-luvulle

16

- ★ Samaten Bayes-verkkojen tutkimus lähti liikkeelle tuolloin
  - ◇ Ehkä toinen merkittävä kaupallinen sovellus Microsoftin vahvan näkyvyyden takia
- ★ Sittenmin samoja asioita on tutkittu myös tietämyksen muodostamisen nimikkeellä
- ★ Agenttitekniologia on nouseva trendi monella tietojenkäsittelyalueella
- ★ Viimeaikainen kehitys on ollut myös suuntautuminen analyttiseen tutkimukseen *ad hoc* tekniikkojen sijaan
  - ◇ Koneoppimisen teoreettiset mallit
  - ◇ Toiminnan suunnittelun uudet menetelmät
  - ◇ Peliteorian uusi kukoistus

17

minen ja tietoliikennepakettien lähettäminen

- ★ Oletamme agentin havainnoivan oman toimintansa, muttei välttämättä niiden vaikutusta
- ★ Yleisesti ottaen agentin toiminta kullakin hetkellä voi riippua kokosen havaintohistoriasta
- ★ *Agenttifunktio* kuvaa havaintojonon toiminnaksi
- ★ Funktion kaikkien mahdollisten syöte-vaste -parien taulukko on täydellinen ulkoinen kuvaus agentista
- ★ Sisäisesti agentin määrittää sen kontrolliohjelma

19

### Agentin toiminnan arviointi

- ★ Tavoittelemme 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

18

## 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

21

### ★ 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 rangastetaan toiminnan suunnitteluun kuluva ajasta

### ★ Diskreetti vai jatkuva-arvoinen?

- ◇ Jako kohdistuu maailman tilaan, aikaan ja agentin havaintoihin ja toimintoihin

23

## 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 laseretäisyysmittarit, aktuaattoreita ovat eri moottorit
  - ◇ Ohjelmistoagentti puolestaan havainnoi näppäinpainalluksia, tiedostojen sisältöjä ja tietoliikennepaketteja
  - ◇ Sen toimintoja puolestaan ovat näyttöille tai tiedostoon kirjoitta-

18

### ★ 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 toiminnot eivät vaikuta tulevissa episodeissa
- ◇ Sekventiaalisessa maailmassa puolestaan tehtävät päätökset vaikuttavat myös tuleviin päätöksiin

22

### ★ Yksi vai monta agenttia?

- ◇ Monen agentin maailmassa voidaan kilpailla tai tehdä yhteistyötä
- ★ Esimerkiksi robotin kannalta "reaali-maailma" 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

24

#### ◇ Sekventiaalinen

Toimintojen seuraukset muuttuvat ajan myötä ⇒ robotin on hallittava sekventiaalisia päätösongelmia ja kyettävä oppimaan

#### ◇ Dynaaminen

Milloin pohtia, milloin toimia

#### ◇ Jatkuva/Ääretön

Algoritmien on toimittava tässä ympäristössä, ei esimerkiksi äärellisessä diskreetissä hakuavuudessa

#### ◇ Yhden/monen agentin maailma

Riippuu siitä halutaanko muut olit nähdä agenteina vai stokastisesti toimivina ympäristön osina

25

## 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 maailmanmallia, joka kattaa osia siitä, mitä ei voida havainnoida

27

## Hyötyjä 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 *hyötyfunktio* (utility f.), niin agentti voi pyrkiä parantamaan sen arvoa
- ★ Hyötyftio kuvaa tilan (tai tilajonon) reaalityöväksi
- ★ Funktiota ei tarvitse olla eksplisiitistisesti olemassa, jotta menetelmää voitaisiin käyttää

29

## 2. Logiikka, tietämys ja päättely

- ★ Ryhdymme nyt tarkastelemaan *tietämuskannan* (knowledge base, KB) omaavia agenteja
- ★ KB:n avulla agentti pyrkii pitämään yllä tietoa vain osittain havainnoimastaan maailmasta ja tekemään päätelmiä sen tilasta
- ★ Tietämyksen esittämiseen käytetään logiikkaa
- ★ KB on joukko *lauseita* (sentence)
- ★ Lähtötilanteessa agentin KB sisältää ennalta-annetun *taustatietämyksen*
- ★ Agentti tallettaa (TELL) KB:hen kaik-

31

- ★ 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 perustyyppit ovat yleisimmät ratkaisut tähän ongelmaan

26

## Tavoitehakuinen agentti

- ★ Agentilla on havaintojensa lisäksi tiedossaan tavoite, johon se pyrkii
- ★ Tavoite on jokin ympäristöä koskeva väittäjä, 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)

28

## Oppivat agentit

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

30

- ki havaintonsa ja pyytää (ASK) KB:lta toimintaohjetta
- ★ Toimintaohjeen valitseminen KB:n tietojen perusteella voi edellyttää laajamittaistakin päättelyä
- ★ Myös tieto valitusta toiminnosta talletetaan KB:hen
- ★ KB:n käyttö mahdollistaa agentin toiminnan määrittämisen *tietämyksen tasolla* sen sijaan että antaisimme suoraan toteutusmääritelmän
- ★ Agentin toiminnan määräämiseksi riittää antaa sen tiedot ja tavoite
- ★ Tietämyksen tasolla toimiminen on *deklaratiiivinen* lähestymistapa; *proseduraalisessa* tavassa puolestaan toiminnot koodataan suoraan

32

## Esimerkipeli

L		V	
(H)	(K)		V
L		V	
(A)	V		V

- ★ 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

33

35

- ★ 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
- ★ Koordinaattiakselin 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 kuuluu kaikkialle ruudukkomailmaan

- ★ 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]

(H)	(K)		
(A)			V
	V		

34

36

## 2.1 Logiikka

- ★ KB:n muodostavien lauseiden syntaksi määräytyy valitun tietämyksen-esityskielen perusteella
- ★ Logiikassa kielen semantiikka määrittää lauseiden *totuuden* suhteessa *malleihin* (mahdollisiin maailmoihin)
- ★ Lause  $\beta$  seuraa loogisesti lauseesta  $\alpha$ ,
 
$$\alpha \models \beta,$$
 joss kaikissa malleissa, joissa  $\alpha$  on tosi, myös  $\beta$  on tosi
- ★ Toisin sanoen: jos  $\alpha$  on tosi, niin myös  $\beta$ :n on oltava tosi
- ★ Tarkastellaan esimerkipelin ruutuja [1,2], [2,2] ja [3,1] sekä sitä sisältävätkö ne kuopan

37

- ★ Tieto on binäärinen, joten tälle tilanteelle on  $2^3 = 8$  mahdollista mallia
- ★ Koska ruudussa [1,1] ei tehty havaintoja ja ruudussa [2,1] havaittiin viimaa, niin KB: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  $\alpha_1 = \text{"Ruudussa [1,2] ei ole kuoppaa"}$  mallit
- ★ Kaikissa malleissa, joissa KB on tosi myös  $\alpha_1$  on tosi, joten  $KB \models \alpha_1$
- ★ Sen sijaan johtopäätös  $\alpha_2 = \text{"Ruudussa [2,2] ei ole kuoppaa"}$  on epätosi joissain malleissa, joissa KB on tosi

38

- ★ Näin ollen  $KB \not\models \alpha_2$
- ★ Täten toimiva looginen päättelyalgoritmi on *mallintarkistus*, koska kaikki mahdolliset mallit käydään läpi sen selvittämiseksi onko  $\alpha$  tosi kaikissa malleissa, joissa KB on
- ★ Jos päättelyalgoritmi  $i$  voi johtaa  $\alpha$ :n KB:stä, niin merkitään  $KB \vdash_i \alpha$
- ★ Päättelyalgoritmi, joka johtaa vain loogisia seurauksia on *eheä* (sound, myös totuudensäilyttävä)
- ★ Toinen päättelyalgoritmita kaivattu ominaisuus on *täydellisyys* (completeness): se kykenee johtamaan kaikki loogiset seuraukset

39

## 2.2 Propositiologiikka

- ★ *Atomilauseita* ovat propositiesymbolit  $P, Q, R, \dots$
- ★ Kukin propositiesymboli vastaa väittämää, joka voi olla tosi tai epätosi
- ★ Erikoisasemassa olevat symbolit T on aina tosi ja E on aina epätosi
- ★ Loogisilla konnektiiveilla muodostetaan monimutkaisempia propositiota yksinkertaisemmista
  - ◇  $\neg$  *Negatio*. *Literaali* on joko (positiivinen) atomilause tai negatoitu atomilause
  - ◇  $\wedge$  (ja) *Konjunktion* tekijöitä ovat *konjunktit*
  - ◇  $\vee$  (tai) *Disjunktion* osat ovat *dis-*

40

junkteja

- ◇ ⇒ 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)

41

★ Mv. lauseen totuusarvo voidaan laskea rekursiivisesti

- ◇ E on epätosi ja T tosi missä tahansa mallissa
- ◇ Mallin on asetettava propositiesymbolien totuusarvo
- ◇ P-lauseen arvo määräytyy totuustaulun mukaisesti

P	Q	¬P	P ∧ Q	P ∨ Q
E	E	T	E	E
E	T	T	E	T
T	E	E	E	T
T	T	E	T	T

P	Q	P ⇒ Q	P ⇔ Q
E	E	T	T
E	T	T	E
T	E	E	E
T	T	T	T

43

## Tietämyskanta

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

- $R_6 : (V_{1,1} \Rightarrow (K_{1,2} \vee K_{2,1})) \wedge ((K_{1,2} \vee K_{2,1}) \Rightarrow V_{1,1})$
- $R_7 : ((K_{1,2} \vee K_{2,1}) \Rightarrow V_{1,1})$
- $R_8 : (\neg V_{1,1} \Rightarrow \neg (K_{1,2} \vee K_{2,1}))$
- $R_9 : \neg (K_{1,2} \vee K_{2,1})$
- $R_{10} : \neg K_{1,2} \wedge \neg K_{2,1}$
- $R_{11} : \neg V_{1,2}$
- $R_{12} : V_{1,2} \Leftrightarrow (K_{1,1} \vee K_{2,2} \vee K_{3,1})$
- $R_{13} : \neg K_{2,2}$
- $R_{14} : \neg K_{1,3}$
- $R_{15} : K_{1,1} \vee K_{2,2} \vee K_{3,1}$
- $R_{16} : K_{1,1} \vee K_{3,1}$
- $R_{17} : K_{3,1}$

45

- ★ Mallintarkistukseen perustuva algoritmi on eheä, koska se toteuttaa suoraan loogisen seuraamisen määritelmän
- ★ Se on myös täydellinen, koska se toimii mille tahansa KB:lle ja  $\alpha$ :lle aina pysähtyen — mahdollisia malleja on "vain" äärellinen määrä
- ★ Jos KB ja  $\alpha$  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

47

★ 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 määrittämiseksi mallien suhteen

★ Propositiologiikassa malli kiinnittää kaikkien propositiesymbolien totuusarvon

★  $\mathcal{M}_1 = \{ K_{1,2} = \mathbf{E}, K_{2,2} = \mathbf{E}, K_{3,1} = \mathbf{T} \}$

42

★ 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 tietämuskanta KB, joka on muodostettu alkaen tyhjästä operaatioin  $\text{TELL}(KB, S_1), \dots, \text{TELL}(KB, S_n)$  on lauseiden konjunktio  $KB = S_1 \wedge \dots \wedge S_n$

★ KB:tä voidaan siis käsitellä loogisena lauseena

★ Seuraavassa propositiesymbolien tulkinta on:

◇  $K_{i,j}$  on tosi jos ruudussa  $[i, j]$  on kuoppa

◇  $V_{i,j}$  on tosi jos ruudussa  $[i, j]$  on viimaa

44

★ Varsinainen tavoitteemme on päätellä pätee  $KB \models \alpha$  jollekin  $\alpha$

★ Onko esim.  $K_{2,2}$  looginen seuraus tietämuksestämme

★ Esimerkissämme kahden ensimmäisen ruudun vierailun aikana relevantteja propositiesymboleja on 7 kpl, joten mahdollisia malleja on  $2^7 = 128$  kpl

★ Vain kolmessa näistä  $KB = R_1 \wedge \dots \wedge R_5$  on tosi

★ Mallintarkistusalgoritilla 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 KB:n mallista, joten emme vielä kykene päättämään ruudun  $[2, 2]$  kuoppaisuutta

46

★ Lauseet  $\alpha$  ja  $\beta$  ovat loogisesti ekvivalentteja  $\alpha \equiv \beta$  jos ne ovat tosia samoissa malleissa:

$$\alpha \equiv \beta \text{ joss } \alpha \models \beta \text{ ja } \beta \models \alpha$$

★ Lause on *validi* eli *tautologia* jos se on tosi kaikissa malleissa

$$P \vee \neg P$$

★ Jokainen validi lause on loogisesti ekvivalentti arvon T kanssa

★ *Deduktioteoreema*: Mille tahansa lauseille  $\alpha$  ja  $\beta$ ,

$$\alpha \models \beta \text{ joss lause } (\alpha \Rightarrow \beta) \text{ on validi}$$

★ Mallintarkistusalgoritmin voi ajatella tarkistavan lauseen  $(KB \Rightarrow \alpha)$  validisuuden

48

★ Lause on *toteutuva* (satisfiable), jos on olemassa malli, jossa se on tosi

★ Esim. KB  $R_1 \wedge \dots \wedge R_5$  on toteutuva, koska on olemassa kolme sen *toteuttavaa* mallia

★ Propositiologiikan toteutuvuusongelma oli ensimmäinen NP-täydelliseksi todistettu ongelma (Cook 1971)

★  $\alpha$  on validi joss  $\neg\alpha$  ei ole toteutuva

★ Kääntäen:  $\alpha$  on toteutuva joss  $\neg\alpha$  ei ole validi

★ Ristiriitaan perustuva todistus:

$\alpha \models \beta$  joss lause  $(\alpha \wedge \neg\beta)$  ei ole toteutuva

49

$(\alpha \wedge \beta) \equiv (\beta \wedge \alpha)$	kommutatiivisuus
$(\alpha \vee \beta) \equiv (\beta \vee \alpha)$	kommutatiivisuus
$((\alpha \wedge \beta) \wedge \gamma) \equiv (\alpha \wedge (\beta \wedge \gamma))$	assosiatiivisuus
$((\alpha \vee \beta) \vee \gamma) \equiv (\alpha \vee (\beta \vee \gamma))$	assosiatiivisuus
$\neg(\neg\alpha) \equiv \alpha$	negation poisto
$(\alpha \Rightarrow \beta) \equiv (\neg\beta \Rightarrow \neg\alpha)$	kontraposio
$(\alpha \Rightarrow \beta) \equiv (\neg\alpha \vee \beta)$	implikaation poisto
$(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha))$	ekvivalenssin poisto
$\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$	de Morgan
$\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$	de Morgan
$(\alpha \wedge (\beta \vee \gamma)) \equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$	distriutiivisuus
$(\alpha \vee (\beta \wedge \gamma)) \equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma))$	distriutiivisuus

★ Pyritään *todistamaan* ettei ruudussa [1, 2] ole kuoppaa

★ Ekvivalenssin poistolla säännöstä  $R_2$  saadaan

$$(V_{1,1} \Rightarrow (K_{1,2} \vee K_{2,1})) \wedge ((K_{1,2} \vee K_{2,1}) \Rightarrow V_{1,1})$$

★ Soveltamalla tähän konjunktin poistoa seuraa  $((K_{1,2} \vee K_{2,1}) \Rightarrow V_{1,1})$

★ Kontrapositiolla

$$(\neg V_{1,1} \Rightarrow \neg(K_{1,2} \vee K_{2,1}))$$

★ Soveltamalla tähän ja sääntöön  $R_4$ :  $\neg V_{1,1}$  Modus Ponensia saadaan

$$\neg(K_{1,2} \vee K_{2,1})$$

★ Lopulta de Morganin säännöllä voidaan todeta

$$\neg K_{1,2} \wedge \neg K_{2,1}$$

52

★ Koska päättely propositiologiikassa on NP-täydellistä, niin pahimmassa tapauksessa todistuksen etsiminen on yhtä tehotonta kuin mallintarkastus

★ Käytännössä kuitenkin irrelevanttien tietojen huomiotta jättäminen usein tekee todistamisen tehokkaaksi

★ Esim. edellisen todistuksen maalisymboli  $K_{1,2}$  esiintyy vain KB:n säännössä  $R_2$ , jossa mainitut muut symbolit  $V_{1,1}$  ja  $K_{2,1}$  puolestaan esiintyvät sen lisäksi vain säännössä  $R_4$

★ Näin ollen säännöissä  $R_1$ ,  $R_3$  ja  $R_5$  mainittuja symboleja  $V_{2,1}$ ,  $K_{1,1}$ ,  $K_{2,2}$  ja  $K_{3,1}$  ei tarvitse huomioida

★ Logiikan *monotonisuus*: loogisten seurausten määrä voi vain kasvaa lisääntyneen tiedon myötä

53

## Resoluutio

★ Yhdistettynä täydelliseen hakualgoritmiin edellä esitetyt päättelysääntöt ovat täydellinen päättelyalgoritmi

★ Kuitenkin yhdenkin päättelysääntön poistaminen tuhoaa algoritmin täydellisyyden

★ Resoluutio on yksi ainoa päättelysääntö, joka yhdistettynä täydelliseen hakualgoritmiin antaa täydellisen päättelyalgoritmin

★ Kahden literaalin lauseisiin rajoitettu resoluutioaskel on

$$\frac{l_1 \vee l_2, \neg l_2 \vee l_3}{l_1 \vee l_3}$$

54

★ Kun agentti siirtyy ensimmäisen kerran tutkimaan ruutua [1, 2], niin se aistii löyhkän muttei viimaa

$$R_{11} : \neg V_{1,2}$$

$$R_{12} : V_{1,2} \Leftrightarrow (K_{1,1} \vee K_{2,2} \vee K_{1,3})$$

★ Vastaavalla päättelyllä kuin edellä, voimme todeta ettei ruuduissa [2, 2] ja [1, 3] ole kuoppaa

$$R_{13} : \neg K_{2,2} \text{ ja } R_{14} : \neg K_{1,3}$$

★ Ekvivalenssin poisto sovellettuna sääntöön  $R_3$  yhdessä Modus Ponensin ja sääntön  $R_5$  kanssa antaa

$$R_{15} : K_{1,1} \vee K_{2,2} \vee K_{3,1}$$

★ Resoluutioaskel sovellettuna  $R_{13}$  ja

55

$R_{15}$  sääntöihin antaa

$$R_{16} : K_{1,1} \vee K_{3,1}$$

★ Edelleen resoluutioaskel sovellettuna  $R_1$  ja  $R_{16}$  sääntöihin antaa  $R_{17} : K_{3,1}$

★ Edellä käytettiin yksikköresoluutiota

$$\frac{l_1 \vee \dots \vee l_k, m}{l_1 \vee \dots \vee l_{i-1} \vee l_{i+1} \vee \dots \vee l_k},$$

missä  $l_i$  ja  $m$  ovat komplementaariset literaalit

★ Yleisessä muodossa resoluutioaskel sallii mv. määrän komplementaarisia literaaleja

★ Resoluutioaskelen johtopäätöksestä tulee poistaa literaalien duplikaatit

$$\frac{(A \vee B), (A \vee \neg B)}{A}$$

56

## 2.3 Päättely propositiologiikassa

★ **Modus Ponens**

$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

★ **Konjunktin eliminointi**

$$\frac{\alpha \wedge \beta}{\alpha}$$

★ Nämä kaksi sääntöä ovat eheitä aina, ei ole tarvetta tehdä mallintarkistusta, vaan sääntöjä voidaan soveltaa suoraan

★ Seuraavista tunnetuista loogisista ekvivalensseista saadaan kustakin kaksi päättelysääntöä

★ Sen sijaan esim. Modus Ponensia ei voi kääntää

50

- ★ Resoluutio on ehea päättelysääntö ja se on myös täydellinen
- ★ Tosin resoluutiolla ei voi tiedosta, että  $A$  on tosi saada johtopäätöstä  $A \vee B$
- ★ Sen sijaan resoluutiolla voidaan tarkistaa onko  $A \vee B$  tosi
- ★ Osoittaaksemme, että  $KB \models \alpha$  osoitamme, että  $(KB \wedge \neg \alpha)$  on toteutumaton
- ★ Ensinnäkin  $(KB \wedge \neg \alpha)$  muunnetaan konjunkttiiviseen normaalimuotoon (CNF) ja resoluutioalgoritmia sovelletaan sen tekijöihin
- ★ Lopulta joko ei ole uusia lisättäviä tekijöitä, jolloin  $\alpha$  ei ole  $KB$ :n looginen seuraus, tai resoluutioaskel tuottaa tyhjän tekijän (= E), jolloin  $KB \models \alpha$

57

## Hornin lauseet

- ★ Tietämyksen esittämiseen riittävät usein implikaatiosäännöt, esim.

$$(Paikka_{1,1} \wedge Viima) \Rightarrow V_{1,1}$$

- ★ Säännön etujäsentä, joka muodostuu positiivisten literaalien konjunktiossa, kutsutaan lauseen *rungoksi* (body)
- ★ Myös takajäsen on negatoimaton ja sitä nimitetään lauseen *kärjeksi* (head)
- ★ Jos Hornin lauseella ei ole lainkaan runkoa, niin kyseessä on *fakta*
- ★ Hornin lause  $P_1 \wedge \dots \wedge P_n \Rightarrow Q$  on loogisesti ekvivalentti disjunktion  $(\neg P_1 \vee \dots \vee \neg P_n \vee Q)$  kanssa. Literaaleista siis korkeintaan yhden sallitaan olla positiivinen propositio

58

- ★ Päättely Hornin lauseilla voi perustua joko *eteen- tai taaksepäin ketjutukseen*
- ★ Molemmat päättelysuunnat ovat luontevia
- ★ Loogisen seuraamisen tutkimisen aikavaativuus Hornin lauseilla on vain lineaarista tietämyskannan koon suhteen
- ★ Eteenpäin ketjutus (*datalähtöinen päättely*) tutkii minkä kaikkien sääntöjen runko toteutuu tunnettujen faktojen perusteella ja lisää niiden kärjet uusiksi faktoiksi tietämyskantaan
- ★ Kyseessä on siis toistuva Modus Ponens -säännön soveltaminen

59

- ★ Prosessi jatkuu kunnes annettu kysely  $q$  on lisätty faktojen joukkoon tai kunnes uusia päätelmiä ei enää voida tehdä
- ★ Eteenpäin ketjutus on sekä ehea että täydellinen päättelyalgoritmi
- ★ Taaksepäin ketjutus on *tavoiteohjattua päättelyä* (goal-directed)
- ★ Annetun kyselyn  $q$  totuus pyritään toteamaan tarkastelemalla sellaisia sääntöjä, joiden kärki  $q$  on
- ★ Taaksepäin ketjuttaen tutkitaan sääntöjen runkojen konjunktien totuutta
- ★ Taaksepäin ketjutus käsittelee vain relevantteja faktoja kun taas eteenpäin ketjutus tuottaa sokeasti kaikki

60

## 2.4 Propositiologiikan riittämättömyys

- ★ Jo äärimmäisen yksinkertaisessa peliesimerkissämme propositiologiikan ilmaisuvoima osoittautuu riittämättömäksi
- ★ Tietämyskannan alustamiseksi pelin säännöillä meidän on jokaiselle ruudulle  $[x, y]$  annettava sääntö viiman tuntemuksesta (sekä vastaava sääntö löyhkän aistimukselle)

$$\forall x, y \Leftrightarrow (K_{x,y+1} \vee K_{x,y-1} \vee K_{x+1,y} \vee K_{x-1,y})$$

- ★ Hirviöitä on ainakin yksi

$$H_{1,1} \vee H_{1,2} \vee \dots \vee H_{4,3} \vee H_{4,4}$$

ja toisaalta korkeintaan yksi, joka voidaan ilmaista sääntöparein

$$\neg H_{1,1} \vee \neg H_{1,2}$$

- ★ Kun  $KB$ :hen vielä lisätään säännöt  $\neg K_{1,1}$  ja  $\neg H_{1,1}$ , niin pelkästään näihin yksinkertaisiin sääntöihin on tarvittu  $2 + 2 \cdot 16 + 1 + 120 = 155$  sääntöä ja eri symboleita on käytössä 64 kappaletta
- ★ Mallintarkastuksen tulisi käydä läpi  $2^{64} \approx 1.8 \times 10^{19}$  mahdollista mallia
- ★ Tehokkaammilla päättelyalgoritmeilla tätäkin esitystä toki voidaan käyttää
- ★ Lisäksi  $KB$  ei vielä suinkaan sisällä pelin kaikkia sääntöjä (nuoli, kulta, seinät) eikä tietoa agentin mahdollisista toiminnoista (rintamasuunta, askelet eteenpäin), joten sitä ei voi käyttää toimintojen valintaan
- ★ Raskautensa lisäksi propositiologiikka on epätydyttävä ratkaisu

62

## 2.5 Predikaattilogiikka

Lause	→	Atomilause
		(Lause Konnektiivi Lause)
		Kvanttori M-lista Lause
		$\neg$ Lause
Atomilause	→	Predikaatti (T-lista)
		Termi = Termi
T-lista	→	Termi   Termi, T-lista
Termi	→	Funktio (T-lista)
		Vakio
		Muuttuja
Konnektiivi	→	$\wedge$   $\vee$   $\Rightarrow$   $\Leftrightarrow$
Kvanttori	→	$\forall$   $\exists$

63

M-lista	→	Muuttuja   Muuttuja, M-lista
Vakio	→	$A$   $X_1$   <i>Jussi</i>   <i>Meeri</i>   ...
Muuttuja	→	$a$   $x$   $s$   ...
Predikaatti	→	Teekkari   On-väriiltään   Äiti
Funktio	→	Isä   Vasen-jalka   ...

