Nimitys "kvanttori" tulee *lukumääristä (lat. quantus, "kuinka paljon?")*. Sanat "jollakin" ja "kaikilla" toimivat näet luonnollisessa kielessä samoin kuin "ainakin kolme", "tasan viisi",. . .

Luvussa 1.2 todettiinkin, että predikaattilogiikka kehitettiin matematiikan filosofisten perusteiden tutkimiseen. Siinä lukumääriä koskevien lauseiden formalisointi oli tietenkin keskeistä.

Ensimmäisen kertaluvun predikaattilogiikalla ei kuitenkaan pysty formalisoimaan sitä *miten* nämä "kvanttorin kaltaiset" sanat toimivat luonnollisessa kielessä: pelkkä '∀' ei vielä ole tämän kielen kokonainen ilmaisu.

Korkeamman kertaluvun predikaattilogiikassa tällaisten sanojen toiminta voidaan formalisoida funktioina *väitelauseilta väitelauseille* [Car97, luku 3]:

Operaatio '∀' ottaa argumenttinaan *yhtä* oliota koskevan väitelauseen ja tuottaa tuloksenaan saman lauseen mutta yleistettynä *jokaiseen* olioon.

(Ensimmäisessä kertaluvussahan kaikki funktiot ovat *olioilta olioille*.)

Jos tähän suuntaan lähdetään, niin samalla on luontevaa luopua tarkastelemasta vain *predikaatti*logiikkaa, jossa lopulliset tulokset ovat aina totuusarvotyyppiä $\mathbb{B}$. Esimerkiksi $\forall : (\mathcal{U} \mapsto \mathbb{B}) \mapsto \mathbb{B}$.

Näin päästään *lambda-laskentaan (lambda calculus)*, jolla on useita sovelluksia tietojenkäsittelyssä:

- ohjelmointikielissä — teoriassa ja käytännössä
- luonnollisen kielen käsittelyssä [RN95, osa VII][Nil98, osa V]
- . . .

Kurssi ei kuitenkaan lähde tähän suuntaan, koska päättely korkeamman kertaluvun logiikoissa on vielä vaikeampaa kuin ensimmäisen [Vää87, luku VI.3]. Niillä esitettyä tietämystä on siis vielä vaikeampaa käyttää.

5.2.3. Käyttöesimerkkejä

"... [H]e construed the world as a computer construes it, by means of key features and schematic relationships. The scheme might be identified – in an 'identiti-kit' way – without the reality being grasped at all."

O. Sacks: The Man Who Mistook His Wife for a Hat

Esitetään predikaattilogiikalla ensiksi *ihmisten välisiä sukulaisuussuhteita*:

1. Tietämyksen esittäminen logiikalla alkaa siitä, että *sovitaan ne alkiot*, jotka sisällytetään perusjoukkoomme $\mathcal{U}$.
   Eli millaisia otuksia ajattelemme olevan olemassa.
   "Mistä lauseidemme on tarkoitus puhua?"
   Tässä tilanteessa joukkoon $\mathcal{U}$ on luontevaa sisällyttää ihmisiä.
2. Sitten *sovitaan looginen aakkosto (alphabet)*: ne sanat joiden välityksellä tätä tietämystä halutaan kuvailla.
   "Millaisin sanoin haluamme ilmaista lauseemme?"
   Yksi tapa löytää aakkosto on alkaa luetella erilaisia sovellusalueeseen (nyt siis sukulaisuuteen) liittyviä käsitteitä ja lauseita:
   - Sukulaisuussuhde "x on y:n isä/äiti" lienee ainakin tarpeen.
   - Erottelu "x on mies/nainen" lienee myös tarpeen jos haluamme puhua isistä ja äideistä. . .
   - Muita ovat esimerkiksi "x on y:n veli/sisar/sisarus/serkku/esi-isä/jälkeläinen/. . . "