- ★ (1. kertaluvun) predikaattilogiikan maailmassa on *olioita*, joilla on *ominaisuuksia*, ja joiden välillä vallitsee *suhteita*
  - ★ Vakiosymbolit viittaavat olioihin, predikaattisymbolit suhteisiin eli  $n$ -paikkaisiin relaatioihin ja ominaisuudet ovat yksipaikkaisia relaatioita
- Teekkari(Jussi), Äiti(Jussi, Meeri)

64

\* Funktio on totaalinen kuvaus, joka liittää argumenttiensa järjestettyyn kokoelmaan täsmälleen yhden tuloksen

\* Kvanttorit antavat mahdollisuuden universaaliin ja eksistentiaaliseen kvantifointiin

$$\forall y \text{ Teekkari}(y) \Rightarrow \text{Tupsuilee}(y)$$

$$\exists x \text{ Isä}(\text{Jussi}) = x$$

\* Semantiikan määrittämiseksi syntaktiset merkinnät (vakio-, predikaatti- ja funktiosymbolit) on sidottava *tulkinnalla* (interpretation) (reaali)maailman vastineisiinsa

\* Maailma yhdessä merkintöjen tulkinnan kanssa muodostavat predikaattilogiikan mallin

65

\* Koska  $\forall$  on itse asiassa konjunktio yli maailman olioiden ja  $\exists$  disjunktio, niin ne noudattavat de Morganin sääntöjä

$$\forall x \neg \phi(x) \equiv \neg \exists x \phi(x)$$

$$\neg \forall x \phi(x) \equiv \exists x \neg \phi(x)$$

$$\forall x \phi(x) \equiv \neg \exists x \neg \phi(x)$$

$$\neg \forall x \neg \phi(x) \equiv \exists x \phi(x)$$

\* Yhtäsuuruus on erikoisasemassa oleva relaatio, jonka tulkintaa ei voi asettaa

\* Termit  $t_1$  ja  $t_2$  ovat tässä relaatiossa vain jos niiden tulkinta on sama olio

67

## Sukulaisuussuhteita

$$\forall m, c \text{ Mother}(c) = m$$

$$\Leftrightarrow \text{Female}(m) \wedge \text{Parent}(m, c).$$

$$\forall w, h \text{ Husband}(h, w) \Leftrightarrow \text{Male}(h) \wedge \text{Spouse}(h, w).$$

$$\forall x \text{ Male}(x) \Leftrightarrow \neg \text{Female}(x).$$

$$\forall p, c \text{ Parent}(p, c) \Leftrightarrow \text{Child}(c, p).$$

$$\forall g, c \text{ Grandparent}(g, c)$$

$$\Leftrightarrow \exists p \text{ Parent}(g, p) \wedge \text{Parent}(p, c).$$

$$\forall x, y \text{ Sibling}(x, y)$$

$$\Leftrightarrow x \neq y \wedge \exists p \text{ Parent}(p, x) \wedge \text{Parent}(p, y).$$

66

## Peanon aksioomat

\* Määrittelevät *luonnolliset luvut* ja yhteenlaskun yhden vakiosymbolin 0 ja seuraajafunktion  $S$  avulla

$$\text{NatNum}(0).$$

$$\forall n \text{ NatNum}(n) \Rightarrow \text{NatNum}(S(n)).$$

\* Luonnolliset luvut siis ovat  $0, S(0), S(S(0)), \dots$

\* Seuraajaftion ominaisuuksia

$$\forall n 0 \neq S(n).$$

$$\forall m, n m \neq n \Rightarrow S(m) \neq S(n).$$

\* Yhteenlasku ja seuraajaftio

$$\forall m \text{ NatNum}(m) \Rightarrow +(m, 0) = m.$$

$$\forall m, n \text{ NatNum}(m) \wedge \text{NatNum}(n)$$

$$\Rightarrow +(S(m), n) = S(+(m, n)).$$

69

## Joukot

\* Joukkojen määrittelemiseksi käytämme

◊ vakiosymbolia  $\{ \}$ , joka viittaa tyhjään joukkoon

◊ yksipaikkaista predikaattia  $Set$ , joka on tosi joukoille

◊ kaksipaikkaisia predikaatteja  $x \in s$  ja  $s_1 \subseteq s_2$

◊ Binäärisiä funktioita ovat  $s_1 \cup s_2$ ,  $s_1 \cap s_2$  ja  $\{ x \mid s \}$ , joka viittaa siihen joukkoon, joka saadaan kun alkio  $x$  lisätään joukkoon  $s$

\* Joukkoja ovat vain tyhjä joukko ja ne, jotka on muodostettu lisäämällä alkioita joukkoihin

$$\forall s \text{ Set}(s) \Leftrightarrow s = \{ \} \vee$$

$$\exists x, s_2 \text{ Set}(s_2) \wedge s = \{ x \mid s_2 \}.$$

70

\* Tyhjällä joukolla ei ole alkioita

$$\neg \exists x, s \{ x \mid s \} = \{ \}.$$

\* Samaa alkioita ei voi lisätä joukkoon kahdesti  $\forall x, s x \in s \Leftrightarrow s = \{ x \mid s \}$ .

\* Joukossa on vain siihen lisätyt alkiot

$$\forall x, s x \in s \Leftrightarrow [\exists y, s_2$$

$$(s = \{ y \mid s_2 \} \wedge (x = y \vee x \in s_2))].$$

\* Joukkojen sisältyvyys

$$\forall s_1, s_2 s_1 \subseteq s_2 \Leftrightarrow (\forall x x \in s_1 \Rightarrow x \in s_2).$$

\* Joukkojen ekvivalenssi

$$\forall s_1, s_2 (s_1 = s_2) \Leftrightarrow (s_1 \subseteq s_2 \wedge s_2 \subseteq s_1).$$

$$\forall x, s_1, s_2 x \in (s_1 \cap s_2) \Leftrightarrow (x \in s_1 \wedge x \in s_2).$$

$$\forall x, s_1, s_2 x \in (s_1 \cup s_2) \Leftrightarrow (x \in s_1 \vee x \in s_2).$$

71

## 2.6 Päätely

### predikaattilogiikassa

\* Jos predikaattilogiikan lauseista eliminoidaan kvanttorit, niin voidaan siirtyä propositiologiikan lauseisiin ja käyttää tuttuja päätelysääntöjä

\* Universaalikvantioidun lauseen  $\forall v \alpha$  muuttuja  $v$  voidaan korvata millä tahansa *perustermillä* (ground term), joka ei sisällä muuttujia

\* *Sijoitus* eli *substituutio*  $\theta$  on sidontalista, jossa kullekin muuttujalle annetaan sen korvaava perustermi

\* Merk.  $\alpha(\theta)$  on lause, joka saadaan kun sijoitusta  $\theta$  sovelletaan lauseeseen  $\alpha$

72

- ★ Universaalikvanttorin eliminoimiseksi voimme päätellä

$$\frac{\forall v \alpha}{\alpha(\{v/g\})}$$

missä  $v$  on mikä tahansa muuttuja ja  $g$  mv. perustermi

- ★ Eksistenssivanttorin eliminoimisessa muuttuja  $v$  korvataan *Skolem vakiolla*  $k$ , joka ei ennestään esiinny tietämuskannassa

$$\frac{\exists v \alpha}{\alpha(\{v/k\})}$$

- ★ Esimerkiksi voimme lauseesta  $\exists x \text{Isä}(Jussi) = x$  päätellä *instantiation*  $\text{Isä}(Jussi) = F_1$ , missä  $F_1$  on uusi vakio

73

- ★ Herbrandin kuuluisan tuloksen mukaan 1. kertaluvun teorian loogisilla seurauksilla on todistus "propositionalisoidussa" teoriassa, joka käsittelee vain äärellistä osaa siitä

- ★ Täten sisäkkäisiä funktioita voidaan käsitellä syvyyden mukaan kasvavassa järjestyksessä ilman, että menetetään mahdollisuus johtaa kaikki loogiset seuraukset

- ★ Päätely on siis täydellistä

- ★ Analogia Turingin koneiden pysähtymisongelmaan kuitenkin osoittaa, että *ongelma on ratkeamaton*

- ★ Tarkemmin: *ongelma on osittain ratkeava*, hahmoteltu menetelmä löytää todistuksen loogisille seurauksille

75

## Samaistus

- ★ Eliminoimalla eksistenssivanttorit korvaamalla muuttujat Skolem vakioilla ja universaalikvanttorit kaikilla mahdollisilla instantiaatioillaan, muuttuu tietämuskanta oleellisesti propositionaaliseksi

- ★ Muuttujattomat atomilauseet kuten *Teekkari(Jussi)* ja *Äiti(Jussi, Meeri)* on vain nähtävä propositiesymboleina

- ★ Täten propositiologiikan päätelyä voidaan soveltaa predikaattilogiikan lauseisiin

- ★ Jos tietämuskannassa kuitenkin käytetään funktiosymboleita, niin mahdollisten korvaavien perustermien lkm on ääretön

$$\text{Isä}(\text{Isä}(\text{Isä}(Jussi)))$$

74

- ★ Päätely propositiologiikassa on ilmiselvästi turhan raskasta

- ★ Muuttujasijoitusten aukikirjoittaminen vaikuttaa turhalta

- ★ Jos meillä on sijoitus  $\theta$  s.e.  $p'_i(\theta) = p_i$ , kaikilla  $i$ , missä  $p'_i$  ja  $p_i$  ovat atomilauseita kuten myös  $q$ , niin voimme käyttää *yleistettyä Modus Ponensia*

$$\frac{p'_1, \dots, p'_n, (p_1 \wedge \dots \wedge p_n \Rightarrow q)}{q(\theta)}$$

- ★ Esimerkiksi faktasta *Teekkari(Jussi)* sekä lauseista  $\forall x \text{Ahkera}(x)$  ja  $\forall y \text{Teekkari}(y) \wedge \text{Ahkera}(y) \Rightarrow \text{Menestyjä}(y)$  voimme päätellä *Menestyjä(Jussi)* sidonnan  $\{y/Jussi, x/Jussi\}$  perusteella

76

- ★ Yleistetty Modus Ponens on eheä päätelysääntö

- ★ Samaan tapaan kuin yMP voidaan "korottaa" propositiologiikasta predikaattilogiikkaan, voidaan myös eteen- ja taaksepäin ketjutus sekä resoluutio muokata

- ★ Päätelyalgoritmeissa keskeinen käsite on lauseiden *samaistus* (unification)

- ★ Samaistusalgoritmi UNIFY palauttaa syötteenä annetut lauseet samastavan sijoituksen, jos sellainen on olemassa

$$\text{UNIFY}(p, q) = \theta, \text{ s.e. } p(\theta) = q(\theta)$$

muuten samaistus epäonnistuu (fail)

77

$$\text{UNIFY}(\text{Tuntee}(Jussi, x), \text{Tuntee}(Jussi, Jaan)) = \{x/Jaana\}$$

$$\text{UNIFY}(\text{Tuntee}(Jussi, x), \text{Tuntee}(y, Pekka)) = \{x/Pekka, y/Jussi\}$$

$$\text{UNIFY}(\text{Tuntee}(Jussi, x), \text{Tuntee}(y, \text{Äiti}(y))) = \{y/Jussi, x/\text{Äiti}(Jussi)\}$$

$$\text{UNIFY}(\text{Tuntee}(Jussi, x), \text{Tuntee}(x, \text{Elina})) = \text{fail}$$

- ★ Viimeinen samaistus epäonnistuu, koska muuttujaa  $x$  ei voida samanaikaisesti sitoa sekä *Jussi*in että *Elina*an

- ★ Koska muuttujat ovat universaalikvantifioituja, niin *Tuntee(x, Elina)* tarkoittaa, että kaikki tuntevat *Elinan*

- ★ Näin ollen samaistuksen pitäisi onnistua

78

- ★ Ongelman edellä aiheuttaa saman muuttujanimen käyttö

- ★ Toisaalta eri lauseiden muuttujanimillä ei ole merkitystä (ennen sidontaa) ja muuttujien nimentä voidaan standardoida

- ★ Mahdollisia samaistuksia on useita, mikä niistä tulisi palauttaa?

- ★ Samaistusalgorin edellytetään palauttavan kaikkein yleisimmän samaistuksen

- ★ Mitä vähemmän rajoituksia (sitomisia vakioihin) muuttujasijoitus asettaa, sitä yleisempi se on

$$\begin{aligned} &\text{UNIFY}(\text{Tuntee}(Jussi, x), \text{Tuntee}(y, z)) \\ &\quad \{y/Jussi, z/x\} \\ &\quad \{y/Jussi, x/Jussi, z/Jussi\} \end{aligned}$$

79

## Eteenpäin ketjutus

- ★ Kuten edellä, tarkastelemme Hornin normaalimuodossa olevia tietämuskantoja

- ★ Jokainen lause on siis atomilause tai implikaatio, jonka runko on positiivisten atomilauseiden konjunktio ja kärki on yksittäinen positiivinen atomilause

$$\text{Teekkari}(Jussi)$$

$$\text{Ahkera}(x)$$

$$\text{Teekkari}(y) \wedge \text{Ahkera}(y) \Rightarrow \text{Menestyjä}(y)$$

- ★ Nyt vaan atomilauseet voivat sisältää myös muuttujia

- ★ Muuttujat ovat aina universaalisesti kvantifioituja

80

\* Kuten propositiologiikassa lähtien liikkeelle faktoista, soveltaen yleistettyä Modus Ponensia voidaan tehdä eteenpäin ketjuttavaa päättelyä

\* Nyt on huolehdittava siitä, ettei "uusi" fakta ole pelkkä tunnetun faktan uudelleennimintä

*Pitää(x, Karkki) Pitää(y, Karkki)*

\* Yleistetyn Modus Ponensin soveltuksena eteenpäin ketjutus on eheä päättelyalgoritmi

\* Se on myös täydellinen siinä mielessä, että tietämiskannan mitä tahansa loogista seurausta koskevaan kyselyyn saadaan vastaus

81

## Datalog

\* Datalog tietämiskannassa ei esiinny laisinkaan funktiosymboleja

\* Tässä tapauksessa voimme helposti todistaa päättelyn täydellisyyden

\* Olk. tietämiskannan

- ◊ predikaattien lkm  $p$ ,
- ◊ predikaattien maksimi ariteetti (= argumenttien lkm)  $k$  ja
- ◊ vakioiden lkm  $n$

\* Vakioarvoihin sidottuja faktoja on kork.  $pn^k$

\* Täten näin monen kierroksen jälkeen on saavutettu *kiintopiste* (fixed point), jossa uudet päättelyt eivät enää ole mahdollisia

82

\* Datalogilla polynominen askelten lkm riittää kaikkien loogisten seurausten generoimiseen

\* Yleisessä tapauksessa joudumme vetoamaan Herbrandin tulokseen

\* Kyselyllä, joka ei ole teorian (=tietämiskannan) looginen seuraus, eteenpäin ketjutus ei välttämättä pysähdy

\* Esim. Peanon aksiomilla tietämiskantaan lisättäisiin faktat

*NatNum(S(0)).*

*NatNum(S(S(0))).*

*NatNum(S(S(S(0)))).*

...

\* Päättely on vain osittain ratkeavaa

83

## Taaksepäin ketjutus

\* Predikaattilogiikassa taaksepäin ketjutuksessa tutkitaan niiden sääntöjen runkoja, joiden kärki samaistuu maalin kanssa

\* Rungon kaikista konjunkteista tehdään rekursiivisesti maaleja

\* Kun maali samaistuu faktaan (lause, jolla on kärki vaan ei runkoa) ei uusia (ali)maaleja tarvitse generoida

\* Syvyysuuntainen haku

\* Palautettava muuttujasijoitus koostetaan kaikkien välivaiheiden ratkaisemiseksi tarvituista sijoituksista

\* Prologin päättely perustuu taaksepäin ketjutukseen

84

## Logiikkaohjelmointi

\* Prolog, Alain Colmerauer 1972

\* Ohjelma = Hornin lauseina ilmaistu tietämiskanta

\* Tietämiskantaan kohdistuvat kyselyt

\* *Suljetun maailman oletus*:  $\neg\phi$  oletetaan todeksi, jos lausetta  $\phi$  ei voi johtaa tietämiskannasta

\* Syntaksi: isot kirjaimet muuttujia, pienet vakioita, säännön kärki edeltää runkoa, implikaatiomerkin sijaan  $:-$ , pilkku konjunktioimerkin tilalla, piste päättää lauseen

```
menestyjä(X) :-
    teekkari(X), ahkera(X).
```

\* Prologissa on paljon erikoismerkintöjä listoille ja aritmetiikalle

85

\* Ohjelma `append(X, Y, Z)` onnistuu, jos lista  $Z$  on listojen  $X$  ja  $Y$  yhdistämisen (katenoinnin) tulos

```
append( [ ], Y, Y ).
append( [A|X], Y, [A|Z] ) :-
    append(X, Y, Z).
```

\* Kysely: `append([1],[2],Z)?`  
 $Z=[1,2]$

\* Voimme esittää ohjelmalle myös kyselyn `append(A,B,[1,2])?`: Minkä kahden listan yhdistäminen tuottaa listan  $[1,2]$ ?

\* Vastauksena saamme mahdolliset muuttujasidonnat

```
A=[ ]   B=[1,2]
A=[1]   B=[2]
A=[1,2] B=[ ]
```

86

\* Ohjelman lauseet suoritetaan järjestyksessä

\* Myös säännön rungon konjunkteja käsitellään järjestyksessä (vasemmalta oikealle)

\* Aritmetiikkaa varten on valmiita funktioita, joita ei enää tarvitse päättellä

◊ Esim.  $X \text{ is } 4+3 \implies X=7$

\* Mm. I/O hoituu sisäänrakennetuilla predikaateilla, joilla on sivuvaikutuksia

\* Suljetun maailman oletus: negaatio

```
alive(X) :- not dead(X).
```

"Kaikki, joiden ei voi todistaa kuolleen, ovat elossa"

87

\* Prologin negaatio ei vastaa logiikan negaatiota (suljetun maailman oletuksella)

```
vapaa_teeikkari(X) :-
    not naimisissa(X),
    teekkari(X).
```

```
teeikkari(pekka).
naimisissa(jussi).
```

\* Suljetun maailman oletuksen perusteella  $X=pekka$  on ohjelman ratkaisu

\* Ohjelman suoritus kuitenkin epäonnistuu, koska kun  $X=jussi$  epäonnistuu rungon ensimmäinen predikaatti

\* Jos säännön konjunktit kirjoitettaisiin toiseen järjestykseen, niin se onnistuisi

88

★ Yhtäsuuruusvertailu onnistuu, jos tarkasteltavat termit voidaan samais-  
taa

◊ Esim.  $X+Y=2+3 \implies X=2, Y=3$

★ Prolog ei tee kaikkia tarpeellisia tarkistuksia muuttujasijoituksien yhteydessä

$\implies$  Päätely ei ole eheää

★ Yleensä ei kuitenkaan ongelmia

★ Syvyysuuntaisen päättelyn seurauksena voi olla ikuinen silmukka

`path(X,Z):- path(X,Y), link(Y,Z).`

`path(X,Z):- link(X,Z).`

★ Huolellisuus kuitenkin usein auttaa

`path(X,Z):- link(X,Z).`

`path(X,Z):- path(X,Y), link(Y,Z).`

89

★ Prologin suoritusta voidaan karsia *leikkauksella* (cut)

★ Negaatio leikkauksen avulla

`not X:- X, !, fail.`

`not X.`

★ Edellä *fail* aiheuttaa epäonnistumisen

★ Leikkauksen kohdalla kiinnitetään kaikki ne sidonnat, jotka on tehty säännön tutkimisen aloittamisesta lähtien

★ Rungon konjunkteille, jotka edeltävät leikkausta, ei enää yritetä muodostaa uusia ratkaisuja

★ Muita saman kärjen omaavia sääntöjä ei enää kokeilla

90

★ Prolog voi päätyä samaan ratkaisuun monen päättelypolun kautta

★ Tällöin sama ratkaisu palautetaan useita kertoja

`minimum(X,Y,X):- X<=Y.`

`minimum(X,Y,Y):- X>=Y.`

★ Molempien sääntöjen kautta löytyy sama ratkaisu kyselylle `minimum(2,2,M)?`

★ Leikkauksen käytössä päättelyn optimointiin on oltava huolellinen

`minimum(X,Y,X):- X<=Y, !.`

`minimum(X,Y,Y).`

★ Yo. ohjelma on virheellinen, esimerkiksi `minimum(2,8,8)` on siinä tosi

91

★ Prologin ja logiikkaohjelmoinnin kynnyskysymys tietysti on tehokkuus

★ Prologissa on monia tehostuskeinoja käytössä

★ Esim. sen sijaan, että alimaalille tuotettaisiin kaikki mahdolliset ratkaisut ennen seuraavaan siirtymistä, Prolog tulkki tyytyy (toistaiseksi) yhteen

★ Samoin muuttujasidonta on kullakin hetkellä yksikäiteinen, vasta umpikujaan ajautuminen ja *peruutus* voi johtaa muuttujan uudelleensitomiseen

★ Peruutus edellyttää *historiajärljen* (trail) ylläpitämistä

92

## Resoluutio

★ Kurt Gödelin *täydellisyyslause* (1930)  
1. kertaluvun logiikalle: jokaisella loogisella seurauksella on äärellinen todistus

★  $T \models \phi \Leftrightarrow T \vdash \phi$

★ Vasta Robinsonin (1965) resoluutioalgoritmi antoi käytännöllisen menetelmän todistusten muodostamiseksi

★ Gödelin tuloksista tunnetumpi on tietysti *epätäydellisyyslause*: matemaattisen induktion omaavan teorian kaikkia loogisia seurauksia ei voida todistaa aksioomista

★ Erityisesti tämä koskee lukuteoriaa, jota siis ei voi aksiomisoida

93

★ Resoluutiota varten lauseet on muutettava konjunktiiiviseen normaalimuotoon. Esim.