   On vaikea sanoa milloin aakkostoa on kerätty tarpeeksi.
   Aakkostoa ja perusjoukkoa $\mathcal{U}$ voidaan joutua täydentämään myöhemmin, kun tavataan tietämystä jota ei pystytäkään esittämään.
   Vertaa olio-ohjelmoinnin käsiteanalyysiavaihe: "hauki on sellainen kala, jonka. . . " Siis meillä on luokka *kalat*, ja sen aliluokkana *hauet*,. . .
   Vertaa myös tietokantamallin hahmotteluvaihe.
3. Seuraavaksi *jaotellaan kerätty aakkosto* vakioihin, funktioihin ja predikaatteihin:

- Jos isyys/äitiys tulkitaan puhtaan biologisesti, niin voidaan käyttää 1-paikkaisia funktioita $\mathrm{father}(y)$ ja $\mathrm{mother}(y)$, koska jokaisella ihmisellä y on tasan yksi isä/äiti. Sovitaan esimerkissämme näin.

  Voitaisiin käyttää myös 2-paikkaisia predikaatteja $\mathrm{father}(x, y)$ ja $\mathrm{mother}(x, y)$, jolloin samalla ihmisellä y voisi olla monta isää/äitiä x.

  Jaottelu ei siis synny automaattisesti, vaan rajatapaukset vaativat aina harkintaa.
- Sukupuoli voidaan ilmaista 1-paikkaisilla predikaateilla $\mathrm{male}(x)$ ja $\mathrm{female}(x)$.

  Niiden sijasta voitaisiin valita myös funktiosymboli $\mathrm{gender}(x)$. Tällöin jouduttaisiin lisäämään myös
  - aakkostoon symboliset arvot male ja female
  - perusjoukkoon näiden symbolien tulkinnat, eli käsitteet "miehuus" ja "naiseus".

  Valinta näiden kahden vaihtoehdon välillä vaikuttaa myöhemmin kirjoitettavien tietämyksenesityslauseiden muotoiluun, eikä "oikeaa" valintaa ole helppo tehdä tarkastelematta, kummassa nämä lauseet muotoutuvat helpommin.

  Tähän palataan hieman myöhemmin.

  Jo tästä näkee, ettei tämä(kään) tietojenkäsittelyn suunnitteluprosessi etene koko ajan vain yhteen suuntaan. . .
- Muut sukulaisuussuhteet lienee luontevinta esittää 2-paikkaisilla predikaateilla $\mathrm{brother}(x, y)$, $\mathrm{sister}(x, y)$, $\mathrm{sibling}(x, y)$, $\mathrm{cousin}(x, y)$, $\mathrm{ancestor}(x, y)$, $\mathrm{descendant}(x, y)$,. . . koska samalla ihmisellä x voi aina periaatteessa olla useita veljiä/sisaria/sisaruksia/serkkuja/esi-isiä/jälkeläisiä/. . . y.

4. Luokitellaan aakkoston predikaatit *perus-* ja *johdettuihin käsitteisiin* kysymällä kustakin "voinko päätellä sen havainnoista joistakin muista aakkoston esittämistä käsitteistä?"

   Sama predikaatti voi olla sekä perus- että johdettu käsite: esimerkiksi kuvassa 59 (b) 0-paikkainen predikaatti $P_{[3,1]}$ pääteltiin, mutta se olisi voitu havaita suoraankin — astumalla kuoppaan.

   Esimerkiksi "x on y:n sisarus" on johdettu käsite, koska se voidaan määritellä isyyden ja äitiyden avulla muodossa "x ja y omaavat saman isän ja/tai äidin". (Jälleen täytyy samalla sopia hyväksytäänkö sisaruspuoletkin vaiko ei. Sovitaan hyväksytyiksi.)