$\forall x [\forall y \text{Eläin}(y) \Rightarrow \text{Rakastaa}(x, y)]$

$\Rightarrow [\exists y \text{Rakastaa}(y, x)].$

★ Implikaation poisto

$\forall x [-\forall y \neg \text{Eläin}(y) \vee \text{Rakastaa}(x, y)]$

$\vee [\exists y \text{Rakastaa}(y, x)].$

★ Negaation siirto sisään

$\forall x [\exists y \text{Eläin}(y) \vee \neg \text{Rakastaa}(x, y)]$

$\vee [\exists y \text{Rakastaa}(y, x)].$

★ Muuttujanimien standardointi

$\forall x [\exists y \text{Eläin}(y) \vee \neg \text{Rakastaa}(x, y)]$

$\vee [\exists z \text{Rakastaa}(z, x)].$

94

★ Skolemisointi

$\forall x [\text{Eläin}(F(x)) \vee \neg \text{Rakastaa}(x, F(x))]$

$\vee \text{Rakastaa}(G(x), x).$

★ Universaalikvanttorin poisto

$[\text{Eläin}(F(x)) \vee \neg \text{Rakastaa}(x, F(x))]$

$\vee \text{Rakastaa}(G(x), x).$

★ Distributiivisuuden soveltaminen

$[\text{Eläin}(F(x)) \vee \text{Rakastaa}(G(x), x)]$

$\wedge [\neg \text{Rakastaa}(x, F(x)) \vee \text{Rakastaa}(G(x), x)].$

★ Lopputulos ei enää ole helposti ymmärrettävä, mutta se ei haittaa, koska muunnosproseduuri on automatisoitavissa

95

★ 1. kertaluvun logiikan kaksi literaalaa ovat komplementaaraisia, jos yksi voidaan samaistaa toisen negaation kanssa

★ Täten resoluutioaskel yhdellä komplementaarisella literaalilla on

$$\frac{\ell_1 \vee \dots \vee \ell_k, m}{(\ell_1 \vee \dots \vee \ell_{i-1} \vee \ell_{i+1} \vee \dots \vee \ell_k)(\theta)}$$

missä  $\text{UNIFY}(\ell_i, \neg m) = \theta$

★ Resoluutio on myös predikaattilogiikalle täydellinen päättelyjärjestelmä siinä mielessä, että sen avulla voidaan tarkastaa tietämuskannan loogiset seuraukset

★  $\text{KB} \models \alpha$  todistetaan osoittamalla, että  $\text{KB} \wedge \neg \alpha$  on toteutumaton ristiriidan avulla

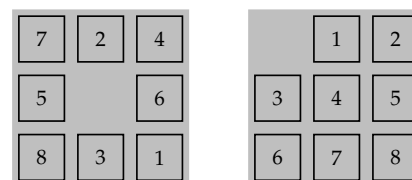
96

- ★ *Teoreemantodistimissa* logiikkaohjelmoinnin Hornin lauseisiin rajoitettu kieli on laajennettu täyteen 1. kertaluvun logiikkaan
- ★ Myös esim. Prologin tapa antaa kontrollin sekoittaa lauseiden logiikkaa on yleensä poistettu
- ★ Päättely ei riipu lauseiden tai niiden osien kirjoitusjärjestyksestä
- ★ Sovelluskohde: ohjelmien ja laitteiston verifiointi
- ★ Matematiikassa teoreemantodistimet ovat nousseet näkyvään asemaan
- ★ Esimerkiksi 1996 Otter-ohjelman seuraaja ensimmäisenä todisti, että Robbinsin 1933 esittämät aksiomat määrittävät Boolean algebran

97

- ★ Maailman mahdolliset tilat määräävät etsintäongelman *tila-avaruuden*  $\mathcal{S}$
- ★ Lähtötilanteessa maailma on *alkutilassa*  $s_1 \in \mathcal{S}$
- ★ Agentin tavoitteena on päätyä alkutilasta yhteen maalitiloista  $G \subseteq \mathcal{S}$
- ★ Agentin mahdolliset toiminnot määritellään usein *seuraajafunktion*  $S : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{S})$  avulla
- ★ Kullakin tilalla  $s \in \mathcal{S}$  on joukko seuraajajaloja  $S(s)$ , joihin voidaan siirtyä yhdessä askeleessa
- ★ Poluilla on ei-negatiivinen *kustannus*, joka yleensä on askelkustannusten summa

99



- ★ Esimerkiksi 8-puzzlen maailman mahdollisia tiloja ovat kaikki  $9!/2 = 181\,440$  mahdollisuutta asettaa palat
- ★ Alkutila on yksi mahdollisista tiloista
- ★ Maalitila on yllä oikealla annettu asetelma
- ★ Seuraajafunktion mahdollisia arvoja ovat tyhjän ruudun siirtäminen vasemmalle, oikealle, ylös tai alas
- ★ Kukin siirto maksaa yhden yksikön ja polun kustannus on siirtojen lkm

101

- ★ Hakupuun solmuihin liitetään
  - ◇ tieto sitä vastaavasta tilasta,
  - ◇ linkki isäsolmuun,
  - ◇ tieto sovelletusta toiminnosta, joka on johtanut solmun laajentamiseen,
  - ◇ polun kustannus alkutilasta lähtien tätä solmua vastaavaan tilaan saakka ja
  - ◇ solmun syvyys
- ★ Hakupuun globaaleja ominaisuuksia ovat
  - ◇ (keskimääräinen tai maksimaalinen) haarautumisaste  $b$ ,
  - ◇ matalimmalla olevan maalisolmun syvyys  $d$  sekä
  - ◇ tila-avaruuden pisimmän polun pituus  $m$

103

### 3. Ongelmanratkaisu ja hakualgoritmit

- ★ Tavoitehakuinen agentti pyrkii ratkaisemaan ongelmia suorittamalla toimintoja, jotka johtavat haluttuihin tiloihin
- ★ Tarkastelemme ensin tilannetta, jossa agentilla ei ole mitään ylimääräistä tietoa ongelmasta
- ★ *Sokeassa haussa* vain ratkaistavan ongelman rakenne on määritelty
- ★ Agentti pyrkii saavuttamaan maalitilan
- ★ Maailma on staattinen, diskreetti ja deterministinen

98

- ★ Ed. määritelmät määräävät luontevasti suunnatun, painotetun verkon
- ★ Yksinkertaisin tavoite etsinnässä on selvittää onko alkutilasta  $s_1$  saavutettavissa yhtään maalitilaa
- ★ Ongelman varsinainen *ratkaisu* kuitenkin on polku alkutilasta maalitilaan
- ★ Usein tietysti huomioidaan myös polkujen kustannukset ja pyritään löytämään niiden suhteen optimaalinen polku maalitilaan
- ★ Useat tavoitehakuisen agentin tehtävät on helppo formuloida suoraan tässä esityksessä

100

### Hakupuun

- ★ Kun maalin etsintä lähtee liikkeelle alkutilasta edeten seuraajafunktion määräämin askelin, niin etsinnän etenemistä voidaan tarkastella puurakenteessa
- ★ Kun hakupuun alkutilaa vastaava juurisolmu laajennetaan, niin siihen sovelletaan seuraajafunktiota, jonka tuloksena hakupuun uusiksi solmuiksi — juuren lapsiksi — luodaan seuraajajaloja vastaavat solmut
- ★ Edelleen muita solmuja laajentamalla voidaan etsintää jatkaa
- ★ Etsintästrategia (hakualgoritmi) määrää missä järjestyksessä solmuja puussa laajennetaan

102

- ★ Hakualgoritmit toteutetaan normaalin puun läpikäynnin erikoistapauksina
- ★ Haun aikavaativuutta mitataan useimmiten puuhun luotujen solmujen lukumäärällä
- ★ Tilavaativuus puolestaan mittaa yht'aikaisesti muistissa säilytettävien solmujen lkm:ää
- ★ Hakualgoritmi on *täydellinen*, jos jokin maalitila voidaan saavuttaa alkutilasta
- ★ Täydellisen algoritmin palauttama ratkaisu ei välttämättä ole *optimaalinen*: alkutilasta voi olla saavutettavissa useita eri kustannuksen omaavia maaleja

104

## Leveysuuntainen haku

- ★ Kun puun läpikäyntijärjestys on esijärjestys, niin hakupuun solmut tulevat käsitellyiksi taso kerrallaan
- ★ Kaikki samalla syvyydellä olevat solmut laajennetaan ennen kuin yhtään alemman tason solmuista käsitellään
- ★ Jos maalitila on syvyydellä  $d$ , niin kaikki edeltävien  $d - 1$  tason solmut on käsiteltävä (ja talletettava) ennen sen löytymistä
- ★ Kun hakupuun haarautumisaste on  $b$ , niin aika- ja tilavaativuus on pahimmillaan

$$b + b^2 + \dots + b^d + (b^{d+1} - b) = O(b^{d+1})$$

105

- ★ Leveysuuntainen haku palauttaa optimaalisen ratkaisun, jos kaikkien askelten kustannus on sama
- ★ Toisaalta tähän erikoistapaukseen voidaan soveltaa *ahnetta* (greedy) algoritmia, joka aina laajentaa sen solmun, jonka tähänastinen polun kustannus on pienin
- ★ Jos ongelman kaikki askelkustannukset ovat aidosti positiivisia, niin ahneen algoritmin palauttama ratkaisu on optimaalinen
- ★ Algoritmin tilavaativuus on edelleenkin suuri
- ★ Kun askelkustannukset ovat kaikki samoja, on ahne algoritmi yhteneväinen leveysuuntaisen haun kanssa

107

## Syvyysuuntainen haku

- ★ Kun hakupuun solmut laajennetaan sisäjäestyksessä, niin hakualgoritmi käsittelee aina ensinnä syvimmällä olevan solmun
- ★ Kun yksi puun haara juuresta lehteen on tullut käsitellyksi, haku peruuttaa puussa takaisin syvimmällä olevaan käsittelemättömään solmuun
- ★ Syvyysuuntaisen haun hyvä puoli on pieni muistitilavaativuus
- ★ Kullakin hetkellä riittää pitää muistissa yksi polku juuresta lehteen ja polulle kuuluvien solmujen laajentamattomat sisärsolmut
- ★ Tilavaativuus on siis  $bm + 1$

108

- ★ Edellisen esimerkin tapauksessa, jossa maalisolmu on tasolla 12 muistia vaaditaan vain 118 kB
- ★ Jos hakupuu on ääretön, niin syvyysuuntainen haku ei ole täydellinen menetelmä
- ★ Ainut maalisolmu voi sijaita vasta viimeksi tutkittavassa puun haarassa
- ★ Pahimmassa tapauksessa syvyysuuntainen hakukin vaatii eksponentiaalisen määrän aikaa:  $O(b^m)$
- ★ Pahimmillaan  $m \gg d$ , syvyysuuntaisen haun vaatima aika voi olla paljon suurempi kuin leveysuuntaisen haun
- ★ Myöskään löydetyn ratkaisun optimaalisuutta ei voida taata

109

## Rajoitetun syvyyden haku

- ★ Äärettömän syvyyden haaran tutkiminen voidaan estää rajoittamalla etsintä syvyyteen  $\ell$
- ★ Tason  $\ell$  solmuja käsitellään lehtinä
- ★ Syvyysuuntainen haku on erikoistapaus, jossa  $\ell = m$
- ★ Tapauksessa  $d \leq \ell$  hakualgoritmi on täydellinen, mutta yleisesti ratkaisun löytymistä ei voi taata
- ★ Algoritmi ei myöskään (tietysti) takaa optimaalisen ratkaisun löytämistä
- ★ Aikavaativuus on nyt  $O(b^\ell)$  ja tilavaativuus  $O(b\ell)$

110

## Iteratiivinen syventäminen

- ★ Rajoitetun syvyyden haun ja yleisen syvyysuuntaisen haun hyvät puolet voidaan yhdistää, kun annetaan parametrin  $\ell$  arvon vähitellen kasvaa
- ★ Esim.  $\ell = 0, 1, 2, \dots$  kunnes maalisolmu löytyy
- ★ Itseasiassa näin saadaan yhdistettyä myös leveys- ja syvyysuuntaisen haun hyvät puolet
- ★ Tilavaativuus pysyy kurissa, koska hakualgoritmina on syvyysuuntainen haku
- ★ Toisaalta parametrin  $\ell$  vähittäinen kasvattaminen takaa, että menetelmä on täydellinen ja optimaalinen silloin kun polun kustannus on solmun syvyyden suhteen ei-vähenevä

111

## Kaksisuuntainen haku

- ★ Jos edeltäjäsolmut ovat helposti lasketavissa — esim.  $\text{Pred}(n) = S(n)$  — niin haku voi edetä yht'aikaisesti alkutilasta eteenpäin ja maalitilasta taaksepäin
- ★ Etsintä päättyy kun haut kohtaavat
- ★ Motivaatio tälle ajatukselle on, että  $2b^{d/2} \ll b^d$
- ★ Jos molempiin suuntiin etenevät haut toteutetaan leveysuuntaisella haulla, on menetelmä täydellinen ja johtaa optimaaliseen tulokseen
- ★ Jos maalitila on yksikäsitteinen, niin taaksepäin eteneminen on selvää, mutta monet maalitilat voivat edellyttää uusien väli aikaisten maalitilojen luomista

112

Syvyys	Aika	Muistitila
2	.11 s	1 MB
4	11 s	106 MB
6	19 min	10 GB
8	31 h	1 teraB
10	129 vrk	101 teraB
12	35 a	10 petaB

106

- ★ Haun merkittävä ongelma on mahdollisuus laajentaa uudelleen jo ker- taalleen tutkittu tila
- ★ Näin äärellinen tila-avaruus voi johtaa äärettömään hakupuuhun
- ★ Ratkeava ongelma voi muuttua käy- tännössä ratkeamattomaksi
- ★ Jo laajennettujen tilojen tunnistami- nen edellyttää käsiteltyjen tilojen tallentamista muistiin
- ★ Näin on tehtävä myös syvyys-suun- taisessa haussa
- ★ Toisaalta karsinta voi johtaa optimaalisen polun karsimiseen

113

- ★ Ilman sensoreita toimiva agentti jou- tuu tekemään päättelyä tilajoukoissa tilojen sijaan
- ★ Kullakin hetkellä agentilla on usko- mus siitä, missä tiloissa se voi olla
- ★ Toimitaan siis tila-avaruuden  $\mathcal{S}$  po- tenssijoukossa  $\mathcal{P}(\mathcal{S})$ , jossa on  $2^{\mathcal{S}}$  uskomustilaa (belief state)
- ★ Nyt ratkaisu on polku, joka johtaa uskomustilaan, jonka kaikki jäsenet ovat maalitiloja
- ★ Jos toimintaympäristö on osittain havainnoitavissa tai jos toiminnot ovat epävarmoja, niin jokainen uusi toiminto antaa uutta informaatiota
- ★ Jokainen mahdollinen havainto mää- rää tilanteen jota varten on tehtävä suunnittelua

115

- ★ Edellä teimme (epärealistisen) ole- tuksen, että maailma on täysin ha- vainnoitavissa ja deterministinen
- ★ Tällöin agentti aina tietää täsmälleen missä tilassa se on ja voi laskea toi- mintojonon suorituksesta seuraavan jonon
- ★ Toisaalta agentin havainnot eivät anna mitään uutta tietoa
- ★ Realistisempi tilanne on, että agen- tin tieto tiloista ja toiminnoista on epätäydellinen
- ★ Jos agentilla ei ole sensoreita lain- kaan, niin sen näkökulmasta alkutila voi olla jokin monista ja täten myös toiminnot voivat johtaa moniin seu- raajatiloihin

114

- ★ Epävarmuuden aiheuttaja voi olla toinen agentti, *vastustaja* (adversary)
- ★ Maailman tilojen ja toimintojen seu- rausten ollessa epävarmoja, on agen- tin *tutkiskeltava* (explore) ympäristöä tietojen saamiseksi
- ★ Osittain havainnoitavassa maailmas- sa toimintojonoja ei voi kiinnittää ennakkoon suunniteltaessa, vaan niiden tulee olla ehdollisia uusille havainnoille
- ★ Kun agentti voi kerätä uutta tietoa toimintansa kautta, niin yleensä ei kannata tehdä suunnitelmia kaikkia mahdollisia tilanteita varten, vaan pikemminkin lomittaa toiminta ja suunnittelu

116

### 3.1 Heuristinen haku

- ★ Sokean haun sijaan meillä on nyt on- gelmakohtaista tietoa käytössämme etsinnän tehostamiseksi
- ★ Tila-avaruuden säännöllisestä raken- teesta on ylimääräistä tietoa
- ★ Hakupuun solmun  $n$  lupaavuus maalitilan löytymisen kannalta arvo- tetaan *evaluointifunktiolla*  $f(n)$
- ★ Paras ensin -haku laajentaa mahdol- lisista solmuista sen, joka näyttää evaluointifunktion arvon perusteella lupaavimmalta
- ★ Yleensä pyrkimyksenä on funktion  $f$  arvon minimoiminen

117

#### Heuristiset funktiot

- ★ Evaluointifunktion keskeinen kom- ponentti on heuristinen funktio  $h(s)$ , joka antaa arvion tilasta  $s$  maalitilaan johtavan lyhimmän polun kustan- nukselle
- ★ Etsintäongelman yhteydessä heuris- tiikalla tarkoitetaan varmaa (mutta löysää) ylä- tai alarajaa parhaan rat- kaisun kustannukselle
- ★ Maalitilaa vastaavassa solmussa  $n$  kuitenkin vaaditaan, että  $h(n) = 0$
- ★ Esim. mitään informaatiota tuomaton varma alaraja olisi asettaa  $h(s) \equiv 0$
- ★ Heuristiset funktiot ovat yleisin tapa antaa ylimääräistä tietoa etsintäon- gelmaan

118

### Ahne haku

- ★ Pyritään ratkaisuun mahdollisimman nopeasti laajentamalla se solmu, joka on lähinnä maalia
- ★ Evaluointifunktio siis on  $f(n) = h(n)$
- ★ Esim. maantie-etäisyyksien mini- moinnissa heuristinen alaraja kau- punkien etäisyyksille voisi olla nii- den etäisyys linnuntietä
- ★ Ahne haku jättää huomiotta solmuun  $n$  jo kuljetun polun kustannuksen
- ★ Täten sen tuottama ratkaisu ei ole välttämättä optimaalinen
- ★ Jos toistuvia tiloja ei havaita, voi ahne haku oskilloida ikuisesti kahden lupaavan tilan välillä

119

- ★ Äärettömien polkujen mahdollisuus merkitsee, ettei ahne haku ole myös- kään täydellinen
- ★ Ahneutensa vuoksi haku tekee valin- toja, jotka voivat johtaa umpikujaan; tällöin hakupuussa peruutetaan ta- kaisin syvimmällä olevaan laajenta- mattomaan solmuun
- ★ Ahne haku muistuttaa syvyys-suun- taista hakua tavassaan tutkia haku- puuta oksa kerrallaan ja peruuttaa umpikujaan joutuessaan
- ★ Pahimman tapauksen aika- ja tilavaa- tivuus ahneella haulalla on  $O(b^m)$
- ★ Heuristisen funktion hyvyys määrää ahneen haun käytännön toimivuus- den

120

## Hakualgoritmi A\*

- \* A\* laskee yhteen heuristisen funktion  $h(n)$  arvon ja solmun  $n$  saavuttamisen vaatiman kustannuksen  $g(n)$
- \* Evaluointifunktio  $f(n) = g(n) + h(n)$  siis arvioi halvimman solmun  $n$  kautta kulkevan ratkaisun kustannusta
- \* Jos  $h(n)$  ei koskaan yliarvioi maalin saavuttamisen kustannusta, niin puun läpikäynnissä A\* palauttaa optimaalisen ratkaisun
  - ◊ Olkoon  $g_2$  puuhun generoitu, ei-optimaalinen maalisolmu
  - ◊ Olk.  $C^*$  optimiratkaisun kustannus
  - ◊ Nyt  $h(g_2) = 0$ , joten  $f(g_2) = g(g_2) > C^*$

121

- \* Näille pätee *kolmioepäyhtälö* (triangle inequality) muodossa
$$h(n) \leq c([n, a], n') + h(n'),$$
missä  $n' \in S(n)$  (valittu toiminto on  $a$ ) ja  $c([n, a], n')$  on askelkustannus

- \* Linnuntie on myös monotoninen heuristiikka
- \* Monotonisella heuristiikalla A\* on optimaalinen myös verkkoetsinnässä
- \* Jos  $h(n)$  on monotoninen, niin arvot  $f(n)$  millä tahansa polulla ovat eiväheneviä, joten ensinnä tutkittu maali on optimaalinen ratkaisu

$$\begin{aligned} f(n') &= g(n') + h(n') \\ &= c([n, a], n') + g(n) + h(n') \\ &\geq g(n) + h(n) = f(n) \end{aligned}$$

123

- ◊ Toisaalta, jos ongelmaan on ratkaisu, niin puuhun on generoitu solmu  $n$ , joka kuuluu optimaaliseen polkuun
- ◊ Koska  $h(n)$  on aliarvio, niin  $f(n) = g(n) + h(n) \leq C^*$
- ◊ Näin ollen solmua  $g_2$  ei valita tutkittavaksi
- \* Esim. linnuntie on arvio, joka ei koskaan yliarvioi maantietä kuljettavaa matkaa
- \* Verkkoetsinnässä optimaalisen ratkaisun löytäminen vatii huolehtimaan, ettei optimaalista ratkaisua karsita toistuvissa tiloissa
- \* Erityisen tärkeä erikoistapaus ovat *monotoniset* heuristiikat

122

- \* Ratkaisua etsiessään A\* laajentaa kaikki solmut  $n$ , joilla  $f(n) < C^*$ , ja osan niistä, joilla  $f(n) = C^*$
- \* Sen sijaan kaikki solmut solmut  $n$ , joilla  $f(n) > C^*$ , karsitaan
- \* On selvää, että A\* on täydellinen hakualgoritmi
- \* Se on myös *optimaalisen tehokas* menetelmä, sillä minkä tahansa hakualgoritmin on tutkittava kaikki solmut, joilla pätee  $f(n) < C^*$
- \* Täydellisyydestä, optimaalisesta tehokkuudesta ja optimaalisen ratkaisun palauttamisesta huolimatta A\*:llä on myös heikkoutensa
- \* Solmujen lkm, joille pätee  $f(n) < C^*$ , voi olla eksponentiaalinen ratkaisun pituuden suhteen

124

- \* Jälleen kerran hakualgoritmin tila-vaativuus on kuitenkin pahempi ongelma
- \* Sen tähden monia vain rajoitetun muistin käyttäviä versioita A\*-algoritista on jouduttu kehittämään
- \* IDA\* (Iterative Deepening A\*) soveltaa iteratiivisen syventämisen ideaa
- \* Solmujen syvyyden sijaan katkaisuehtona on tällä kertaa  $f$ -kustannus
- \* Kullakin kierroksella uudeksi katkaisuehdoksi valitaan edelliseltä kierrokselta karsittujen solmujen pienin  $f$ -arvo
- \* Uudemmat algoritmit tekevät monimutkaisempaa karsintaa

125

## Heuristisista funktioista

- \* 8-puzzlessa voimme määritellä seuraavat heuristiset funktiot, jotka eivät koskaan yliarvioi
  - ◊  $h_1$ : väärässä positiossa olevien laattojen lkm: kutakin on siirrettävä ainakin kerran ennen kuin laatat ovat halutussa järjestyksessä
  - ◊  $h_2$ : väärässä positiossa olevien laattojen *Manhattan etäisyys* oikeasta paikastaan: laatat on kuljettava oikeille paikoilleen ennen kuin ne ovat halutussa järjestyksessä
- \* Alkutilanteessa kaikki laatat ovat väärässä positiossa:  $h_1(s_1) = 8$
- \* Toisen heuristiikan arvo esimerkikivalle on:

$$3 + 1 + 2 + 2 + 2 + 2 + 3 + 3 + 2 = 18$$

126

- \* Heuristiikka  $h_2$  antaa aina tiukemman arvion kuin  $h_1$
- \* Sen sanotaan *dominoivan* jälkimmäistä
- \* Koska A\* laajentaa kaikki solmut joilla
$$f(n) < C^* \Leftrightarrow h(n) < C^* - g(n),$$
niin tiukempi heuristinen arvio johtaa suoraan tehokkaampaa etsintään
- \* 8-puzzlen haarautumisaste on noin 3
- \* Efektiivisellä haarautumiskertoimella mitataan ongelman ratkaisussa generoitavien solmujen keskimääräistä määrää
- \* Esim. kun A\* käyttää heuristiikkaa  $h_1$ , niin 8-puzzlen efektiivinen aste on keskimäärin n. 1.4, ja heuristiikalla  $h_2$  n. 1.25

127

- \* Heurististen funktioiden kehittäiseksi voidaan tarkastella *relaksoituja* ongelmia, joista on poistettu joitain alkuperäisen ongelman rajoitteita
- \* Relaksoidun ongelman optimiratkaisun kustannus on aina aliarvio alkuperäisen ongelman ratkaisun kustannuksesta
- \* Alkuperäisen ongelman optimiratkaisu on myös helpotetun ongelman ratkaisu
- \* Esim. 8-puzzlen heuristiikka  $h_1$  on antaa optimaalisen kustannuksen helpotettuun peliin, jossa laatat voivat pomputa mihin paikkaan tahansa
- \* Samoin  $h_2$  antaa optimaalisen ratkaisun relaksoituun 8-puzzle peliin, jossa laatat voivat liikkua myös miehitettyihin ruutuihin

128

- ★ Jos tarjolla on joukko heuristisia funktioita, joista yksikään ei dominoi toisia, niin voimme käyttää koostettua heuristiikkaa

$$h(n) = \max \{ h_1(n), \dots, h_m(n) \}$$

- ★ Koostefunktio dominoi kaikkia komponenttifuntioita ja on monotoninen, jos komponentit eivät yliarvioi
- ★ Yksi relaxoimisen muoto on aliongelmiin tarkastelu
- ★ Esim. 8-puzzlessa voisimme tarkastella vain neljää laattaa kerrallaan ja antaa loppujen ajautua mihin tahansa positioon
- ★ Koostamalla eri laattoja koskevat heuristiikat yhteen saadaan koko ongelman heuristinen funktio, joka on paljon Manhattan etäisyyttä tehokkaampi

129

## Lokaali haku

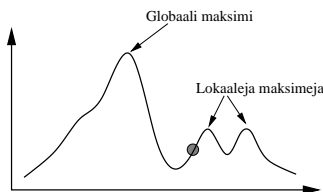
- ★ Kaikissa ongelmissa maaliin johtavalla polulla ei ole merkitystä
- ★ Vain ongelman ratkaisu on tarpeen tietää
- ★ Lokaalit hakualgoritmit pitävät yllä vain tietoa nykyisestä tilasta ja etsintä keskittyy yleensä vain sen naapureihin
- ★ Etsinnän kulkemia polkuja ei yleensä talleteta
- ★ Muistitilan tarve on vähäinen, vain vakiotila
- ★ Toimivat suurissa, jopa äärettömissä (jatkuissa), maailmoissa löytäen hyviä ratkaisuja kun systemaattiset hakumenetelmät eivät ole laisinkaan sovellettavissa

130

- ★ Lokaali haku on myös optimointiongelmiin ratkaisumenetelmä
- ★ Optimoinnissa tavoitteena on *kohdefunktion* (objective f.) suhteen parhaan tilan löytäminen
- ★ Optimointiongelmat eivät aina ole hakuongelmia samassa mielessä kuin edellä olemme tarkastelleet
- ★ Esimerkiksi Darwinilainen evoluutio pyrkii optimoimaan lisääntymiskykyisyyttä
- ★ Mitään lopullista maalitilaa ei ole olemassa
- ★ Myöskään polun kustannuksella ei ole merkitystä tässä tehtävässä

131

- ★ Kohdefunktion arvon optimointia voidaan kuvata tilamaisemana, jossa funktion arvoa vastaa huippujen korkeus ja laaksojen mataluus
- ★ Maksimointiongelmaan optimaalisen ratkaisun tuottava hakualgoritmi löytää *globaalin maksimin*
- ★ *Lokaalit maksimit* ovat korkeampia huippuja kuin yksikään niiden naapureista, mutta globaalia maksimia matalampia



132

## Mäenkapuaminen

- ★ Hill-climbing
- ★ Mäenkapuamisessa toistuvasti valitaan nykytilan  $s$  seuraajista  $se$ , jolla on korkein kohdefunktion  $f$  arvo

$$\max_{s' \in \mathcal{S}(s)} f(s')$$

- ★ Haku pysähtyy kun tilan naapurit ovat matalammalla
- ★ Yleensä haku siis päättyy lokaaliin maksimiin, sattumalta joskus globaaliin maksimiin
- ★ Myös tasangot aiheuttavat ongelmia tälle ahneelle lokaalille hakumenetelmälle, toisaalta alkutilanteen paraneminen on usein nopeaa

133

- ★ Sivuttaissiirtymät voidaan sallia kun etsintä saa edetä myös nykytilan kanssa yhtä hyviin tiloihin
- ★ *Stokastinen* mäenkapuaminen valitsee tilannetta parantavista naapureista satunnaisesti yhden
- ★ Naapureita voidaan tutkia satunnaisessa järjestyksessä ja valita ensimmäinen, joka on nykytilaa parempi
- ★ Myös nämä versiot mäenkapuamisesta ovat epätäydellisiä, sillä ne voivat edelleen juuttua lokaaliin maksimiin
- ★ *Satunnaisilla uudelleenkäynnistyksillä* voidaan varmistaa menetelmän täydellisyys
  - ◇ Mäenkapuaminen käynnistetään satunnaisesta alkutilasta, kunnes ratkaisu löytyy

134

## Simuloitu jäähtytys

- ★ Simulated annealing
- ★ *Satunnaiskulku* (random walk) — siirtyminen satunnaisesti valittuun naapuriin on se parempi kuin nykytila tai ei — on täydellinen etsintämenetelmä, mutta ohjaamattomana tehoton
- ★ Sallitaan "huonoja" siirtymiä jollain todennäköisyydellä  $p$
- ★ Tilannetta huonontavien siirtymien todennäköisyys pienenee eksponentiaalisesti ajan myötä ("lämpötila")
- ★ Siirtymäkandidaatti valitaan satunnaisesti ja se hyväksytään, jos 1) kohdefunktion arvo paranee ja 2) muuten tn.:llä  $p$
- ★ Jos lämpötilan pieneminen on riittävän hidasta, niin menetelmä konvergoi globaaliin optimiin tn.:llä  $\rightarrow 1$

135

## Lokaali keilahaku

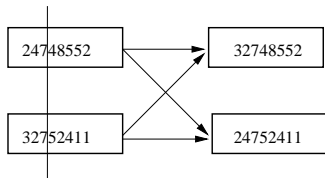
- ★ Local beam search
- ★ Lähdetään liikkeelle  $k$  kappaleesta satunnaisesti valittuja tiloja
- ★ Jokaisessa askeleessa kaikkien muistissa olevien tilojen seuraajat generoidaan
- ★ Jos jokin seuraajista on maalitila, algoritmi päättyy
- ★ Muutoin  $k$  parasta seuraajaa pidetään muistissa ja etsintä jatkuu
- ★ Keilahaun rinnakkainen etsintä johtaa nopeasti huoneiden polkujen hylkäämiseen
- ★ Stokastisessa keilahaussa jatkoon pääsevät seuraajasolmut valitaan niiden hyvyden mukaisella tn.:llä

136

## Geneettiset algoritmit

- ★ Geneettiset algoritmit (GA) soveltavat perinnöllisyydestä tuttuja operaatioita hakuun
- ★ GAn toiminta lähtee liikkeelle  $k$  satunnaisesti valitun tilan *populaatiosta* kuten keilahaku
- ★ Nyt merkkijonoin esitettyjä tiloja kutsutaan *yksilöiksi*
- ★ Seuraavan sukupolven populaation muodostamiseksi kaikki yksilöt evaluoidaan *kelpoisuusftiolla* (fitness  $f$ )
- ★ Yksilöiden tn. päästä lisääntymään riippuu niiden kelpoisuudesta (*valinta*)
- ★ Valittujen yksilöiden satunnaisia pareja *risteytetään* (crossover) keskenään s.e. molemmista merkkijonoista valitaan sopiva katkaisukohta ja niiden loppuosat vaihdetaan keskenään

137



- ★ Muodostettujen jälkeläisten kaikki merkkipositiot ovat lopulta *mutaation* mahdollisena kohteena
- ★ Mutaatiossa merkkejä vaihdetaan satunnaisesti toisiksi (hyvin) pienellä todennäköisyydellä
- ★ GAn operaatioiden toimivuus edellyttää huolellista yksilöiden koodauksen valintaa ja usein operaatioiden rajoittamista s.e. jälkeläiset ovat mielekkäitä
- ★ Hyvin suosittu heuristinen optimointimenetelmä nykyisin, tehoton

138

## Jatkuvat avaruudet

- ★ Olk. kohdeftio  $f(x_1, y_1, x_2, y_2, x_3, y_3)$  kuuden jatkuva-arvoisen muuttujan ftio
- ★ Funktion *gradientti*  $\nabla f$  on vektori, joka kertoo jyrkimmän nousun suunnan ja suuruuden
$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)$$
- ★ Usein gradienttia ei voi soveltaa globaalisti, mutta lokaalisti kyllä
- ★ Jyrkimmän nousun mäenkapuamista voidaan tehdä päivittämällä nykytilaa
$$\vec{x} \leftarrow \vec{x} + \alpha \nabla f(\vec{x}),$$
missä  $\alpha$  on "pieni vakio"

139

- ★ Jos kohdeftio ei ole derivoituva, niin gradientin suuntaa voidaan hakea empiirisesti tekemällä pieniä muutoksia kuhunkin koordinaattiarvoon
- ★ Vakion  $\alpha$  arvon määrittäminen on keskeistä: liian pieni arvo johtaa liian moneen askeleen ja liian suuri arvo puolestaan voi johtaa maksimin hukkaamiseen
- ★ Usein käytetty menetelmä on vakion  $\alpha$  tuplaaminen aina siihen asti kunnes  $f$ :n arvo pienenee
- ★ Muotoa  $g(x) = 0$  olevien yhtälöiden ratkaisemiseen voidaan käyttää Newton-Raphson -menetelmää
- ★ Ratkaisulle  $x$  lasketaan arvio Newtonin kaavalla  $x \leftarrow x - g(x)/g'(x)$

140

- ★ Funktion  $f$  maksimoimiseksi tai minimoimiseksi tulee löytää  $\vec{x}$  s.e.  $\nabla f(\vec{x}) = 0$
- ★ Soveltamalla Newtonin kaavaa

$$x \leftarrow x - \mathbf{H}_f^{-1}(\vec{x}) \nabla f(\vec{x}),$$

- missä  $\mathbf{H}_f(\vec{x})$  on toisten derivaattojen Hess-matriisi,  $H_{ij} = \partial^2 f / \partial x_i \partial x_j$
- ★ Hess-matriisi on neliöllinen ja suuriulotteisissa avaruuksissa sen käyttäminen on kallista
- ★ Myös jatkuva-arvoista optimointia koskevat mm. lokaalien optimien ja tasanakojen ongelmat
- ★ *Optimointi rajoitteiden vallitessa* edellyttää ratkaisulta joidenkin ehtojen täyttymistä
- ★ Esimerkiksi *lineaarissa ohjelmoinnissa* rajoitusehdot ovat lineaarisia epäyhtälöitä, jotka muodostavat konveksin alueen ja kohdeftio on myös lineaarinen

141

## Tuntemattomassa ympäristössä etsiminen

- ★ Online haussa agentin on reagoitava havaintoihinsa välittömästi tekemättä pitkälle tulevaisuuteen ulottuvia suunnitelmia
- ★ Tuntemattomassa ympäristössä tutkiskelu on välttämätöntä: agentin on kokeiltava toimintojensa vaikutuksia saadakseen tietoja maailman tiloista
- ★ Nyt agentti ei voi laskea nykytilan seuraajia, vaan sen on kokeiltava mikä tila seuraa toiminnon suorittamisesta
- ★ Online-algoritmia verrataan usein parhaan offline-algoritmin toimintaan
- ★ Näiden antamien ratkaisujen suhdetta kutsutaan online-algoritmin *kilpasuh-teeksi* (competitive ratio)
- ★ Online-hakualgoritmin kilpasuhteen määrittämiseksi verrataan agentin kulkeman polun kustannusta sen polun kustannukseen, joka olisi kuljettu, jos ympäristö olisi ollut ennalta tuttu
- ★ Mitä pienempi kilpasuhde on, sen parempi
- ★ Online-algoritmeja voidaan analysoida ajattelemalla niiden toiminta pelinä pahantahtoisen vastustajan kanssa
- ★ Vastustaja saa valita maailman lennosta pakottaakseen online-algoritmin huonoon käyttäytymiseen
- ★ Kaikki käsittelemämme hakualgoritmit eivät sovellu online-hakuun
- ★ Esimerkiksi A\* perustuu oleellisesti siihen, että hakupuussa voidaan laajentaa mikä tahansa generoitu solmu
- ★ Online-maailmassa laajennuksen tulee kohdistua siihen solmuun, jossa agentti on
- ★ Syvyyssuuntaisen haun lokaalisuus (paitsi peruutusvaiheessa) tekee algoritmistä käyttökelpoisen (jos toiminnot voidaan konkreettisesti peruttaa)
- ★ Syvyyssuuntaisen haku ei ole kilpailukykyinen (kilpasuhteelle ei voida asettaa rajaa)

142

143

144

- ★ Mäenkapuaminen on itse asiassa online-algoritmi, mutta juuttuu lo-kaaleihin maksimeihin
- ★ Satunnaisia uudelleenkäynnistyksiä ei voi käyttää
- ★ Satunnaiskulut ovat liian tehottomia
- ★ Lisämuistin avulla mäenkapuaminen voi olla käyttökelpoinen menetelmä
- ★ Kullekin vierailulle tilalle ylläpidetään tämänhetkistä parasta arviota matkasta maaliin
- ★ Lokaalissa minimissä pysymisen sijaan agentti pyrkii arvionsa mukaan lyhintä reittiä maaliin
- ★ Samalla lokaalin minimin arvot päivityvät pois

145

- ★ Olkoon pelaajien nimet **min** ja **max**
- ★ Lähtötilanteessa peliasetus on pelin sääntöjen mukainen ja ensimmäinen siirtovuoro on pelaajalla **max**
- ★ Seuraajafunktio määrää pelin sallitut siirrot
- ★ Pelin loppuminen tunnustetaan
- ★ Hyötyfunktio (utility  $f$ ) antaa loppu-tiloille numeerisen arvon, numeerinen arvo voi olla esim. shakissa vain -1, 0, +1 tai laudalla olevien nappuloiden laskennallinen arvo
- ★ **max** pyrkii maksimoimaan ja **min** minimoimaan hyötyfunktion arvon
- ★ Alkutila ja seuraajaftio määräävät *pelipuun*, jossa pelaajat vuorotellen valitsevat kuljettavan kaaren

147

- ★ Kahden optimaalisen pelaajan välinen pelinkulku määräytyy täysin minimax-arvojen mukaan
- ★ Pelaajan **max** kannalta minimax-arvot määräävät pahimman tapauksen tuloksen — vastustaja **min** on optimaalinen
- ★ Jos vastustaja ei teekään parhaita mahdollisia siirtoja, niin **max** pärjää vähintään yhtä hyvin
- ★ Ei-optimaalista vastustajaa vastaan jokin muu strategia kuin minimax voi johtaa parempaan lopputulokseen
- ★ Minimax-algoritmi käy läpi koko pelipuun, joten sen aikavaativuus on  $O(b^d)$ , missä  $b$  siirtojen lkm kussakin vaiheessa ja  $d$  on puun maksimisyyvyys
- ★ Eksponentiaalinen aika on liikaa oikeissa pelipuissa

149

- ★ Juuren minimax-arvo ei riipu lehtien  $x$  ja  $y$  arvosta
- ★ Yleisemmin karsinnan periaate on:
  - ◇ pohdittaessa solmuun  $n$  siirtymistä,
  - ◇ jos pelaajalla on parempi valinta  $m$  tehtävänään joko  $n$ :n isäsolmussa tai missä tahansa pelipuun ylemmällä tasolla,
  - ◇ niin solmua  $n$  ei koskaan valita pelissä
- ★ Alpha-beta-karsinnan nimi tulee ylläpi-detyistä muuttujista
  - ◇  $\alpha$  = parhaan (korkeimman) tava-tun valintamahdollisuuden arvo pelaajan **max** polulla
  - ◇  $\beta$  = parhaan (pienimmän) tavatun valintamahdollisuuden arvo pelaajan **min** polulla

151

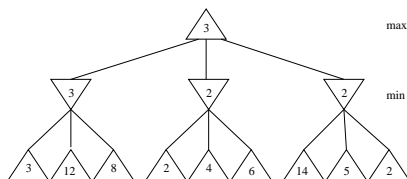
### 3.2 Kahden pelaajan pelit

- ★ Nyt ei enää tarkastella pelkkää po-lunetsintää alkutilasta maalitilaan
- ★ Myös vastustaja saa tehdä tilasiir-tymiä ja pyrkii suistamaan meidät hyvältä polulta
- ★ Tavoitteena on etsiä *siirtostrategia*, joka johtaa maalitilaan riippumatta vastustajan vastasiirroista
- ★ Kahden pelaajan deterministiset, vuorottaiset, täydellisen informaation, *nollasumma* (zero sum) (lau-ta)pelit
- ★ Pelin lopussa toinen pelaajista on hävinnyt ja toinen voittanut

146

- ★ Parhaan mahdollisen pelistrategian etsimisessä oletamme vastustajankin olevan erehtymätön
- ★ Pelaaja **min** valitsee kannaltaan parhaat siirrot
- ★ Optimaalisen strategian määrittämiseksi kullekin solmulle  $n$  lasketaan *minimax-arvo*  $MM(n) =$

$$\begin{cases} \text{hyöty}(n) & n \text{ on loppusolmu} \\ \max_{s \in S(n)} MM(s) & n \text{ on max-solmu} \\ \min_{s \in S(n)} MM(s) & n \text{ on min-solmu} \end{cases}$$



148

### Alpha-beta-karsinta

- ★ Minimax-strategian eksponentiaalista vaativuutta voidaan helpottaa karsi-malla pelipuun evaluoitavia solmuja
- ★ Oikea minimax-päätös voidaan laskea tutkimatta kaikkia puun solmuja
- ★ Esim. edellä olleen puun minimax-arvon määrittämiseksi kaksi lehteä voi-daan jättää tutkimatta, koska  $MM(\text{juuri})$ 

$$\begin{aligned} &= \max(\min(3, 12, 8), \min(2, x, y), \\ &\quad \min(14, 5, 2)) \\ &= \max(3, \min(2, x, y), 2) \\ &= \max(3, z, 2), \text{ missä } z \leq 2 \\ &= 3 \end{aligned}$$

150

- ★ Muuttujien  $\alpha$  ja  $\beta$  arvoja päivitetään puuta läpikäydessä
- ★ Heti kun solmun minimax-arvon tiedetään jäävän heikommaksi kuin  $\alpha$  (**max**) tai  $\beta$  (**min**) jäljellä olevat haarat voidaan karsia
- ★ Alpha-beta-karsinnan tehokkuus riippuu oleellisesti seuraajasolmujen tutkimisjärjestyksestä
- ★ Esimerkkipuun viimeisestä haara-sta ei voitu karsia yhtään seuraajaa, koska minimoinnin kannalta huo-noimmat seuraajat generoitiin ensin
- ★ Jos viimeinen lehti olisi generoitu ennen sisariaan, ne olisi molemmat voitu karsia

152

- ★ Jos parhaat seuraajat pystyttäisiin tutkimaan ensin, putoaisi minimax-haun  $O(b^d)$  aikavaativuus luokkaan  $O(b^{d/2})$
- ★ Efektiivisesti siis haarautumisaste  $b$  putoaisi  $\sqrt{b}$ :hen
- ★ Esim. shakissa tämä tarkoittaa käytännössä astetta 6 alkuperäisen 35:n sijaan
- ★ Peleissä ei voida evaluoida kokonaisia hakupuita, vaan niissä pyritään evaluoimaan osittaisia hakupuita mahdollisimman monta siirtoa eteenpäin
- ★ Alpha-beta-haku siis pystyisi tutkimaan hakupuuta suunnilleen kaksi kertaa niin pitkälle kuin minimax-haku samassa ajassa

153

## Tosiaikaiset päätökset

- ★ Koko pelipuun läpikäyminen millä tahansa realistisella pelillä on liian hidasta
- ★ Pelipuun generoiminen on katkaistava johonkin syvyyteen ja absoluuttiset lopputilojen arvot on korvattava heuristisilla arvioilla
- ★ Pelitilanteita on siis arvoitettava sen mukaan kuinka hyviltä ne vaikuttavat (maalitilaan pääsemisen kannalta)
- ★ Perusvaatimus heuristiselle evaluointifunktioille on, että se arvottaa lopputilanteet oikeaan järjestykseen
- ★ Tietysti pelitilanteiden evaluointi ei saa olla liian hidasta ja arvioiden tulisi vastata tilanteiden oikeaa hyödyllisyyttä

154

- ★ Heuristinen evaluointifunktio perustuu usein peliaseman *piirteiden* (feature) huomioimiseen
- ★ Esimerkiksi shakissa yksi piirre voisi olla kummankin pelaajan sotamiesten lukumäärä laudalla
- ★ Kun peliasetelma typistetään valittujen piirteiden arvoiksi, erilaiset pelitilanteet voivat näyttää ekvivalenteilta, vaikka osa niistä johtaakin voittoon, osa tasapeliin ja osa tappioon
- ★ Tällaisessa ekvivalenssiluokassa voimme laskea odotusarvoisen tuloksen
- ★ Jos esim. 72% peleistä päättyy voittoon (+1), 20% tappioon (-1) ja 8% tasapeliin, niin tästä piirrearvojen asetuksesta jatkuvan pelin hyödyn odotusarvo on:  

$$0,72 \times 1 + 0,20 \times -1 + 0,08 \times 0 = 0,52$$

155

- ★ Koska piirteitä ja niiden mahdollisia arvoja yleensä on paljon, ei tällaista kategorioiden perustuvaa menetelmää usein voida käyttää
  - ★ Sen sijaan kukin piirre  $f_i$  usein arvotetaan yksinään ja pelitilanteen  $s$  evaluoinniksi otetaan valittujen piirteiden painotettu lineaarikombinaatio
- $$eval(s) = \sum_{i=1}^n w_i f_i(s)$$
- ★ Esim. shakissa piirteitä  $f_i$  voisivat olla sotilaiden, lähettien, tornien ja kuningattarien lkm
  - ★ Näiden piirteiden painoja  $w_i$  puolestaan olisivat nappuloiden materiaaliset arvot (1, 3, 5 ja 9)

156

- ★ Piirteiden lineaarikombinaatioon sisältyy ajatus piirteiden *riippumattomuudesta*
- ★ Kuitenkin esim. shakissa lähetit ovat loppupelissä, kun laudalla on paljon tilaa, vahvoja nappuloita
- ★ Siksi nykyiset shakkiohjelmat sisältävätkin myös piirteiden *epälineaarisia* kombinaatioita
- ★ Esim. lähettiä on painoarvoltaan enemmän kuin kaksi kertaa yhden lähetin paino ja loppupelissä lähetin arvoa nostetaan
- ★ Jos eri nappuloiden painoarvoilla ei ole vuosisatojen kokemuksen tuomaa taustaa kuten shakissa, voidaan tällaisia sääntöjä pyrkiä oppimaan koneoppimisen menetelmin

157

## Sattuman huomioiminen

- ★ Peleihin sattuma voidaan lisätä eksplisiittisesti esim. nopanheitolla
- ★ Lautapeleistä ainakin backgammon on tällainen
- ★ Vaikka pelaaja nyt tuntee omat siirtonsa, hän ei tiedä vastustajan pyöräyttämiä silmäluukuja eikä täten vastustajan sallittuja siirtoja
- ★ Tavallista pelipuuta ei siis voida muodostaa
- ★ Pelipuuhun voidaan lisätä max- ja min-solmujen lisäksi sattumasolmuja, joista johtavat kaaret nimitään nopanheiton mahdollisilla tuloksilla ja niiden todennäköisyyksillä

158

- ★ Esim. backgammonissa heitetään kahta noppaa kerrallaan, joten erilaisia parimahdollisuuksia on 6+15 ja niiden tn.:t ovat 1/36 ja 1/18
- ★ Tarkkojen minimax-arvojen sijaan meidän on nyt tyydyttävä odotusarvon laskemiseen
- ★ Esim. pelipuun max-solmun  $n$  arvo määrätään nyt  $= \max_{s \in S(n)} \mathbf{E}[MM(s)]$
- ★ Sattumasolmussa  $n$  lasketaan kaikkien nopanheittojen todennäköisyydellä painotettu keskiarvo  $= \sum_{s \in S(n)} \mathbf{P}(s) \cdot \mathbf{E}[MM(s)]$
- ★ Tilanteiden arvottaminen sattuman vallitessa on herkempi tehtävä kuin deterministisessä pelissä

159

- ★ Sattuman mukaantuominen nostaa pelipuun läpikäynnin vaativuuden luokkaan  $O(b^m n^m)$ , missä  $n$  on nopanheiton mahdollisten tulosten lkm
- ★ Backgammonissa  $n = 21$  ja  $b$  on yleensä noin 20, mutta voi olla jopa 4000 kun nopanheiton tulos on pari
- ★ Vaikka laskentasyvyys olisi rajoitettu, niin sattuman vallitessa ei pelejä voi laskea pitkälle eteenpäin
- ★ Alpha-beta-karsinnan voi nähdä keskityvän todennäköisiin puun haaroihin
- ★ Sattumanvaraisessa pelissä ei ole todennäköisiä siirtojonoja
- ★ Kuitenkin jos hyötyarvot ovat rajoitettuja, niin myös sattumasolmuja sisältävässä pelipuussa voidaan tehdä karsintaa

160

## Deep Blue

- ★ IBM:llä kehitetty rinnakkaislaitteistoon perustuva shakkiohjelmisto, joka vuonna 1997 voitti maailmanmestari Garry Kasparovin kuuden pelin ottelussa
- ★ Hakunopeus keskim. 126 miljoonaa solmua sekunnissa (max 330 miljoonaa)
- ★ Normaali hakusyvyys pelipuuissa: 14
- ★ Perusmenetelmä on iteratiivisesti syvenevä alpha-beta haku
- ★ Menestyksen salaisuus mahdollisuus nähdä hakusyvyyttä pidemmälle joisain riittävän mielenkiintoisissa tapauksissa (jopa 40 puolisiirtoa)
- ★ Evaluointiftiossa 8000 pürrettä
- ★ Laajat alku- ja loppupelikirjastot

161

- ★ Suunnittelussa tietämyksen esitys mahdollistaa tapausriippumattoman heuristiikan käytön
- ★ Myös ongelmien jakamista (lähes) riippumattomiin osaongelmiin voidaan hyödyntää suunnittelussa
- ★ Suunnittelun ei tarvitse edetä lineaarisesti alkutilasta maalitilaan, vaan osaongelmien ratkaisuja voidaan lomittaa
- ★ Keskeinen tekijä suunnittelun onnistumisessa on tietämyksen esitys
- ★ Valitun kielen tulisi olla riittävän ilmaisuvoimainen ollakseen yleiskäyttöinen, mutta toisaalta riittävän rajoittunut tehokkaan suunnittelun takaamiseksi

163

## STRIPS-kieli

- ★ STRIPS = Stanford Research Institute Problem Solver
  - ★ Klassinen suunnitteluympäristö = täysin havainnoitava, deterministinen, äärellinen, staattinen ja diskreetti
  - ★ Tilojen esittämiseen käytetään propositionaalisia predikaattiliteraalien konjunktioita
- Kentällä(Kone<sub>1</sub>, Helsinki) ∧ ...
- ★ Literaalit eivät saa sisältää sitomattomia muuttujia eikä funktioita
  - ★ Muuttujien eksistentiaalinen kvantifiointi on sallittua
  - ★ Suljetun maailman oletuksen perusteella kaikkien mainitsemattomien literaalien oletetaan olevan epätosia

164

- ★ Maalitila on positiivisten literaalien konjunktio, jossa ei ole sitomattomia muuttujia:  $Rikas \wedge Kuuluisa$
- ★ Tila  $s$  toteuttaa maalin  $g$ , jos se sisältää kaikki  $g$ :n literaalit:  $Rikas \wedge Kuuluisa \wedge Onneton$
- ★ Toiminnot määritellään ennakkoehtojen ja vaikutusten avulla:

$Action( Fly(p, from, to)$   
 $Precond : At(p, from) \wedge Plane(p) \wedge Airport(from) \wedge Airport(to)$   
 $Effect : \neg At(p, from) \wedge At(p, to)$

- ★ Tämä on erikoistapaus *toimintoskeemasta*
- ★ Seuraukset voidaan ilmoittaa listaamalla erikseen todeksi muuttuvat literaalit ja epätodeksi muuttuvat literaalit

165

- ★ Toimintoa voidaan soveltaa, jos sen ennakkoehdot toteutuvat

$At(P_1, JFK) \wedge At(P_2, SFO) \wedge Plane(P_1) \wedge Plane(P_2) \wedge Airport(JFK) \wedge Airport(SFO)$

- ★ toteuttaa sijoituksella  $\{ p/P_1, from/JFK, to/SFO \}$  toiminnon  $Fly(p, from, to)$  ennakkoehdot, joten  $Fly(P_1, JFK, SFO)$  on sovellettavissa
- ★ Koska STRIPS-kielessä ei ole funktiosymboleja, on mikä tahansa toimintoskeema muutettavissa propositionaalisiksi
- ★ Esimerkiksi lentokuljetusongelmassa, jossa koneita on 10 ja kenttiä 5, toiminto  $Fly(p, from, to)$  voitaisiin muuttaa  $10 \times 5 \times 5 = 250$  propositionaaliseksi toiminnoksi

166

## Palikkamaailma

$Init ( On(A, Table) \wedge On(B, Table) \wedge On(C, Table) \wedge Block(A) \wedge Block(B) \wedge Block(C) \wedge Clear(A) \wedge Clear(B) \wedge Clear(C)$   
 $Goal ( On(A, B) \wedge On(B, C))$

$Action ( Move(b, x, y)$   
 $Precond : On(b, x) \wedge Clear(b) \wedge Clear(y) \wedge Block(b) \wedge (b \neq x) \wedge (b \neq y) \wedge (x \neq y)$   
 $Effect : On(b, y) \wedge Clear(x) \wedge \neg On(b, x) \wedge \neg Clear(y)$

$Action ( MoveToTable(b, x)$   
 $Precond : On(b, x) \wedge Clear(b) \wedge Block(b) \wedge (b \neq x)$   
 $Effect : On(b, Table) \wedge Clear(x) \wedge \neg On(b, x)$

167

## Haku tila-avaruudessa

- ★ Suoraviivaisin suunnittelumenetelmä on haku tila-avaruudessa
- ★ Etsintä voi edetä *progressiivisesti* alkutilasta eteenpäin
- ★ Koska funktiosymbolien puuttumisen myötä tila-avaruus on äärellinen, niin esim. A\* on täydellinen suunnittelumenetelmä
- ★ Irrelevanttien toimintojen myötä suunnittelusta tulee tehottomaa, jollei heuristinen funktio ole riittävän hyvin määritelty
- ★ Toisaalta haku voi edetä *regressiivisesti* maalitilasta taaksepäin
- ★ Tällöin vain relevantit toiminnot on huomioitava

168

## 4. Toiminnan suunnittelu

- ★ Tavoitteenamme on valita sarja toimintoja, jotka johtavat agentin alkutilasta maalitilaan
- ★ Näinhän hakualgoritmitkin toimivat
- ★ Hakualgoritmit eivät skaalautu tosielämän suuriin ongelmiin
- ★ Tavoitteen kannalta irrelevantit toimintomahdollisuudet lisäävät ongelman kompleksisuutta
- ★ Taustatietämys auttaa karsimaan turhia hakupolkuja
- ★ Hakualgoritmile heuristinen funktio on määriteltävä ongelmakohtaisesti uudelleen ja huolellisesti

162

- ★ Maalin edeltäjätalalta vaaditaan sopivan toiminnon esiehtojen toteuttamista ja voimaantulevan maalin konjunktin lisäksi kaikkien muiden toteuttamista

$$On(A, x) \wedge Clear(A) \wedge Clear(B) \wedge Block(A) \wedge (A \neq x) \wedge (A \neq B) \wedge (x \neq B) \wedge On(B, C)$$

- ★ Lisäksi tietyistä tarkastellun konjunktin ei tulisi olla ennalta voimassa

$$On(A, B),$$

sillä muuten toiminto on irrelevantti

- ★ Toiminnon suorittaminen ei saisi myöskään muuttaa yhtään tavoiteluista literaaleista epätodeksi

169

- ★ Suunnitelma koostuu toiminnoista. Lähtötilanteessa suunnitelmassa on vain toiminnot
  - ◇ *Start*: ei ennakkoehtoja, vaikutuksena suunnitteluongelman alkutilanne
  - ◇ *Finish*: ei vaikutuksia, ennakkoehtoina maalin literaalit

- ★ *Järjestysrajoite*  $A \prec B$ : toiminto  $A$  on suoritettava ennen  $B$ :tä. Näiden on muodostettava aito osittaisjärjestys
- ★ Toimintojen  $A$  ja  $B$  *kausallinen yhteys*  $A \xrightarrow{p} B$ :  $A$  aikaansaa  $p$ :n  $B$ :tä varten
- ★ Esim. *RightSock*  $\xrightarrow{RightSockOn}$  *RightShoe* takaa, että oikean sukan pukemisen jälkeen se on jalassa aina oikean kengän pukemiseen asti
- ★ Jos toiminnot eivät ole toteuttaneet ennakkoehtoja, niin ne pysyvät avoimina. Näiden määrä pyritään supistamaan nollaan luomatta ristiriitoja

171

## Osittainjärjestetty suunnittelu

- ★ Tila-avaruuden etsintä eteen- ja taaksepäin ovat molemmat täysin järjestettyjen suunnitelmien hakuja
- ★ Toiminnot valitaan kronologisessa järjestyksessä
- ★ Niissä ei voi käyttää hyväksi ongelman jakoa osaongelmiin
- ★ Olisi hyödyllistä voida ratkaista osaongelmia riippumattomasti osasuunnitelmin, jotka lopulta voidaan yhdistää
- ★ Tällöin suunnitelma voidaan rakentaa halutussa järjestyksessä, esim. kriittiset päätökset ensin

170

- ★ Suunnitelma on *konsistentti*, jos järjestysrajoitteissa ei ole syklejä eikä kausaaliin yhteyksiin ole konflikteja
- ★ Konsistentti suunnitelma, jossa ei ole avoimia ennakkoehtoja on *ratkaisu*
- ★ Jokainen osittainjärjestetyn ratkaisun linearisointi on täysin järjestetty ratkaisu, jonka suoritus alkutilasta johtaa maalitilaan
- ★ Osittainjärjestetyn suunnitelman suorittamiseksi kullakin hetkellä valitaan mikä tahansa sallituista seuraavista toiminnoista
- ★ Näin saavutettava joustavuus on käytännössä erittäin arvokasta

172

## POP-algoritmi

- ★ Lähtötilanteessa suunnitelmassa on toiminnot *Start* ja *Finish*, järjestysrajoite  $Start \prec Finish$ , ei kausaalisia yhteyksiä sekä kaikki toiminnon *Finish* ennakkoehdot avoimina
- ★ Seuraajaftio valitsee mv. avoimen ennakkoehdon  $p$  toiminnolle  $B$  ja tuottaa seuraajasuunnitelman kaikille mahdollisille konsistenteille tavoille valita toiminto  $A$  joka saavuttaa  $p$ :n
- ★ Konsistenssin varmistamiseksi
  - ◇ suunnitelmaan lisätään yhteys  $A \xrightarrow{p} B$  ja järjestys  $A \prec B$ . Jos  $A$  on uusi toiminto, niin se ja  $Start \prec A$  sekä  $A \prec Finish$  lisätään suunnitelmaan
  - ◇ Uuden kausaalisien yhteyden ja

173

olemassaolevien toimintojen väliset konfliktit, sekä jos  $A$  on uusi toiminto, niin sen ja olemassaolevien yhteyksien väliset konfliktit, ratkaistaan

Konflikti yhteyden  $A \xrightarrow{p} B$  ja toiminnon  $C$  välillä ratkaistaan siirtämällä  $C$ :n suoritus suojatun intervallin ulkopuolelle:  $B \prec C$  tai  $C \prec A$

Jos tuloksena on konsistentti suunnitelma, niin toiselle tai molemmille toiminnoille lisätään seuraajatilat

- ★ Koska menetelmä tuottaa konsistentteja suunnitelmia, niin sen tarkistamiseksi, että kyseessä on suunnitteluongelman ratkaisu, riittää tarkistaa, ettei ole olemassa avoimia ennakkoehtoja

174

- ★ Menetelmän tarkastelemat toiminnot ovat suunnitelman hienonnuksia, eivät yhsinomaan tehtävän varsinaisia toimintoja
- ★ Näin ollen polkukustannuksilla ei ole merkitystä
- ★ Jos kuitenkin kustakin suunnitelmaan lisätystä todellisesta toiminnosta laskutetaan 1 ja muista suunnitelman hienonnuksista 0, niin suunnitelman kustannus vastaa polun todellista kustannusta
- ★ Myös heuristisia suunnitelman kustannusarvioita voidaan tällöin käyttää

175

## Vararenkaan vaihto

*Init*( $At(Flat, Axle) \wedge At(Spare, Trunk)$ )  
*Goal*( $At(Spare, Axle)$ )

*Action*(*Remove*( $Spare, Trunk$ ))  
*Precond*:  $At(Spare, Trunk)$   
*Effect*:  $\neg At(Spare, Trunk) \wedge At(Spare, Ground)$   
*Action*(*Remove*( $Flat, Axle$ ))  
*Precond*:  $At(Flat, Axle)$   
*Effect*:  $\neg At(Flat, Axle) \wedge At(Flat, Ground)$

*Action*(*PutOn*( $Spare, Axle$ ))  
*Precond*:  $At(Spare, Ground) \wedge \neg At(Flat, Axle)$   
*Effect*:  $\neg At(Spare, Ground) \wedge At(Spare, Axle)$

*Action*(*LeaveOvernight*)  
*Precond*:  
*Effect*:  $\neg At(Spare, Ground) \wedge \neg At(Spare, Axle) \wedge \neg At(Spare, Trunk) \wedge \neg At(Flat, Ground) \wedge \neg At(Flat, Axle)$

176

- ★ Lähtötilanteessa toiminnon *Start* vaikutus on  $At(Flat,Axle) \wedge At(Spare,Trunk)$  ja *Finish*'n ennakkoehto on  $At(Spare,Axle)$
- ★ Ainut avoin ennakkoehto valitaan, vain toimintoa  $PutOn(Spare,Axle)$  voidaan soveltaa
- ★ Valitaan avoimista ennakkoehdoista  $At(Spare,Ground)$ , jolloin ainut valittavissa oleva toiminto on  $Remove(Spare,Trunk)$
- ★ Avoimen ennakkoehdon  $\neg At(Flat,Axle)$  toteuttamiseksi valitsemmekin nyt toiminnon  $LeaveOvernight$
- ★ Seurauksiin kuuluu myös  $\neg At(Spare,Ground)$ , joka on konfliktissa seur. kausaalisen yhteyden kanssa

$Remove(Spare,Trunk) \xrightarrow{At(Spare,Ground)}$   
 $PutOn(Spare,Axle)$

177

- ★ Avoimen ennakkoehdon  $\neg At(Flat,Axle)$  toteuttamiseksi valitaan toiminto  $Remove(Flat,Axle)$
- ★ Ennakkoehdon  $At(Spare,Trunk)$  toteuttamiseksi valitaan nyt  $Start$  ja tällä kertaa ei ole konflikteja
- ★ Lopulta ennakkoehdon  $At(Flat,Axle)$  toteuttamiseksi valitaan myös  $Start$
- ★ Näin on muodostunut täydellinen, konsistentti suunnitelma; ts. ratkaisu annettuun ongelmaan
- ★ Suunnitelman osittaisjärjestys käsittelee mahdollisuuden poistaa ensin puhjennut rengas tai ottaa ensin esiin vararengas, jotka kummatkin on suoritettava ennen vararengaan asentamista

181

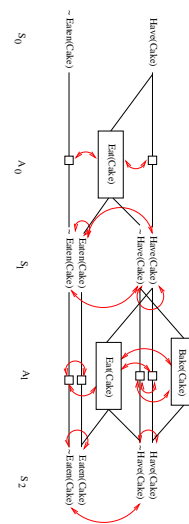
## Suunnitteluverkot

- ★ A. Blum & M. Furst (1995, 1997)
- ★ Verkossa on tasoja, jotka vastaavat suunnitelman aika-askeleita
- ★ Kukin taso käsittää literaaleja, jotka voisivat olla tosia, ja toimintoja, joiden esiehdot voisivat olla täytettyjä

$Init(Have(Cake))$   
 $Goal(Have(Cake) \wedge Eaten(Cake))$

$Action(Eat(Cake))$   
 $Precond: Have(Cake)$   
 $Effect: \neg Have(Cake) \wedge Eaten(Cake)$   
 $Action(Bake(Cake))$   
 $Precond: \neg Have(Cake)$   
 $Effect: Have(Cake)$

180



- ★ Tilataso  $S_0$  vastaa ongelman alkutilaa
- ★ Toimintotasolla  $A_0$  on kaikki ne toiminnot, joiden ennakkoehdot edellinen taso täytti
- ★ Jokainen toiminto on kytketty ennakkoehtoihinsa tasolla  $S_0$  ja vaikutuksiinsa tasolla  $S_1$
- ★ Tasolle  $S_1$  tulee siis literaaleja, joita ei ollut tasolla  $S_0$
- ★ Verkkoon lisätään myös *persistenssi* toimintoja, joiden avulla literaalit "elävät" tasolta toiselle
- ★ Tasoja muodostetaan kunnes kaksi peräkkäistä tasoa ovat identtiset, jonka jälkeen verkon tasot eivät enää muutu

182

- ★ Toimintojen negatiivisista vuorovaiikutuksista vain osa saa esityksen verkossa
- ★ Verkkoon perustuva arvio literaalien toteuttamiseksi vaadittavien askelten lukumäärälle saattaa siksi olla optimistinen
- ★ Askelten määrä verkossa antaa kuitenkin hyvän arvion sille kuinka vaikeaa jonkin literaalien toteuttaminen on alkutilasta lähtien
- ★ Lisäksi suunnitteluverkko on muodostettavissa tehokkaasti
- ★ Suunnitteluverkot ovat sovellettavissa vain propositionaalisii suunnitteluongelmiin

183

- ★ Konfliktin ratkaisemiseksi vaaditaan toiminnon  $LeaveOvernight$  edeltävän toimintoa  $Remove(Spare,Trunk)$
- ★ Ainut jäljellä oleva avoin ennakkoehto on  $At(Spare,Trunk)$  ja sen ainut mahdollinen toteuttaja on  $Start$  toiminto
- ★  $Start$ in ja  $Remove(Spare,Trunk)$  toiminnon välinen kausaalinen yhteys on kuitenkin konfliktissa toiminnon  $LeaveOvernight$  vaikutuksen  $\neg At(Spare,Trunk)$  kanssa
- ★ Konfliktia ei voi selvittää, koska  $LeaveOvernight$ in on edeltävä  $Remove(Spare,Trunk)$  toimintoa ja seurattava  $Start$ ia
- ★ Suunnittelussa on siis peruutettava

178

- ★ Sekä toiminnoille että literaaleille lasketaan *poissulkemis- l. mutex-suhteita*
- ★ Esimerkiksi toiminto  $Eat(Cake)$  on poissulkeva tapahtumien  $Have(Cake)$  ja  $\neg Eaten(Cake)$  persistenssille
- ★ Literaaleillekin lasketaan mitkä niistä eivät voi esiintyä yhdessä riippumatta valituista toiminnoista
- ★ Esim.  $Have(Cake)$  ja  $Eaten(Cake)$  ovat mutex-suhteessa: riippuen tasolla  $A_0$  valituista toiminnoista jompikumpi, eivät molemmat, voi olla seuraus
- ★ Lopulta verkossa kukin  $A_i$  taso kertoo, mitkä toiminnot ovat mahdollisia tilassa  $S_i$  ja antaa rajoitteita sille mitkä toiminnot voidaan suorittaa yhdessä

184

- ★ Jokainen  $S_i$  taso sisältää kaikki literaalit, jotka voivat seurata tason  $A_{i-1}$  toimintovalinnoista, sekä antaa rajoitteita, jotka kertovat mitkä literaaliparit eivät ole mahdollisia
- ★ Suunnitteluverkon rakentaminen ei pakota valitsemaan toimintoja
- ★ Verkko vain tallettaa tiedon joidenkin valintojen mahdottomuudesta mutex-suhtein
- ★ Jos (litteraalien lkm:n suhteen) eksponentiaalisessa tila-avaruuudessa jouduttaisiin etsimään, ei menetelmä olisi tehokas
- ★ Nyt verkon konstruoinnin aikavaativuus on matala-asteista polynomiaalista luokkaa

185

- ★ Kaksi toimintoa sulkevat toisensa pois, jos
  - ◊ niiden vaikutukset ovat ristiriitaisia; esim. *Eat(Cake)* on poissulkeva literaalin *Have(Cake)* persistenssin kanssa, koska ne ovat inkonsistentteja *Have(Cake)*:n suhteen,
  - ◊ yhden toiminnon vaikutus on toisen ennakkoehdon negaatio; esim. *Eat(Cake)* häiritsee *Have(Cake)*:a negatoimalla sen ennakkoehdon,
  - ◊ yhden toiminnon ennakkoehto on poissulkeva toisen toiminnon ennakkoehdon kanssa; esim. *Bake(Cake)* ja *Eat(Cake)* kilpailevat keskenään arvosta *Have(Cake)*

186

- ★ Kaksi saman tason literaalia ovat mutex-suhteessa, jos
  - ◊ yksi on toisen negaatio tai
  - ◊ kukin literaalit mahdollisesti toteuttavien toimintojen pari on toisensa poissulkeva
- ★ Esim. tasolla  $S_1$  *Have(Cake)* ja *Eaten(Cake)* ovat toisensa poissulkevia, koska ainut *Have(Cake)*:n toteuttava toiminto literaalien persistenssi ja *Eat(Cake)* toisen literaalien toteuttamiseksi ovat mutex-suhteessa
- ★ Sen sijaan tasolla  $S_2$  ne eivät enää ole toisensa poissulkevia, koska esim. *Bake(Cake)* ja *Eaten(Cake)*:n persistenssi eivät ole mutex-suhteessa

187

- ★ Literaalia, joka ei esiinny verkon viimeisellä tasolla ei voi saavuttaa millään suunnitelmalla
- ★ Yleisemmin literaalien toteuttamisen kustannukseksi voidaan arvioida se taso, jolla literaali ensikerran esiintyy verkossa
- ★ Nämä arviot eivät koskaan yliarvioi maalilitteraalien saavuttamisen kustannusta
- ★ Koska verkossa kuitenkin sallitaan useita toimintoja kullakin tasolla, voivat arviot olla reippaita aliarvioita
- ★ Sarjallisessa suunnitteluverkossa sallitaan vain yhden toiminnon toteuttaminen kullakin tasolla (mutex-suhteilla)
- ★ Näin saatavat arviot ovat realistisempia kustannusarviota

188

## Graphplan-algoritmi

- ★ Haetaan suunnitelmaa suoraan suunnitteluverkosta
- ★ Viimeisimmällä tilatasolla tarkastetaan kuuluvatko kaikki maaliliteraalit siihen ilman mutex-suhteita minkään parin välillä
- ★ Jos näin on verkko voi sisältää ratkaisun ongelmaan ja se yritetään muodostaa
- ★ Muutoin verkkoon lisätään nykytason toiminnot ja seuraava tilataso
- ★ Näin jatketaan kunnes ratkaisu löytyy tai voidaan todeta, ettei ratkaisua ole olemassa
- ★ Ratkaisun muodostusyritys perustuu löydetyn ratkaisumahdollisuuden taaksepäin etsimiseen aina tasolle  $S_0$  asti

189

- ★ Koska suunnittelu yleensä on PSPACE-täydellistä ja suunnitteluverkon muodostaminen polynomiaikaisista  $\Rightarrow$  ratkaisun muodostaminen verkosta on laskennallisesti vaativaa pahimman tapauksen analyysin mielessä
- ★ Taaksepäin ketjutuksessa heuristiset toimintojen valintakriteerit ovat tarpeen
- ★ Suunnitteluverkon rakentaminen päättyy, jos ongelmalla ei ole ratkaisua
- ★ Tämä seuraa siitä, että literaalit ja toiminnot lisääntyvät monotonisesti tasolta tasolle kun taas mutex-suhteet vähenevät monotonisesti
- ★ Koska toimitoja ja literaaleja on äärellinen määrä, niin lopulta peräkkäiset tasot ovat identtisiä

190

## SATplan-algoritmi

- ★ Muotoillaan suunnitteluongelma loogiseksi kaavaksi, jonka toteutuvuutta testataan
- ★ alkutila  $\wedge$  kaikkien mahdollisten toimintojen kuvaukset  $\wedge$  maali
- ★ Lauseen toteuttava malli asettaa todeksi oikeaan suunnitelmaan kuuluvat toiminnot ja epätodeksi kaikki muut
- ★ Virheellistä suunnitelmaa vastaava arvoasetus ei voi olla konsistentti maalin kanssa
- ★ Tarkastellaan lentokone esimerkkiä: lähtötilanteessa kone  $P_1$  on kentällä *SFO* ja kone  $P_2$  kentällä *JFK*:

$$At(P_1, SFO)^0 \wedge At(P_2, JFK)^0$$

191

- ★ Propositiologiikassa ei ole suljetun maailman oletusta  $\Rightarrow$  on spesifioitava myös ne propositiot, jotka eivät päde:

$$\neg At(P_1, JFK)^0 \wedge \neg At(P_2, SFO)^0$$

- ★ Tavoitteena esimerkissä on vaihtaa koneiden lentokentät
- ★ Myös maalilauseeseen on liitettävä ajanhetki, mutta emme voi tietää etukäteen montako askelta maalin saavuttaminen vaatii
- ★ Maalilauseetta kokeillaan ajanhetkillä  $0, 1, \dots$  kunnes se toteutuu tai ennalta valittu suunnitelman maksimipituus saavutetaan

$$At(P_1, JFK)^0 \wedge At(P_2, SFO)^0$$

192

- ★ Toimintojen kuvaaminen edellyttää seuraajatila-aksoomien muotoilemista kaikille lentokoneille, -kentille ja ajanhetkille:

$$t(P_1, JFK)^1 \Leftrightarrow (At(P_1, JFK)^0 \wedge \neg Fly(P_1, JFK, SFO)^0) \vee (Fly(P_1, SFO, JFK)^0 \wedge At(P_1, SFO)^0)$$

- ★ Arvoasetus, jossa  $Fly(P_1, SFO, JFK)^0$  ja  $Fly(P_2, JFK, SFO)^0$  ovat tosia ja kaikkia muita toimintoja vastaavat symbolit epätosia on malli lauseelle, jossa tutkitaan maalin toteutuvuutta ajanhetkellä  $T = 1$
- ★ Niin myös arvoasetus, jossa lisäksi  $Fly(P_1, JFK, SFO)^0$  on tosi

193

- ★ Seuraajatila-aksooma ei eksplisiittisesti kiellä suorittamista toimintoa, jonka ennakoehdot eivät toteudu

- ★ Kuvaukseen on lisättävä mukaan myös ennakoehdoaksoomat

$$Fly(P_1, JFK, SFO)^0 \Rightarrow At(P_1, JFK)^0$$

- ★ Tämän jälkeen on enää yksi suunnitelma, joka toteutuu ajanhetkellä 1
- ★ Suunnitelmassa on rinnakkaiset toiminnot kuten POP ja Graphplan suunnitelmissa
- ★ Kolmannen lentokentän mukaantuoaminen esimerkkiin osoittaa, että myös samanaikaiset toiminnot on kiellettävä eksplisiittisesti
- ★ Tämän jälkeen suunnitelmat ovat täysin järjestettyjä

194

- ★ Propositionaaliset ongelman kuvaukset sisältävät  $T \times |A| \times |O|^P$  toimintosymbolia, missä  $A$  on toiminnot,  $O$  objektit ja  $P$  toimintoskeeman maks. ariteetti
- ★ Esim. 10 aika-askelen, 12 koneen ja 30 kentän täysin samanaikaiset toiminnot kieltävät aksoomat sisältävät 583 miljoonaa lausetta
- ★ Eksponentiaalisesta riippuvuudesta toimintojen ariteetista voidaan päästä eroon siirtymällä korkeintaan binäärisiin predikaatteihin
- ★ Esim.  $Fly(P_1, SFO, JFK) \Rightarrow Fly_1(P_1, Fly_2(SFO) \text{ ja } Fly_3(JFK))$
- ★ Näin saatava yksinkertaistus pudottaa edellä mainitun 583 miljoonaa lausetta 9360 lauseeseen

195

## 5. Epävarma tietämys ja päättely

- ★ Käytännössä agentilla ei ole koskaan täyttä tietämystä toimintaympäristöstään, vaan se joutuu toimimaan epävarmuuden vallitessa
- ★ Logiikkaan perustuva agentti voi joutua tilanteeseen, jossa se ei saa varmuudella tarvitsemaan tietoja
- ★ Jos agentti ei voi todeta jonkin toimintasuunnitelman toteuttavan sen tavoitetta, niin se ei voi toimia
- ★ Ehdollinen suunnittelu voi helpottaa tilannetta, mutta ei ratkaise sitä
- ★ Yksinomaan logiikkaan perustuva agentti ei kykene valitsemaan rationaalisia toimintoja epävarmassa toimintaympäristössä

196

- ★ Looginen tietämyksen esitys vaatii poikkeuksettomia sääntöjä
- ★ Usein käytännössä voimme taata vain jonkin uskomuksen asteen esitylle väittämälle
- ★ Uskomusten käsittelyssä turvauksumme todennäköisyysteoriaan
- ★ Todennäköisyys 0 on vastaansanomaton usko lauseen epätotuudesta ja 1 usko sen totuuteen
- ★ Väliin jäävät arvot kertovat uskomme lujuudesta lauseen totuudesta, eivät lauseen totuuden suhteellisuudesta
- ★ Tn.:illä painotetut hyötyarvot (utility) antavat agentilla mahdollisuuden rationaaliseen toimintaan hyödyn odotusarvon maksimoinnin kautta

197

## Todennäköisyysteoria

- ★ *Satunnaismuuttuja* (random variable) viittaa maailman osaan, jonka tilannetta ei tunneta ennalta
- ★ Sm.:t vastaavat propositiosymboleita
- ★ Esim. sm. *Reikä* voisi viitata siihen onko vasemmassa alaleuan viisaudenhampaassani reikä
- ★ Sm.:n *arvoalue* voi olla
  - ◇ *totuusarvoinen*: merk.  $Reikä=true \Rightarrow reikä$  ja  $Reikä=false \Rightarrow \neg reikä$ ;
  - ◇ *diskreetti*: esim. sm.:n *Säätila* arvoalue voisi olla  $\langle aurinkoa, sadetta, pilveä, lunta \rangle$ ;
  - ◇ *jatkuva*: tällöin tarkastellaan useinmiten kertymäftiota, esim.  $X \leq 4.02$

198

- ★ Alkeispropositioita kootaan loogisiin konnektiivein monimutkaisemmiksi lauseiksi:  $reikä \wedge \neg hammassärky$
- ★ Atominen  *tapahtuma* (event) on kaikkien muuttujien täydellinen arvoasetus
- ★ Atomisille tapahtumille pätee, että
  - ◇ ne ovat toisensa poissulkevia,
  - ◇ niiden joukko on kattava — ainakin yksi niistä on tosi,
  - ◇ Mikä tahansa atominen tapahtuma määrää kaikkien propositioiden totuuden ja
  - ◇ Mikä tahansa propositio on loogisesti ekvivalentti kaikkien niiden atomisten tapahtumien disjunktion kanssa, joista sen totuus seuraa

$$reikä \equiv (reikä \wedge hammassärky) \vee (reikä \wedge \neg hammassärky)$$

199

- ★ Uskomuksen aste kohdistetaan propositionaalisiiin lauseisiin
- ★ *Prioritodennäköisyys*  $P(a)$  antaa proposition  $a$  uskomuksen asteen muun tiedon puuttuessa
- ★  $P(reikä) = 0.1$
- ★ Erityisesti siis prioritn. on agentin alkuperäinen uskomus, ennen omia havaintoja
- ★ Muuttujan  $X$  *todennäköisyysjakauma*  $\vec{P}(X)$  on sen (järjestetyn) arvoalueen alkioiden todennäköisyyksien vektori
- ★ Esim. kun  $P(aurinkoa) = 0.02$ ,  $P(sadetta) = 0.2$ ,  $P(pilveä) = 0.7$  ja  $P(lunta) = 0.08$ , niin  $\vec{P}(Säätila) = \langle 0.02, 0.2, 0.7, 0.08 \rangle$

200

## Todennäköisyyden Kolmogorov aksioomat

- ★ Todennäköisysteorian aksiomatiointi (1933) kolmen yksinkertaisen aksiooman pohjalta
1. Minkä tahansa proposition  $a$  tn. on välillä 0 ja 1:

$$0 \leq P(a) \leq 1$$

2. Välttämättä toden (validin) proposition tn. on 1 ja välttämättä epätoden (toteutumattoman) proposition tn. on 0:

$$P(\text{true}) = 1 \quad P(\text{false}) = 0$$

3. Disjunktion tn. on

$$P(a \vee b) = P(a) + P(b) - P(a \wedge b)$$

- ★ Aksiomien perusteella voimme päätellä mm. seuraavaa

$$P(a \vee \neg a) = P(a) + P(\neg a) - P(a \wedge \neg a)$$

$$P(\text{true}) = P(a) + P(\neg a) - P(\text{false})$$

$$1 = P(a) + P(\neg a)$$

$$P(\neg a) = 1 - P(a)$$

- ★ Kolmannen rivin sääntö yleistyy diskreetille muuttujalle  $D$ , jonka arvoalue on  $\langle d_1, \dots, d_n \rangle$ :

$$\sum_{i=1}^n P(D = d_i) = 1$$

- ★ Jatkuva-arvoisella muuttujalla  $X$  summa on korvattava integraalilla:

$$\int_{-\infty}^{\infty} P(X = x) dx = 1$$

- ★ Kahden muuttujan yhteistn.jakauma (joint probability distribution) on niiden arvoalueiden tulo

- ★ Esim.  $\vec{P}(\text{Säätilä, Reikä})$  on  $4 \times 2$  taulukko todennäköisyyksiä

- ★ Täysi yhteistn.jakauma on kaikkien maailman kuvaamiseen käytettyjen sm.:ien yhteistnjakauma

- ★ Jatkuva-arvoisille muuttujille ei voida kirjoittaa jakaumataulukkoa, vaan sen sijaan on tarkasteltava todennäköisyyden tiheysfktiä

- ★ Pistetn.:ien (joiden arvo on 0) sijaan tarkastellaan osavälien tn.:iä

- ★ Tarkastelemme enimmäkseen diskreettejä sm.:iä

201

- ★ Kun agentti saavuttaa tietoa ympäristön sm.:ien arvoista, niin prioritn.:ien sijaan on siirryttävä ehdollisiin (posteriori) tn.:iin

- ★  $P(a | b)$  on proposition  $a$  tn., kun kaikki mitä tiedämme on  $b$

- ★  $P(\text{reikä} | \text{hammassärky}) = 0.8$

- ★  $P(\text{reikä}) = P(\text{reikä} | )$

- ★ Kun  $P(b) > 0$ , niin

$$P(a | b) = \frac{P(a \wedge b)}{P(b)},$$

joka voidaan kirjoittaa myös tulosääntönä:  $P(a \wedge b) = P(a | b)P(b)$

- ★ Sääntö voidaan tietysti myös kääntää  $P(a \wedge b) = P(b | a)P(a)$

- ★  $\vec{P}(X | Y) \equiv P(X = x_i | Y = y_j) \forall i, j$

202

- ★ Yksittäisen muuttujan tn.jakauman on siis summauduttava 1:een

- ★ Samoin kaikkien yhteistn.jakaumien

- ★ Propositio  $a$  on ekvivalentti kaikkien niiden atomitapahtumien disjunktion kanssa, joiden looginen seuraus se on

- ★ Merk. näiden atomitapahtumien joukkoa  $e(a)$

- ★ Aksiomian 3 perusteella

$$P(a) = \sum_{e_i \in e(a)} P(e_i)$$

- ★ Jos täysi yhteistn.jakauma on tunnettu, niin voimme tämän perusteella laskea minkä tahansa proposition tn.:in

205

## Probabilistinen päättely täydestä yhteisjakaumasta

	särky		¬särky	
	osuma	¬osuma	osuma	¬osuma
$r$	0.108	0.012	0.072	0.008
$\neg r$	0.016	0.064	0.144	0.576

- ★ Voimme laskea esim. lauseen  $\text{reikä} \vee \text{hammassärky}$  tn.:in  $0.108 + 0.012 + 0.072 + 0.008 + 0.016 + 0.064 = 0.28$

- ★ Muuttujan (tai muuttujajoukon) arvon marginaalitn. saadaan laskemalla vastaavien rivien tai sarakkeiden summat

- ★ Esim.  $P(\text{reikä}) = 0.108 + 0.012 + 0.072 + 0.008 = 0.2$

- ★ Yleisesti mille tahansa muuttujajoukoille  $\vec{Y}$  ja  $\vec{Z}$  pätee marginalisointisääntö

$$\vec{P}(\vec{Y}) = \sum_{\vec{z}} \vec{P}(\vec{Y}, \vec{z})$$

206

- ★ Ehdollisen tn.:n laskeminen

$$P(\text{reikä} | \text{särky}) = \frac{P(\text{reikä} \wedge \text{särky})}{P(\text{särky})} = \frac{0.108 + 0.012}{0.108 + 0.012 + 0.016 + 0.064} = 0.6$$

- ★ Vastaavasti

$$P(\neg \text{reikä} | \text{särky}) = \frac{0.08}{0.2} = 0.4$$

- ★  $1/P(\text{särky}) = 0.2$  on normalisointitekijä, joka varmistaa, että jakauma  $\vec{P}(\text{Reikä} | \text{särky})$  summautuu 1:een

- ★ Merk. normalisointivakiota  $\alpha$ :lla

$$\begin{aligned} \vec{P}(\text{Reikä} | \text{särky}) &= \alpha \vec{P}(\text{Reikä}, \text{särky}) \\ &= \alpha [\vec{P}(\text{Reikä}, \text{särky}, \text{osuma}) + \vec{P}(\text{Reikä}, \text{särky}, \neg \text{osuma})] \\ &= \alpha [(0.108, 0.016) + (0.012, 0.064)] \\ &= \alpha (0.12, 0.08) = (0.6, 0.4) \end{aligned}$$

207

- ★ Yleisemmin: meidän tulee selvittää kyselymuuttujan  $X$  jakauma, todistemuuttujien  $\vec{E}$  havaitut arvot ovat  $\vec{e}$  ja loput havaitsemattomat muuttujat ovat  $\vec{Y}$

- ★ Kyselyn evaluointi:  $\vec{P}(X | \vec{e}) =$

$$\alpha \vec{P}(X, \vec{e}) = \alpha \sum_{\vec{y}} \vec{P}(X, \vec{e}, \vec{y}),$$

missä  $\vec{y}$ :t käyvät yli havaitsemattomien muuttujien kaikkien mahdollisten arvokombinaatioiden

- ★  $\vec{P}(X, \vec{e}, \vec{y})$  on osajoukko kaikkien muuttujien  $X, \vec{E}$  ja  $\vec{Y}$  yhteistn.jakaumasta

- ★ Taulukon koko on  $O(2^n)$  ja sen läpikäymisen aikaavaativuus on  $O(2^n)$ , joten yleisesti ottaen menetelmä ei ole käyttökelpoinen

208

## Muuttujien riippumattomuus

- ★ Jos esimerkiksi lisätään muuttuja *Säätila*, jolla on 4 mahd. arvoa, on edellä ollut yhteisjakauman taulukko monistettava neljäksi
- ★ Koska hammasongelmat eivät vaikuta säätilaan, pätee:

$$P(\text{Sää=pilveä} \mid \text{särky,osuma,reikä}) = P(\text{Sää=pilveä})$$

- ★ Tulosäännön ja yo. havainnon perusteella

$$P(\text{särky,osuma,reikä,Sää=pilveä}) = P(\text{Sää=pilveä})P(\text{särky,osuma,reikä})$$

209

## Bayesin sääntö

- ★ Tulosäännön perusteella  $P(a \wedge b) = P(a \mid b)P(b)$  ja konjunktion kommutatiivisuuden perusteella  $P(a \wedge b) = P(b \mid a)P(a)$
- ★ Oikeat puolet ekvivalisoimalla ja jakamalla  $P(a)$ :lla saadaan *Bayesin sääntö*

$$P(b \mid a) = \frac{P(a \mid b)P(b)}{P(a)}$$

- ★ Yleisemmässä muodossa, muuttujille  $X$  ja  $Y$  ja taustatietämykselle  $\bar{e}$

$$\bar{P}(Y \mid X, \bar{e}) = \frac{\bar{P}(X \mid Y, \bar{e})\bar{P}(Y \mid \bar{e})}{\bar{P}(X \mid \bar{e})}$$

- ★ Normalisointia käyttäen Bayesin sääntö voidaan kirjoittaa muotoon

$$\bar{P}(Y \mid X) = \alpha \bar{P}(X \mid Y)\bar{P}(Y)$$

211

- ★ Sama pätee myös muille muuttujan *Säätila* arvoille, joten

$$\bar{P}(\text{Särky,Osuma,Reikä,Säätila}) = \bar{P}(\text{Säätila})\bar{P}(\text{Särky,Osuma,Reikä})$$

- ★ Tarkasteltavat yhteisjakauma taulukot ovat siis 8 ja 4 alkioiset

- ★ Propositiot  $a$  ja  $b$  ovat *riippumattomia* jos  $P(a \mid b) = P(a) \Leftrightarrow P(b \mid a) = P(b) \Leftrightarrow P(a \wedge b) = P(a)P(b)$

- ★ Vastaavasti muuttujat  $X$  ja  $Y$  ovat toisistaan riippumattomia jos

$$\begin{aligned} \bar{P}(X \mid Y) &= \bar{P}(X) \Leftrightarrow \\ \bar{P}(Y \mid X) &= \bar{P}(Y) \Leftrightarrow \\ \bar{P}(X, Y) &= \bar{P}(X)\bar{P}(Y) \end{aligned}$$

- ★ Riippumattomat kolikonheitot:  $\bar{P}(C_1, \dots, C_n)$ :n sijaan tarkasteltava  $n$ :n  $\bar{P}(C_i)$ :n tuloa

210

- ★ Puolella aivokalvontulehdus potilaista on jäykkä niska
- ★ Aivokalvontulehdusta esiintyy 1/50 000 tapauksessa
- ★ Jäykkää niskaä valittaa joka 20:s potilas
- ★ Mikä on todennäköisyys, että jäykkää niskaansa valittavalla potilaalla on aivokalvontulehdus?

$$\begin{aligned} P(j \mid a) &= 0.5 \\ P(a) &= 1/50000 \\ P(j) &= 1/20 \\ P(a \mid j) &= \frac{P(j \mid a)P(a)}{P(j)} \\ &= \frac{20}{2 \times 50000} = 0.0002 \end{aligned}$$

212

- ★ Bayesin sääntö on kaikkien modernien probabilisten päättelyjärjestelmien pohjana

- ★ Sinänsä yksinkertainen sääntö ei pintapuolisesti katsoen näyttäisi antavan paljon päättelymahdollisuuksia, mutta kuten edellinen esimerkki osoittaa kyse on mahdollisuudesta soveltaa olemassaolevaa tietoa

- ★ Nimittäjässä olevan ehdon tn:n laskemisesta voidaan päästä laskemalla kyselymuuttujan posterioritn. kaikilla sen mahdollisilla arvoilla

- ★ Esim.  $\bar{P}(A \mid j) = \alpha(P(j \mid a)P(a), P(j \mid -a)P(-a))$

- ★ Joskus helpompaa, joskus ei

213

- ★ Ehdollinen riippumattomuus:

$$\begin{aligned} \bar{P}(\text{särky} \wedge \text{osuma} \mid \text{Reikä}) \\ = \bar{P}(\text{särky} \mid \text{Reikä})\bar{P}(\text{osuma} \mid \text{Reikä}) \end{aligned}$$

- ★ Yhdessä Bayes säännön kanssa saamme lopulta

$$\begin{aligned} \bar{P}(\text{Reikä} \mid \text{särky} \wedge \text{osuma}) &= \alpha \bar{P}(\text{Reikä}) \\ \bar{P}(\text{särky} \mid \text{Reikä})\bar{P}(\text{osuma} \mid \text{Reikä}) \end{aligned}$$

- ★ Nyt siis tarvitsee enää käsitellä kolmea erillistä jakaumaa

- ★ Muuttujien  $X$  ja  $Y$  ehdollinen riippumattomuus annettuna  $Z$  on tarkkaan ottaen

$$\bar{P}(X, Y \mid Z) = \bar{P}(X \mid Z)\bar{P}(Y \mid Z)$$

- ★ Ekvivalentisti  $\bar{P}(X \mid Y, Z) = \bar{P}(X \mid Z)$  ja  $\bar{P}(Y \mid X, Z) = \bar{P}(Y \mid Z)$

215

- ★ Kun probabilistinen kysely on ehdollistettu useammalle kuin yhdelle havainnolle

$$\bar{P}(\text{Reikä} \mid \text{särky} \wedge \text{osuma})$$

- ★ Täyteen yhteisjakaumaan perustuva menetelmä ei käy

- ★ Bayesin säännön soveltaminenkaan ei yleisesti ottaen lievennä eksponentiaalista vaativuutta

- ★ Tarvittaisiin muuttujien riippumattomuutta, mutta havaintoja *särky* ja *osuma* vastaavat muuttujat ovat selvästi yhteydessä toisiinsa

- ★ Molempien muuttujien suora vaikuttaja on reikä hampaassa, mutta ilman muuttujaa *Reikä* ne ovat *ehdollisesti riippumattomia* (conditionally independent) — *Särky* ja *Osuma* eivät ole suoraan toisistaan riippuvia

214

- ★ Jos kaikki havaintomuuttujat ovat ehdollisesti riippumattomia annettuna kyselymuuttuja, niin eksponentiaalinen tietämyksenesityksen kasvuvauhti putoaa lineaariseksi

- ★ Tämä on *naiivoin Bayes -mallin* ajatus, kaikki seuraukset  $S_1, \dots, S_n$  ovat ehdollisesti riippumattomia annettuna yksi aiheuttaja  $A$

- ★ Tällöin niiden täysi yhteisjakauma

$$\bar{P}(A, S_1, \dots, S_n) = \bar{P}(A) \prod_i \bar{P}(S_i \mid A)$$

- ★ Mallia käytetään usein yksinkertaistamiseen, vaikka seuraukset eivät olisi ehdollisesti riippumattomia

- ★ Saattaa toimia yllättävän hyvin kaikesta huolimatta

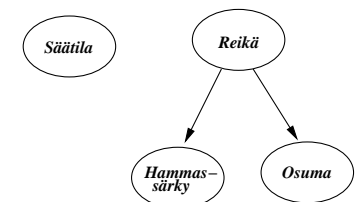
216

## 5.1 Probabilistinen päättely

- \* Bayes-verkko on suunnattu verkko, jonka kuhunkin solmuun liittyy todennäköisyys
1. Satunnaismuuttujien joukko muodostaa verkon solmut. Muuttujat voivat olla diskreetti- tai jatkuva-arvoisia
  2. Solmujen välillä on suunnattuja linkkejä (nuolia). Jos solmusta  $X$  on nuoli solmuun  $Y$ , niin  $X$  on  $Y$ :n vanhempi
  3. Solmulla  $X_i$  on ehdollinen tn.jakauma  $P(X_i | \text{Vanhemmat}(X_i))$
  4. Verkossa ei ole suunnattuja syklejä. Se on siis suunnattu sykliiton verkko DAG

217

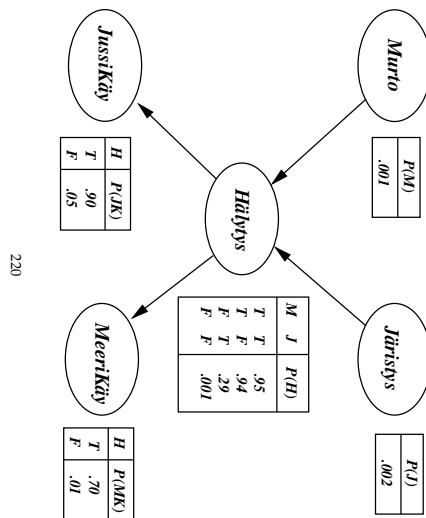
- \* Nuolen  $X \rightarrow Y$  intuitiivinen merkitys on, että  $X$  vaikuttaa suoraan  $Y$ :hyn
- \* Verkon topologia — solmut ja nuolet — määräävät muuttujien ehdolliset riippumattomuudet
- \* Kun topologia on kiinnitetty, pitää vielä määrätä kullekin muuttujalle ehdollinen tn.jakauma annettuna sen vanhemmat
- \* Verkon topologia ja ehdolliset tn.jakaumat määräävät implisiittisesti täyden yhteisjakauman muuttujille



218

- \* Taloon on asennettu varashälytyn, joka on luotettava murtoyritysten tunnistaja, mutta reagoi myös lieviin määnjäristyksiin
- \* Naapurit Jussi ja Meeri ovat luvanneet tarkistaa tilanteen hälytyksen kuullessaan
- \* Jussi kuulee jokaisen hälytyksen, mutta sen lisäksi joskus erehtyy tarkistamaan tilanteen myös puhelimen soidessa
- \* Meeri puolestaan kuuntelee musiikkia lujalla, eikä aina kuule hälytystä
- \* Tavoitteenamme on arvioida murron todennäköisyys annettuna tieto siitä kuka on käynyt tai ei ole käynyt tarkistamassa tilannetta

219



220

- \* Verkon topologiaan sisältyvät seuraavat ajatukset:
  - ◊ Murto ja maanjäritys vaikuttavat suoraan hälyttimen laukeamiseen,
  - ◊ Jussin ja Meerin vierailut riippuvat suoraan vain hälyttimestä
  - ◊ he eivät huomaa murtoa tai maanjäritystä suoraan tai keskustele keskenään
- \* Musiikinkuuntelu ja puhelimen soiminen ovat verkosta luettavissa vain implisiittisesti hälytyksen aiheuttaman tarkistuskäynnin epävarmuutena
- \* Todennäköisyydet tiivistävät äärettömän mahdollisten vaikuttajien joukon
- \* Esim. hälytyn voi jäädä laukeamatta koska ilmankosteus on korkea, on sähkökatko, varapatteri on tyhjentynyt, johdot on katkaistu, kuollut hiiri estää kelloa soimasta, jne.

222

## Bayes-verkon semantiikka

- \* Jokaisen yhteisjakauman solun arvo voidaan laskea Bayes-verkon sisältämästä informaatiosta
- \* Solun arvo on muuttujien tietty arvoasetus  $P(X_1 = x_1 \wedge \dots \wedge X_n = x_n)$ , jota merkitään  $P(x_1, \dots, x_n)$
- \* Solun arvo on

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | \text{vanhemmat}(X_i)),$$

missä  $\text{vanhemmat}(X_i)$  viittaa muuttujien  $\text{Vanhemmat}(X_i)$  määrättyihin arvoihin

$$\begin{aligned} & \star P(JK \wedge MK \wedge H \wedge \neg M \wedge \neg J) \\ &= P(JK | H)P(MK | H) \\ & \quad P(H | \neg M \wedge \neg J)P(\neg M)P(\neg J) \\ &= .90 \times .70 \times .001 \times .999 \times .998 \\ &= .00062 \end{aligned}$$

223

- \* Ehdollisten todennäköisyyksien taulukot luettelevat muuttujan arvojen tn.:n riippuen solmun vanhempien arvoasetuksesta
- \* Kunkin rivin tn.:ien on summauduttava arvoon 1
- \* Edellisessä esimerkissä kaikki muuttujat ovat totuusarvoisia, joten riittää tn.  $p$  tapauksessa, että muuttujan arvo on tosi, koska arvon epätosi tn. on  $1 - p$
- \* Yleisesti ottaen, jos solmulla on  $k$  vanhempaa, niin erikseen määrättäviä tn.:ä on  $2^k$  kpl
- \* Jos solmulla ei ole vanhempia on taulukossa vain yksi rivi

- \* Yhteisjakauma  $P(x_1, \dots, x_n)$  voidaan kirjoittaa tulosäännöllä muotoon

$$P(x_n | x_{n-1}, \dots, x_1)P(x_{n-1}, \dots, x_1)$$

- \* Samaa purkamista voidaan toistaa, jolloin lopulta saadaan kaava

$$\begin{aligned} & P(x_n | x_{n-1}, \dots, x_1)P(x_{n-1} | \\ & \quad x_{n-2}, \dots, x_1) \dots P(x_2 | x_1)P(x_1) \\ &= \prod_{i=1}^n P(x_i | x_{i-1}, \dots, x_1) \end{aligned}$$

- \* Tämä ketjusääntö (chain rule) pätee mille tahansa satunnaismuuttujien joukolle

- \* Jakauma on siis yhtäpitävä väittämän

$$\begin{aligned} & \vec{P}(X_i | X_{i-1}, \dots, X_1) \\ &= \vec{P}(X_i | \text{Vanhemmat}(X_i)) \end{aligned}$$

kanssa kunhan  $\text{Vanhemmat}(X_i) \subseteq \{X_{i-1}, \dots, X_1\}$

224

- ★ Kun solmut indeksoidaan järjestyksessä, joka on konsistentti verkkostruktuurin implisiittisen osittaisjärjestyksen kanssa, niin ehto pätee
- ★ Solmuilta siis vaaditaan ehdollista riippumattomuutta edeltäjäsolmuista annettuna vanhemmat
- ★ Solmun vanhemmiksi tulee valita ne solmut, jotka vaikuttavat muuttujan arvoon suoraan
- ★ Jos esimerkiksi ajatellaan Meerin tarkastuskäyntisolmua, niin murto ja järjestys vaikuttavat sen arvoon, mutta vain hälytyksen kautta
- ★ Uskomme mukaan

$$\bar{P}(MK | JK, H, J, M) = \bar{P}(MK | H)$$

225

- ★ Bayes-verkossa voidaan tarkkuutta vaihtaa kompleksisuuteen
- ★ Heikkoja riippuvuuksia voi kannattaa jättää pois verkosta kompleksisuuden pitämiseksi kurissa, mutta tällöin verkon tarkkuus kärsii
- ★ Verkotopologian oikea määrittäminen on vaikea ongelma
- ★ Solmut pitäisi lisätä järjestyksessä: "juuri aiheuttajat", muuttujat, joihin ne vaikuttavat suoraan, jne., kunnes lopulta lehdeksi tulevat muuttujat jotka eivät vaikuta suoraan mihinkään muihin muuttujiin
- ★ Väärässä järjestyksessä lisätyt solmut tekevät verkosta kompleksisemmän ja epäintuitiivisen

227

## Ehdollisten jakaumien esittäminen

- ★ Bayes-verkon etu täyteen yhteisjakamaan verrattuna on kompaktisuus, joka seuraa sen lokaalisuudesta
- ★ Alikomponentit ovat tekemisissä vain rajoitetun määrän muita komponentteja kanssa riippumatta komponenttien kokonaisuudesta
- ★ Tästä seuraa kompleksisuuden lineaarinen, ei eksponentiaalinen, kasvu
- ★ Bayes-verkko sovellukselle, jossa kuhunkin  $n$ :stä muuttujasta vaikuttaa suoraan korkeintaan  $k$  muuttujaa, vaatii ehdollisten tn.taulukoiden esittämiseen  $2^k$  lukua
- ★ Kaikkiaan siis  $n2^k$  lukua, kun yhteisjakauksessa on  $2^n$  solua
- ★ Esim.  $n = 30$  ja  $k = 5$ :  $n2^k = 960$  ja  $2^n > 10^9$

226

- ★ Epävarmuutta voidaan esittää myös kohinaisilla loogisilla suhteilla
- ★ Kohinaisessa disjunktiossa (noisy-OR) voidaan lapsen totuus estää riippumattomasti vanhemman arvon ollessa lapsen arvon todeksi tekevä muuttuja

$$P(\neg\text{kuume} | \text{kylmä}, \neg\text{flunssa}, \neg\text{malaria}) = 0.6$$

$$P(\neg\text{kuume} | \neg\text{kylmä}, \text{flunssa}, \neg\text{malaria}) = 0.2$$

$$P(\neg\text{kuume} | \neg\text{kylmä}, \neg\text{flunssa}, \text{malaria}) = 0.1$$

$Ky$	$Fl$	$Ma$	$P(Kuume)$	$\neg P(Kuume)$
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	0.6
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

229

## Jatkuva-arvoiset muuttujat

- ★ Näiden käsittely voidaan välttää diskreetoinnilla, jossa jatkuva arvoalue jaetaan joihinkin intervaleihin
- ★ Liian harvasta jaosta seuraa suuri tarkkuuden menetys ja liian tiheästä puolestaan suuri määrä käsiteltäviä arvoja
- ★ Diskreointi voi kuitenkin olla jopa todistettavasti oikea lähestymistapa
- ★ Toinen mahdollisuus on arvioida jatkuvaa arvoaluetta jollain standardilla todennäköisyyden tiheysfunktiopelellä, joilla on äärellinen määrä parametreja
- ★ Esim. Gaussin l. normaalijakauman  $N(\mu, \sigma^2)(x)$  parametrit ovat keskiarvo  $\mu$  ja varianssi  $\sigma^2$

230

- ★ Hybridiverkossa on sekä diskreettejä että jatkuva-arvoisia muuttujia
- ★ Tällöin on kyettävä laskemaan jatkuva-arvoisen solmun ehdollinen tn. ja diskreetin solmun tn. annettuna jatkuva-arvoiset vanhemmat
- ★ Esim. hedelmän (jatkuva-arvoinen) *Hinta* riippuu (jatkuva-arvoisesta) *Sadosta* ja (binääriarvoisesta) *Tuesta*
- ★ Asiakkaan (diskreetti) *Ostopäätös* puolestaan riippuu yksinomaan hinnasta
- ★ *Hinnan* tn.jakauman  $\bar{P}(Hinta | Sato, Tuki)$  määrittämiseksi lasketaan erikseen  $P(Hinta | Sato, tuki)$  ja  $P(Hinta | Sato, \neg tuki)$
- ★ Lisäksi tarkastellaan hinnan  $h$  jakauman muuttujan *Sato* arvosta  $s$

231

- ★ Hinnan jakauman parametrit määritellään siis  $s$ :n funktiona
- ★ Usein käytetty jakauma on lineaarinen Gaussin jakauma, jonka keskiarvo  $\mu$  riippuu lineaarisesti vanhemman arvosta riippuen ja jolla on kiinnitetty hajonta  $\sigma$

$$P(h | s, tuki) = N(a_t s + b_t, \sigma_t^2)(h) = \frac{1}{\sigma_t \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{h - (a_t s + b_t)}{\sigma_t} \right)^2}$$

$$P(h | s, \neg tuki) = N(a_f s + b_f, \sigma_f^2)(h)$$

- ★ Näin muodostuvat yksikyttyräiset jakaumat yhdistämällä saadaan lopulta kaksikyttyräinen jakauma  $P(h | s)$
- ★ Lineaarista normaalijakaumaa noudattavan Bayes-verkon yhteisjakauma "käyttäytyy hyvin"

232

★ Diskreetti *Ostopäätös* riippuu tuotteen hinnasta

★ Hinnan ollessa korkea ei ostoja tehdä ja hinnan ollessa matala asiakas ostaa hedelmän

★ Hinnan ollessa näiden väliltä asiakkaan ostopäätös muuttuu pehmeästi/sileästi

★ Ehdollinen todennäköisyys on siis pehmeä kynnsarvo

★ Yksi mahdollisuus on käyttää ns. probit-jakaumaa, standardin normaalijakauman integraalia

$$\Phi(x) = \int_{-\infty}^x N(0,1)(x)dx$$

233

## Tarkka päättely B-verkoissa

★ Tavoitteemme on selvittää kyselymuuttujan  $X$  posteriorijakauma kun havaintomuuttujien  $\vec{E} = E_1, \dots, E_m$  havaitut arvot ovat  $\vec{e}$  ja piilomuuttujat ovat  $\vec{Y} = Y_1, \dots, Y_l$

★ Täydestä yhteisjakaumasta voimme vastata kyselyyn  $\vec{P}(X | \vec{e})$  laskemalla

$$\alpha \vec{P}(X, \vec{e}) = \alpha \sum_{\vec{y}} \vec{P}(X, \vec{e}, \vec{y})$$

★ Bayes-verkko on yhteisjakauman täysi esitys, erityisesti tekijät  $\vec{P}(X, \vec{e}, \vec{y})$  voidaan ilmaista verkon ehdollisten tn.:ien tulona

★ Täten verkon perusteella voidaan vastata annettuun kyselyyn

235

★ Kyselyn  $\vec{P}(\text{Murto} | \text{jussikäy, meeriikäy})$  arvottaminen

★ Piilomuuttujat ovat *Maanjäristys* ja *Hälytys*

$$\vec{P}(M | jk, mk) = \alpha \sum_j \sum_h \vec{P}(M, j, h, jk, mk)$$

★ Verkon ehdollisten tn.taulukoiden perusteella tarvittavat arvot voidaan laskea, esim.

$$P(m | jk, mk) = \alpha \sum_j \sum_h P(m)P(j)$$

$$P(h | m, j)P(jk | h)P(mk | h)$$

★ Tekijöitä järjestelemällä saadaan tehokkaammin laskettava kaava

$$\alpha P(m) \sum_j P(j) \sum_h P(h | m, j) P(jk | h)P(mk | h)$$

234

★ Tällöin ostopäätöksen ehdollinen tn. voisi olisi

$$P(\text{ostaa} | \text{hinta} = h) = \Phi((-h + \mu)/\sigma)$$

★ Esim. neuraaliverkoissa paljon käytetty pehmeän kynnyksen määrittelytapa on ns. logit-jakauma, joka käyttää sigmoidi-funktiota

$$P(\text{ostaa} | h) = \frac{1}{1 + \exp(-2 \frac{-h + \mu}{\sigma})}$$

★ Logit-jakaumalla on pidemmät hännät kuin probit-jakaumalla

★ Probit on käytännössä tarkempi kuin logit, joka taas puolestaan on helpompi käsitellä matemaattisesti ja siksi yleinen

236

★ Laskemalla  $P(m | jk, mk)$  ja  $P(-m | jk, mk)$ , saadaan aiemmin käyttämillemme arvoilla

$$\vec{P}(M | jk, mk) \approx (0.284, 0.716)$$

★ Bayes-verkon DAGin evaluoiminen vastaa puun syvyysuuntaista läpikäyntiä, joten sen tilavaativuus on vain lineaarinen muuttujien lkm:n suhteen

★ Aikavaativuus  $n$ :n muuttujan verkolle on aina  $O(2^n)$

★ Esim. tulo  $P(jk | h)P(mk | h)$  joudutaan laskemaan uudestaan jokaiselle arvolla  $j$

★ Toistuvien alilauseiden uudelleenvaluoinnin välttämiseksi voidaan saavuttaa lisää säästöä

237

★ Toistuvat alilauseet voidaan evaluoida kerran ja niiden arvo talletetaan käytettäväksi aina tarvittaessa

★ Tarkastellaan muuttujien eliminointi-algoritmia lausekkeen  $\vec{P}(M | jk, mk)$  evaluoinnissa

$$\alpha \underbrace{\vec{P}(M)}_M \sum_j \underbrace{P(j)}_j \sum_h \underbrace{\vec{P}(h | M, j)}_H \underbrace{P(jk | h)}_{JK} \underbrace{P(mk | h)}_{MK}$$

★ Lausekkeen tekijät on merkitty

★ Tekijä  $MK$ ,  $P(mk | h)$ , ei vaadi yli muuttujan *MeeriKäy* summaamista, koska arvo  $mk$  on kiinnitetty

★ Talletetaan tn.:t 2-alkioiseen vektoriin

$$\vec{f}_{MK}(H) = \begin{pmatrix} P(mk | h) \\ P(mk | -h) \end{pmatrix}$$

238

★ Tekijälle  $JK$  talletetaan samanlainen 2-alkioinen vektori  $\vec{f}_{JK}(H)$

★ Tekijä  $H$  on  $\vec{P}(h | M, j)$  ja se vaatii  $2 \times 2 \times 2$  matriisiin  $\vec{f}_H(H, M, J)$

★ Näiden kolmen tekijän tulosta summataan  $H$  pois, jolloin saadaan yli muuttujien  $M$  ja  $J$  arvoalueiden käyvä  $2 \times 2$  matriisi  $\vec{f}_{H,JK,MK}(M, J) =$

$$\sum_h \vec{f}_H(h, M, J) \times \vec{f}_{JK}(h) \times \vec{f}_{MK}(h) = \vec{f}_H(h, M, J) \times \vec{f}_{JK}(h) \times \vec{f}_{MK}(h) + \vec{f}_H(-h, M, J) \times \vec{f}_{JK}(-h) \times \vec{f}_{MK}(-h)$$

★ Vastaavasti muuttuja  $J$  summataan pois tulosta  $\vec{f}_J(J) \times \vec{f}_{H,JK,MK}(M, J)$ , jolloin saadaan matriisi  $\vec{f}_{J,H,JK,MK}(M) =$

$$\vec{f}_J(j) \times \vec{f}_{H,JK,MK}(M, j) + \vec{f}_J(-j) \times \vec{f}_{H,JK,MK}(M, -j)$$

239

★ Lopulta saadaan kyselyn  $\vec{P}(M | jk, mk)$  vastaus

$$\alpha \vec{f}_M(M) \times \vec{f}_{J,H,JK,MK}(M),$$

missä  $\vec{f}_M(M) = \vec{P}(M)$

★ Muuttujien summaamisessa pois voidaan tekijät, jotka eivät riipu muuttujan arvosta, siirtää summauksen ulkopuolelle

★ Kahden matriisin pisteittäisen tulon laskeminen

$A$	$B$	$\vec{f}_1(A, B)$	$B$	$C$	$\vec{f}_2(B, C)$
T	T	.3	T	T	.2
T	F	.7	T	F	.8
F	T	.9	F	T	.6
F	F	.1	F	F	.4

240

A	B	C	$\vec{f}_3(A, B, C)$
T	T	T	.3 × .2
T	T	F	.3 × .8
T	F	T	.7 × .6
T	F	F	.7 × .4
F	T	T	.9 × .2
F	T	F	.9 × .8
F	F	T	.1 × .6
F	F	F	.1 × .4

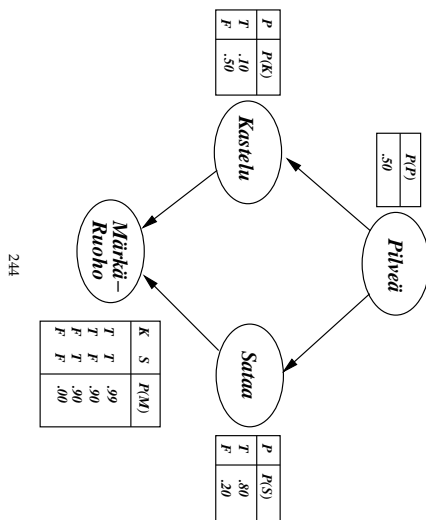
- ★ Kyselyn kannalta irrelevantit tekijät voidaan poistaa
- ★ Kysely  $P(JK | m)$  antaa lauseen, jonka viimeinen tekijä on  $\sum_{mk} P(mk | h)$ , jonka arvo määritelmän mukaan on 1
- ★ Verkon solmut, jotka eivät ole kyselymuuttujan esi-isiä tai havaintomuuttujia ovat kyselyn kannalta irrelevantteja

241

## Approksimatiivinen päättely

- ★ Koska tarkka päättely on laskennallisesti vaativaa, niin on syytä tarkastella ratkaisujen approksimointia
- ★ Approksimointi perustuu satunnaiseen otantaan tunnetusta tn.jakaumasta
- ★ Esim. painottamaton kolikko voidaan mieltää satunnaismuuttujaksi *Lantti*, jonka arvoalue on *(kruuna, klaava)* ja prioritn.  $\vec{P}(\text{Lantti}) = \langle 0.5, 0.5 \rangle$
- ★ Otanta tästä jakaumasta vastaa kolikon heittoa, tn.:llä 0.5 tuloksena on *kruuna* ja tn.:llä 0.5 *klaava*
- ★ Jos satunnaislukugeneraattori, jolta saadaan lukuja väliltä  $[0, 1]$ , niin minkä tahansa yhden muuttujan jakaumasta voidaan helposti tehdä otantaa

243



244

- ★ Bayes-verkko on *monipuuvu* (polytree), jos kunkin solmuparin välillä on korkeintaan yksi suuntaamaton polku
- ★ Tarkan päättelyn aika- ja tilavaativuus monipuussa on verkon koon suhteen lineaarista
- ★ Päättely Bayes-verkossa sisältää erikoistapauksenaan propositiologiikan päättelyn
- ★ Näin ollen yleisessä tapauksessa päätely Bayes-verkossa on NP-kovaa
- ★ Itse asiassa voidaan osoittaa, että päätely on yhtä vaativaa kuin toteuttavien arvoasetusten lukumäärän laskeminen
- ★ Ongelma on aidosti vaikeampi kuin NP-kova, se on #P-kova

242

- ★ Bayes-verkosta, johon ei liity havain-toja, otantaa voidaan tehdä muuttuja kerrallaan topologisessa järjestyksessä
  - ★ Kun vanhempien arvot on arvottu, niin tiedetään minkä jakauman perusteella otanta lapsessa on tehtävä
  - ★ Kiinnitetään esimerkkiverkon solmuille topologinen järjestys [*Pilveä, Kastelu, Sataa, MärkäRuoho*]
1. Vedetään jakaumasta  $\vec{P}(\text{Pilveä}) = \langle 0.5, 0.5 \rangle$  satunnainen arvo, esim. *T*
  2. Vedetään jakaumasta  $\vec{P}(\text{Kastelu} | \text{pilveä}) = \langle 0.1, 0.9 \rangle$  satunnainen arvo, esim. *F*
  3. Vedetään jakaumasta  $\vec{P}(\text{Sataa} | \text{pilveä}) = \langle 0.8, 0.2 \rangle$  satunnainen arvo, esim. *T*
  4. Vedetään jakaumasta  $\vec{P}(\text{MärkäRuoho} | \text{-kastelu, sataa}) = \langle 0.9, 0.1 \rangle$  satunnainen arvo, esim. *T*

245

- ★ Verkon määräämästä prioriyhteisjakaumasta nyt vedetty tapahtuma siis on  $[T, F, T, T]$
- ★ Merk. tn.:ttä, että prioriotanta vetää tietyn tapahtuman  $S_{PO}(x_1, \dots, x_n)$
- ★ Otantamenetelmän perusteella se on

$$\prod_{i=1}^n P(x_i | \text{vanhemmat}(X_i))$$

- ★ Toisaalta tämä on tapahtuman tn. Bayes-verkon esittämässä yhteisjakaumassa, joten

$$S_{PO}(x_1, \dots, x_n) = P(x_1, \dots, x_n)$$

- ★ Merk. tapahtuman  $x_1, \dots, x_n$  frekvenssiä  $N_{PO}(x_1, \dots, x_n)$  yhteensä  $N$ :n otospisteiden joukossa

246

- ★ Tapahtuman otantafrekvenssi konvergoituu rajalla odotusarvoonsa

$$\lim_{N \rightarrow \infty} \frac{N_{PO}(x_1, \dots, x_n)}{N} = \frac{S_{PO}(x_1, \dots, x_n)}{P(x_1, \dots, x_n)}$$

- ★ Esim.  $S_{PO}([T, F, T, T]) = 0.5 \times 0.9 \times 0.8 \times 0.9 = 0.324$ , joten kun  $N$  on iso, niin odotamme, että 32.4% otospisteistä on tämä tapahtuma
  - ★ Menetelmän antama arvio on *konsistentti* siinä mielessä, että tn. on eksakti rajalla
  - ★ Osittain määrätyn tapahtuman  $x_1, \dots, x_m$ ,  $m \leq n$ , tn.:lle saadaan myös konsistentti estimaatti
- $$P(x_1, \dots, x_m) \approx N_{PO}(x_1, \dots, x_m) / N$$
- ★ Otoksesta arvioitua tn.:ttä merk.  $\hat{P}(\cdot)$

247

- ★ Ehdollisen tn.:n  $P(X | \vec{e})$  arvioimiseksi voitaisiin käyttää seur. otantamenetelmää
1. Vedetään otos verkon määräämästä priorijakaumasta
  2. Hylätään kaikki otospisteet, jotka eivät toteuta havaintoja  $\vec{e}$
  3. Arvon  $\hat{P}(X = x | \vec{e})$  määräämiseksi lasketaan kuinka suuressa osassa jäljellejääneistä otospisteistä pätee  $X = x$
- ★ Menetelmän antama jakauma  $\hat{P}(X | \vec{e})$  on algoritmin perusteella

$$\alpha \vec{N}_{PO}(X, \vec{e}) = \frac{\vec{N}_{PO}(X, \vec{e})}{N_{PO}(\vec{e})}$$

- ★ Osittain määrätyn tapahtuman tn.:n arviona tämä on konsistentti estimaatti

$$\hat{P}(X, \vec{e}) \approx \frac{\vec{P}(X, \vec{e})}{P(\vec{e})} = \vec{P}(X | \vec{e})$$

248

- ★ Vedetään 100 otospistettä jakauman  $\vec{P}(Sataa | kastelu)$  estimoisimeksi
  - ◇ Saamistamme tapahtumista 73, joilla pätee  $Kastelu = F$ , hylätään
  - ◇ Lopuilla tapahtumilla pätee  $kastelu$
  - ◇ Näistä 8:ssä tapauksessa  $Sataa = T$  ja 19:ssä  $Sataa = F$
- ★ Tässä tapauksessa  $\vec{P}(Sataa | kastelu) \approx (0.296, 0.704)$ , kun todellinen jakauma on  $(0.3, 0.7)$
- ★ Suurempi otos tuottaa tarkemman estimaatin
- ★ Tn. arvioiden virheen hajonta on suhteessa osamäärään  $1/\sqrt{n}$ , missä  $n$  on otospisteiden lkm
- ★ Turhaan vedettyjen otospisteiden suuri määrä on ongelma: havaintojen kanssa konsistenttien otospisteiden lkm putoaa eksponentiaalisesti ehtomuuttujien lkm:n kasvaessa

249

- ★ Turhaan hylättävien otospisteiden vetämisen välttämiseksi kiinnitetään havaintomuuttujien  $\vec{E}$  arvot ja tehdään otanta vain muuttujien  $X$  ja  $\vec{Y}$  yli
- ★ Vedetyt tapahtumat eivät kuitenkaan kaikki ole samanarvoisia
- ★ Tapahtumia painotetaan uskomusarvoilla (lkelihood)
- ★ Kuinka uskottavasti tapahtuma vastaa havaintoja mitattuna havaintomuuttujien ehdollisten tn.:ien tulolla annettuna niiden vanhemmat
- ★ Intuitiivisesti ajatellen tapahtumille, joissa havaintojen yhdessä esiintyminen vaikuttaa epäuskottavalta, annetaan vähemmän painoarvoa

250

- ★ Kyselyyn  $\vec{P}(Sataa | kastelu, märkäruoho)$  vastaamiseksi painoarvo  $w$  alustetaan arvoon 1.0
- ★ Vedetään otos jakaumasta  $\vec{P}(Pilveä) = (0.5, 0.5)$ , esim. arvo  $T$
- ★ Koska  $Kastelu$  on havaintomuuttuja, jonka arvo on  $T$ , niin painoa päivitetään
 
$$w \leftarrow w \times P(kastelu | pilveä) = 0.1$$
- ★ Vedetään otos jakaumasta  $\vec{P}(Sataa | pilveä) = (0.8, 0.2)$ , esim. arvo  $T$
- ★ MärkäRuoho on havaintomuuttuja, jonka arvo on  $T \Rightarrow$ 

$$w \leftarrow w \times P(märkäruoho | kastelu, sataa) = 0.099$$
- ★ Saatiin siis esimerkki  $[T, T, T, T]$  tapauksesta  $Sataa = T$  painoltaan 0.099

251

- ★ Merk.  $\vec{Z} = \{X\} \cup \vec{Y}$
- ★ Otospisteitä painottava otanta arpoo kullekin muuttujista  $\vec{Z}$  arvon annettuna sen vanhempien arvot
 
$$S_W(\vec{z}, \vec{e}) = \prod_{i=1}^l P(z_i | vanhemmat(Z_i))$$
- ★  $Vanhemmat(Z_i)$  voi sisältää niin havainto- kuin piilomuuttujakin
- ★ Painottava otanta siis ottaa havainnot paremmin huomioon kuin priorijakauma  $P(\vec{z})$
- ★ Toisaalta  $S_W$  ottaa huomioon vain kunkin muuttujan  $Z_i$  esi-isiin lukeutuvat havainnot
- ★ Todellinen posteriorijakauma  $P(\vec{z} | \vec{e})$  huomioi kaikki havainnot

252

- ★ Uskottavuuspainot  $w$  korjaavat jakaumien eron
- ★ Olk. otospiste  $\vec{z}$  muodostunut muuttujien arvoista  $\vec{z}$  ja  $\vec{e}$ , jolloin

$$w(\vec{z}, \vec{e}) = \prod_{i=1}^m P(e_i | vanhemmat(E_i))$$

- ★ Täten otospisteen painotettu tn.  $S_W(\vec{z}, \vec{e})w(\vec{z}, \vec{e})$  on

$$\begin{aligned} & \prod_{i=1}^l P(z_i | vanhemmat(Z_i)) \\ & \prod_{i=1}^m P(e_i | vanhemmat(E_i)) \\ & = P(\vec{z}, \vec{e}), \end{aligned}$$

koska tulojen muuttajat kattavat kaikki verkon muuttajat

253

- ★ Nyt voidaan osoittaa, että painotustannan estimaatit ovat konsistentteja
 
$$\begin{aligned} \hat{P}(x | \vec{e}) &= \alpha \sum_{\vec{y}} N_W(x, \vec{y}, \vec{e}) w(x, \vec{y}, \vec{e}) \\ &\approx \alpha' \sum_{\vec{y}} S_W(x, \vec{y}, \vec{e}) w(x, \vec{y}, \vec{e}) \\ &= \alpha' \sum_{\vec{y}} P(x, \vec{y}, \vec{e}) \\ &= \alpha' P(x, \vec{e}) = P(x | \vec{e}) \end{aligned}$$
- ★ Painotusotanta on tehokas menetelmä, koska kaikki vedetyt otospisteet hyödynnetään
- ★ Menetelmä kuitenkin kärsii kun havaintomuuttujien lkm kasvaa, koska useimpien otospisteiden paino on hyvin pieni ja harvat pisteet dominoivat estimaattia

254

## MCMC-algoritmi

- ★ Markov chain Monte Carlo
- ★ Monte Carlo -algoritmi on satunnaisalgoritmi, joka voi tuottaa väärän vastauksen pienellä tn.:llä (vs. Las Vegas -algoritmi)
- ★ Otospisteitä vedetään tekemällä satunnainen muutos edelliseen tapahtumaan
- ★ Seuraava tila valitaan arpomalla arvo yhdelle ei-havaintomuuttujista  $X_i$  ehdollistettuna sen *Markov-peitteeseen* kuuluvien muuttujien nykyisillä arvoilla
- ★ Solmun Markov-peitteeseen kuuluvat sen vanhemmat, lapset ja lapsien vanhemmat
- ★ MCMC tuottaa satunnaiskulun tila-avaruudessa, jossa havaintomuuttujien arvoja ei muuteta

255

## 5.2 Yksinkertaiset päätökset

- ★ Liitetään tilaan  $S$  numeerinen *hyöty-arvio* (utility)  $U(S)$ , joka kuvaa tilan saavuttamisen haluttavuutta
- ★ Epädeterministisen toiminnon  $A$  mahdollisia tulostiloja ovat  $Tulos_i(A)$ , missä  $i$  käy yli eri tulosten
- ★ Ennen toiminnon  $A$  suorittamista sen mahdollisille tuloksille annetaan tn.:t  $P(Tulos_i(A) | Suorita(A), E)$ , missä  $E$  on agentin havainnot
- ★  $A$ :n odotettu hyöty (expected u.)

$$\begin{aligned} EU(A | E) &= \\ & \sum_i P(Tulos_i(A) | Suorita(A), E) \cdot \\ & U(Tulos_i(A)) \end{aligned}$$

256

\* **Maksimaalisen odotetun hyödyn periaate** edellyttää rationaalisen agentin valitsevan sen toiminnon, jonka odotusarvoinen hyöty on suurin

\* Jos ideaa haluttaisiin soveltaa toimintojen valintaan, niin kaikki mahdolliset jonot tulisi arvottaa, mikä on käytännössä mahdotonta

\* Jos hyötyfunktio heijastaa käytettyä tuloksellisuusmittaa, niin periaatteen mukaan toimiva agentti saavuttaa parhaan mahdollisen tuloksen yli mahdollisten toimintaympäristöjen

\* Mallinnetaan epädeterminististä toimintoa *arvonnalla* (lottery)  $L$ , jossa mahdollisiin tuloksiin  $C_1, \dots, C_n$  liittyvät todennäköisyydet  $p_1, \dots, p_n$

$$L = [p_1, C_1; p_2, C_2; \dots; p_n, C_n]$$

257

$A \succ B$  Agentti preferoi arvontaa  $A$

$A \sim B$  agentille  $A$  ja  $B$  ovat samanarvoisia

$A \succeq B$  agentti preferoi  $A$ :ta tai  $A$  ja  $B$  ovat sille samanarvoisia

\* Deterministinen arvonta  $[1, A] \equiv A$

\* Preferenssirelaatiolle asetetaan rationaalisuuden nimissä seuraavat rajoitteet

\* **Järjestyvyys:** agentin on kyettävä suhtauttamaan mitkä tahansa kaksi tilaa keskenään, valitsemaan niiden väliltä

$$(A \succ B) \vee (B \succ A) \vee (A \sim B)$$

\* **Transitiivisuus:**

$$(A \succ B) \wedge (B \succ C) \Rightarrow (A \succ C)$$

258

\* **Jatkuvuus:**

$$A \succ B \succ C \Rightarrow \exists p : [p, A; 1-p, C] \sim B$$

\* **Korvattavuus:**  $A \sim B \Rightarrow$

$$[p, A; 1-p, C] \sim [p, B; 1-p, C]$$

\* **Monotonisuus:**  $A \succ B \Rightarrow$

$$(p \geq q \Leftrightarrow [p, A; 1-p, B] \succeq [q, A; 1-q, B])$$

\* **Jaettavuus:** Sisäkkäiset arvonnat voidaan todennäköisyyslaskennan sääntöjen mukaan purkaa

$$[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; (1-p)q, B; (1-p)(1-q), C]$$

\* **Huom.:** ei mainintaa hyödyistä

259

1. **Hyötyperiaate:** Jos agentin preferenssit noudattavat edellä olleita aksioomia, niin on olemassa reaaliarvoinen funktio  $U$  s.e.

$$U(A) > U(B) \Leftrightarrow A \succ B$$

$$U(A) = U(B) \Leftrightarrow A \sim B$$

2. **Odotetun hyödyn maksimoimisen periaate:** Arvonnin hyöty on

$$U([p_1, S_1; \dots; p_n, S_n]) = \sum_{i=1}^n p_i U(S_i)$$

Täten epädeterministisen toiminnon hyöty on kuten aiemmin esitimme

260

## Hyötyfunktioita

\* Rahavarat vaikuttaisi suoraviivaiselta hyötymitalta

\* Agentti preferoi monotonisesti rahaa

\* Rahan arvonnoillekin on määrättävä toimintamalli

◊ Olemme voittaneet tietokilpailussa miljoonan

◊ Tarjolla on kolikonheitto, jossa kruuna tietää kaiken rahan häviämistä ja klaava puolestaan kolmen miljoonan voittoa

◊ Onko ainoa rationaalinen valinta odotusarvoltaan puolestoista miljoonan tarjouksen hyväksyminen?

\* Oikeasti kyseessä onkin varallisuuden (ei voiton) maksimointi

261

\* Hyötyarvojen skaala käy parhaasta mahdollisesta palkinnosta  $u_T$  pahimpaan katastrofiin  $u_L$

\* *Normalisoidulla hyödyllä*  $u_L = 0$  ja  $u_T = 1$

\* Ääriarvojen väliin jäävän tilan  $S$  arvottamiseksi agentti voi verrata sitä standardiarvontaan  $[p, u_T; 1-p, u_L]$

\* Todennäköisyyttä  $p$  on säädettävä kunnes kunnes agentin mielestä standardiarvonta ja  $S$  ovat samanarvoisia

\* Jos käytössä on normalisoidut hyödyt, niin lopullinen  $p$  on  $S$ :n hyötyarvo

\* Usein hyötyarvo on monen muuttujan (attribuutin)  $\vec{X} = X_1, \dots, X_n$  arvojen  $\vec{x} = \langle x_1, \dots, x_n \rangle$  määräämä

263

\* Hyödyn aksioomat eivät määrää yksikäsitteistä hyötyfunktiota

\* Voimme esim. tehdä ftiolle  $U(S)$  lineaarisen muunnoksen

$$U'(S) = k_1 + k_2 U(S)$$

( $k_1$  on vakio,  $k_2$  on mv. positiivinen vakio) ilman, että agentin käyttäytyminen muuttuu

\* Deterministisessä maailmassa, jossa ei ole arvontoja, mikä tahansa monotoninen muunnos säilyttää agentin käytöksen

\* Esim.  $\sqrt[3]{U(S)}$

\* Hyötyfunktio on tällöin ordinaalinen — se antaa tiloille järjestyksen, numeerisilla arvoilla ei ole merkitystä

262

\* Tarkastellaan tilannetta, missä muiden arvojen ollessa samat, attribuutin korkeampi arvo tietää myös korkeampaa hyötyfunktion arvoa

\* Jos attribuuttivektoreille  $\vec{x}$  ja  $\vec{y}$  pätee  $x_i \geq y_i \forall i$ , niin  $\vec{x}$  dominoi (aidosti)  $\vec{y}$ :tä

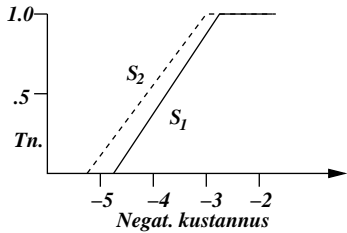
\* Jos esim. lentokentän mahdollinen sijoituspaikka  $S_1$  on halvempi, tuottaa vähemmän äänisaastetta ja on turvallisempi kuin  $S_2$ , niin jälkimmäistä ei enää tarvitse harkita

\* Epävarmuuden vallitessa aidot dominointisuhteet ovat harvinaisempia kuin deterministisessä tapauksessa

\* *Stokastinen dominanssi* on usein käytökelpoinen vertailutapa

264

- ★ Jos lentokentän sijoittamiskustannuksen uskotaan olevan tasaisesti jakautunut välille
  - ◇  $S_1$ : 2.8 ja 4.8 biljoonaa euroa
  - ◇  $S_2$ : 3.0 ja 5.2 biljoonaa euroa



### Informaation arvo

- ★ Öljykenttään myydään porausoikeuksia, palstoja on  $n$  kappaletta, mutta vain yhdessä niistä on  $C$  euron edestä öljyä
- ★ Yhden palstan hinta on  $C/n$  euroa
- ★ Seismologi tarjoaa yritykselle tutkimustietoa palstasta nro 3, joka paljastaa aukottomasti onko palstalla öljyä vai ei
- ★ Paljonko yrityksen kannattaa maksaa tiedosta?
- ★ Tn.:llä  $1/n$  tutkimus kertoo palstalla 3 olevan öljyä, jolloin yritys hankkii sen hintaan  $C/n$  ja ansaitsee  $(n-1)C/n$  euroa
- ★ Tn.:llä  $(n-1)/n$  tutkimus osoittaa, ettei palsta 3 sisällä öljyä, jolloin yritys hankkii jonkin muista palstoista

- ★ Olk.  $E_j$  on satunnaismuuttuja, jonka arvosta saadaan uusi tarkka havainto
- ★ Agentin aiempi tietämys on  $E$
- ★ Ilman lisäinformaatiota parhaan toiminnon  $\alpha$  arvo on  $EU(\alpha | E) =$

$$\max_A \sum_i U(\text{Tulos}_i(A)) \cdot P(\text{Tulos}_i(A) | \text{Suorita}(A), E)$$

- ★ Uusi havainto muuttaa parhaan toiminnon ja sen arvon
- ★ Mutta toistaiseksi  $E_j$  on satunnaismuuttuja, jonka arvoa ei tunneta, joten voimme vain summata yli sen kaikkien mahdollisten arvojen  $e_{jk}$
- ★ Havainnon  $E_j$  arvo on lopulta

$$\left( \sum_k P(E_j = e_{jk} | E) EU(\alpha_{e_{jk}} | E, E_j = e_{jk}) \right) - EU(\alpha | E)$$

- ★ Käytännössä tärkeää on, että informaation arvo on järjestysvapaa

$$VPI_E(E_j, E_k) = VPI_E(E_j) + VPI_{E,E_j}(E_k) = VPI_{E,E_k}(E_j) + VPI_E(E_k)$$

- ★ Tämän perusteella havainnot voidaan erottaa toiminnoista
- ★ Agentin tulisi hankkia lisäinformaatiota kyselyin
  - ◇ järkevässä järjestyksessä,
  - ◇ irrelevantteja kysymyksiä välttämällä,
  - ◇ ottaen huomioon informaation arvon suhteessa sen kustannukseen,
  - ◇ vain silloin kun se on järkevää
- ★ Myopinen agentti tekee toimintapäätöksen heti jos mikään havaintomuuttujista ei näytä riittävän hyödylliseltä — ei tutkita muuttujakombinaatioita

- ★ Kumulatiivinen jakauma on alkupeiräisen jakauman integraali
- ★ Olk. tapahtumien  $A_1$  ja  $A_2$  jakaumat attribuutille  $X$   $p_1(x)$  ja  $p_2(x)$
- ★  $A_1$  dominoi stokastisesti  $A_2$ :ta, jos

$$\forall x \int_{-\infty}^x p_1(x') dx' \leq \int_{-\infty}^x p_2(x') dx'$$

- ★ Jos
  - ◇  $A_1$  dominoi stokastisesti  $A_2$ :ta ja
  - ◇  $U(x)$  on mv. monotonisesti eivähenevä hyötyfunktio,
  - ◇ niin  $A_1$ :n odotusarvoinen hyöty on väh. yhtä korkea kuin  $A_2$ :n

- ★ Jos jokin toiminto on toisen dominoima kaikkien attribuuttien suhteen, niin se voidaan jättää huomiotta

- ★ Koska palstan 3 tilanne jo tunnetaan, niin ostetulta pastalta löytyy nyt öljyä tn.:llä  $1/(n-1)$ , joten yhtiön odotusarvoinen voitto on  $C/(n-1) - C/n = C/n(n-1)$  euroa

$$\text{Odotusarvoinen voitto annettuna tutkimustulos on siis} \\ \frac{1}{n} \frac{(n-1)C}{n} + \frac{n-1}{n} \frac{C}{n(n-1)} = \frac{C}{n}$$

- ★ Seismologille siis kannattaa maksaa aina palstan hintaan asti
- ★ Lisäinformaatio on arvokasta, koska sen avulla toiminta voidaan sopeuttaa vallitsevaan tilanteeseen
- ★ Ilman informaatiota on tyydyttävä kaikissa mahdollisissa tilanteissa keskimäärin parhaaseen toimintaan

- ★ Lisäinformaatiolla on arvoa sikäli kun se voi johtaa suunnitelman muutokseen ja uusi suunnitelma on oleellisesti parempi kuin vanha
- ★ Merk.  $VPI_E(E_j)$  on havainnon  $E_j$  arvo, kun nykyhavainnot ovat  $E$
- ★ Minkä tahansa havainnon arvo on ei-negatiivinen

$$\forall j, E \quad VPI_E(E_j) \geq 0$$

- ★ Arvo riippuu nykyisestä tilasta, joten se voi muuttua tietämyksen myötä
- ★ Äärimmillään informaation arvo putoaa nolnaan, kun tarkastellulle muuttujalle jo tunnetaan arvo
- ★ Siksi informaation arvo ei ole additiivinen

$$VPI_E(E_j, E_k) \neq VPI_E(E_j) + VPI_E(E_k)$$

**INTENTS: INTELLIGENT ONLINE DATA STRUCTURES**

**Prof. Tapio Elomaa**  
elomaa@cs.tut.fi

Suomen Akatemian 2004–07 rahoittama uusi tutkimusprojekti etsii jatko-opiskelijaa tutkijaksi.

Tutkijalta edellytetään erinomaisia matematiikan taitoja. Perustietämys tietorakenne- ja koneoppimisesta luetaan eduksi.

**Keywords:** self-customized and self-adjusting data structures, algorithm analysis, machine learning

### 5.3 Kompleksiset päätökset

- ★ Agentin hyötyarvo riippuukin nyt sarjasta toimintapäätöksiä
- ★ Oheisessa  $4 \times 3$  ruudukkomaailmassa agentti tekee siirtymäpäätöksen  $(Y, O, V, A)$  jokaisella ajanhetkellä
- ★ Kun päädytään toiseen maalitiloista, niin toiminta lakkaa
- ★ Maailma on täysin havainnoitava — agentti tietää sijaintinsa

			+1
			-1
Lähtö			

273

- ★ Jos maailma on deterministinen, niin agentti pääsee lähtötilasta aina lopputilaan +1 siirtymin  $[Y, Y, O, O, O]$
- ★ Koska toiminnot kuitenkin ovat epäluotettavia, niin siirtymäsekvenssi ei aina johda haluttuun tulokseen
- ★ Olk. niin, että aiottu toiminto toteutuu tn.:llä 0.8 ja tn.:llä 0.1 liike suuntautuu-kin kohtisuoriin suuntiin
- ★ Jos agentti törmää maailmansa rajoihin, niin toiminnolla ei ole vaikutusta
- ★ Tällöin sekvenssi  $[Y, Y, O, O, O]$  johtaa-kin maaliin vain tn.:llä  $0.8^5 = 0.32768$
- ★ Lisäksi agentti voi päätyä maaliin satumalta kiertämällä esteen toista kautta tn.:llä  $0.1^4 \times 0.8$ , joten kokonaistn. on 0.32776

274

- ★ *Siirtymämalli* antaa tn.:t toimintojen tuloksille maailman kaikissa mahdollisissa tiloissa
- ★ Merk.  $T(s, a, s')$  on tilan  $s'$  saavuttamistn. kun toiminto  $a$  suoritetaan tilassa  $s$
- ★ Siirtymät ovat *Markovilaisia* sikäli, että vain nykytila  $s$ , ei aiempien tilojen historia, vaikuttaa siihen saavutetaanko  $s'$
- ★ Vielä on määrättävä hyötyfunktio
- ★ Päätösongelma on sekventiaalinen, joten hyötyfunktio riippuu tilajonosta — ympäristöhistoriasta — eikä vain yhdestä tilasta
- ★ Toistaiseksi agentti saa kussakin tilassa  $s$  *palkkion* (reward)  $R(s)$ , joka voi olla positiivinen tai negatiivinen

275

- ★ Esimerkissämme palkkio on  $-0.04$  kaikissa muissa tiloissa paitsi lopputiloissa
- ★ Ympäristöhistorian hyöty puolestaan on palkkioiden summa
- ★ Jos esim. agentti saavuttaa lopputilan +1 kymmenen siirtymän jälkeen, niin hyöty on 0.6
- ★ Pienen negatiivisen palautteen tarkoitus on saada agentti maalitilaan mahdollisimman nopeasti
- ★ Täysin havainnoitavan maailman sekventiaalinen päätösongelma, kun
  - ◇ siirtymämalli on Markovilainen ja
  - ◇ palkkiot summataan,
 on *Markov-päätösongelma* (Markov decision problem, MDP)

276

- ★ MDP on kolmikko
  - ◇ alkutila  $S_0$ ,
  - ◇ siirtymämalli  $T(s, a, s')$  ja
  - ◇ palkkiofunktio  $R(s)$
- ★ Ratkaisuksi MDP:hen ei kelpaa kiinteä siirtymäsekvenssi, koska se voi kuitenkin johtaa muuhun tilaan kuin maaliin
- ★ Vastauksen onkin oltava *politiikka*, joka määrää toiminnan kussakin tilassa, johon agentti voi päätyä
- ★ Poliitiikan  $\pi$  suosittama toiminto tilassa  $s$  on  $\pi(s)$
- ★ Täydellisen politiikan avulla agentti aina tietää mitä tehdä heittipä epäterministinen toimintaympäristö sen mihin tilaan tahansa

277

- ★ Joka kerta kun politiikkaa sovelletaan, maailman stokastisuus johtaa eri ympäristöhistoriaa
- ★ Poliitiikan laadun mittari onkin sen mahdollisesti tuottamien ympäristöhistorioiden hyödyn odotusarvo
- ★ Optimaalisen politiikan  $\pi^*$  odotusarvo on korkein

→	→	→	+1
↑		↑	-1
↑	←	←	←

- ★ Poliitiikka antaa eksplisiittisen kontrollin agentin käyttäytymiselle ja samalla yksinkertaisen refleksiivisen agentin kuvauksen

278

- ★  $-0.0221 < R(s) < 0$

→	→	→	+1
↑		←	-1
↑	←	←	↓

- ★  $-0.4278 < R(s) < -0.0850$

→	→	→	+1
↑		↑	-1
↑	→	↑	←

279

- ★ *Äärettömän horisontin* tapauksessa agentin toiminta-ajalle ei ole asetettu ylärajaa
- ★ Jos toiminta-aika on rajoitettu, niin eri aikoina samassa tilassa voidaan joutua tekemään eri toimintapäätöksiä — optimaalinen politiikka ei ole *stationäärinen*
- ★ Sen sijaan äärettömän horisontin tapauksessa ei ole syytä muuttaa tilan toimintaa kerrasta toiseen, joten optimaalinen politiikka on *stationäärinen*
- ★ Tilajonon  $s_0, s_1, s_2, \dots$  *diskontattu* palkkio on

$$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots,$$

missä  $0 \leq \gamma \leq 1$  on diskonttaustekijä

280

## Arvojen iterointi

- ★ Optimaalisen politiikan selvittämiseksi lasketaan tilojen hyötyarvot ja käytetään niitä optimaalisen toiminnon valitsemiseen
- ★ Tilan hyödyksi lasketaan sitä mahdollisesti seuraavien tilajonojen odotusarvoinen hyöty
- ★ Luonnollisesti jonot riippuvat käytetystä politiikasta  $\pi$
- ★ Olk.  $s_t$  tila, jossa agentti on kun  $\pi$ :tä on noudatettu  $t$  askelta
- ★ Huom.  $s_t$  on satunnaismuuttuja
- ★ Nyt

$$U^\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s \right]$$

- ★ Tilan todellinen hyötyarvo  $U(s)$  on  $U^{\pi^*}(s)$
- ★ Palkkio  $R(s)$  siis kuvaa tilassa  $s$  olemisen lyhyen tähtäyksen hyödyllisyyttä, kun taas  $U(s)$  on  $s$ :n pitkän tähtäyksen hyödyllisyys siitä eteenpäin laskien
- ★ Esimerkkimaailmassamme maalitilan lähellä olevilla tiloilla on korkein hyötyarvo, koska niistä matka on lyhyin

0.812	0.868	0.912	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

$$\pi^* = \arg \max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right],$$

missä odotusarvo lasketaan yli kaikkien mahdollisten tilajonojen, jotka voisivat tulla kyseeseen politiikalla

- ★ Nyt voidaan soveltaa hyödyn odotusarvon maksimointia

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s') U(s')$$

- ★ Koska tilan  $s$  hyöty nyt on diskontattujen palkkioiden summan odotusarvo tästä tilasta eteenpäin, niin se voidaan laskea:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$

- ★ Tämä on *Bellman-yhtälö*
- ★ Toimintaympäristössä, jossa on  $n$  tilaa, on myös  $n$  Bellman-yhtälöä

- ★ Bellman-yhtälöiden yht'aikaisen ratkaisemiseen ei voi käyttää lineaaristen yhtälöryhmien tehokkaita ratkaisumenetelmiä, koska max ei ole lineaarinen operaatio
- ★ Iteratiivisessa ratkaisussa aloitamme tilojen hyötyjen mv. arvoista ja päivitämme niitä kunnes saavutetaan tasapainotila

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s'),$$

missä indeksi  $i$  viittaa iteraation  $i$  hyötyarvioon

- ★ Päivitysten toistuva soveltaminen päättyy taatusti tasapainotilaan, jolloin saavutetut tilojen hyötyarvot ovat ratkaisu Bellman-yhtälöihin
- ★ Löydetyt ratkaisut ovat yksikäsitteisiä ja vastaava politiikka on optimaalinen

## Politiikan iterointi

- ★ Alkaen lähtöpolitiikasta  $\pi_0$  toista
- ★ **Politiikan arviointi:** laske kaikille tiloille hyötyarvo  $U_i = U^{\pi_i}$  politiikkaa  $\pi_i$  sovellettaessa
- ★ **Politiikan parantaminen:** perustuen arvoihin  $U_i$  laske uusi hyödyn odotusarvon maksimoiva politiikka  $\pi_{i+1}$  (vrt. 285)
- ★ Kun jälkimmäinen askel ei enää muuta hyötyarvoja, niin algoritmi päättyy
- ★ Tällöin  $U_i$  on Bellman-päivityksen kiintopiste ja ratkaisu Bellman-yhtälöihin, joten vastaavan politiikan  $\pi_i$  on oltava optimaalinen
- ★ Äärellisellä tila-avaruudella on vain äärellinen määrä politiikkoja, jokainen iteraatio parantaa politiikkaa, joten politiikan iterointi päättyy lopulta

- ★ Koska kullakin kierroksella politiikka on kiinnitetty, niin politiikan arvioinnissa ei ole tarvetta maksimoida yli toimintojen
- ★ Bellman-yhtälö yksinkertaistuu:

$$U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')$$

- ★ Koska epälineaarisesta maksimoinnista on päästy eroon, on tämä lineaarinen yhtälö
- ★ Lineaarinen yhtälöryhmä, jossa on  $n$  yhtälöä ja niissä  $n$  tuntematonta muuttujaa voidaan ratkaista ajassa  $O(n^3)$  lineaarialgebran menetelmin
- ★ Kuutiollisen ajan sijaan voidaan tyytyä approksimoimaan politiikan laatua ajamalla vain tietty määrä yksinkert. arvojen iterointi askelia kelvollisten hyötyarvojen saamiseksi