   Logiikassa vastaava *määritelmä (definition)* on määriteltävän predikaatin ekvivalenssi määrittelevän lauseen kanssa:

$$\forall x, y.\, \mathrm{sibling}(x, y) \Leftrightarrow \mathrm{father}(x) = \mathrm{father}(y) \vee$$
$$\mathrm{mother}(x) = \mathrm{mother}(y).$$

   Vertaa sanakirjamääritelmät:

   sisarukset, n. *Henkilöt, joilla on yhteinen vanhempi.*

   Tässä on jopa määritelmä *tapauksittain (by cases)*: . . . *sama isä <u>tai</u> sama äiti.*

   Esimerkissämme isyys ja äitiys ovat luontevia peruskäsitteitä: niiden pohjalta voidaan määritellä muut sukulaisuussuhteet muttei päinvastoin.

   Sukupuoli on myös otettava peruskäsitteeksi: vaikka tiedämmekin että isät/äidit ovat aina miehiä/naisia eli

$$\forall x.\, \mathrm{male}(\mathrm{father}(x)) \tag{33}$$
$$\forall x.\, \mathrm{female}(\mathrm{mother}(x)) \tag{34}$$

   niin tämä ei vielä kiinnitä lapsettomien ihmisten sukupuolia.

   Luvussa 1.3 puhuttiin loogisten positivistien käännettävyysteesistä. Ero perus- ja määriteltyjen käsitteiden välillä voidaan myös tulkita eroksi
   - suoraan maailmaa tarkastelemalla havaittavissa olevien asioiden
   - (havainnoissa ilmenneille säännönmukaisuuksille sovittujen) kielellisten lyhennysmerkintöjen

   välillä.

   Esimerkissämme voisimme aloittaa havaitsemalla, että ihmiset(kin) ovat sitä läheisempiä toisilleen mitä enemmän heillä on yhteistä geneettistä perimäainesta – siis vanhempi/lapsi-suhde on perustava.

   Tämän päälle kieliyhteisössä on vakiintunut erilaisia muita termejä erottelemaan sitä, kuinka läheisiä ihmiset ovat toisilleen:

"Minä veljeäni vastaan. Minä ja veli serkkuani vastaan. Minä, veli ja serkku muukalaista vastaan." *Arabialainen sananlasku*

Määritelmät voivat myös *viitata itseensä*: "x on y:n jälkeläinen jos ja vain jos x on y:n lapsi tai jonkin y:n *lapsen jälkeläinen*".

$$\forall x, y. \mathrm{child}(x, y) \Leftrightarrow \mathrm{father}(x) = y \vee \mathrm{mother}(x) = y.$$

$$\forall x, y. \mathrm{descendant}(x, y) \Leftrightarrow \mathrm{child}(x, y) \vee \exists z. \mathrm{child}(z, y) \wedge \mathrm{descendant}(x, z).$$

Sukupuissa tällainen itseensäviittaava sääntö esittää navigointia pitkin niitä isä/äiti-linkkejä, joista puu koostuu.

Logiikkaohjelmointikielessä Prolog

- ohjelmat ovat predikaattien määritelmiä tapauksittain
- rekursio (ja siis myös toisto) ilmaistaan itseensäviittaavina määritelminä.

5. Kun käsitteet on valittu, täytyy vielä lausua loogisesti niiden keskinäiset riippuvuudet kuten kaavat (33) ja (34) jotta vältyttäisiin "itsestäänselvyyksien" aiheuttamilta yllätyksiltä:
   - "Jokainen perusjoukon olio on joko mies tai nainen."

$$\forall x. \mathrm{male}(x) \Leftrightarrow \neg \mathrm{female}(x) \tag{35}$$

Jos perusjoukkoomme lisättiin myös sukupuolettomia olioita kuten tiiliskiviä, niin tätä lyhyttä kaavaa pitää monimutkaistaa esimerkiksi muotoon "jokainen *ihminen* on joko mies tai nainen":

$$\forall x.\ \mathrm{human}(x) \Rightarrow (\mathrm{male}(x) \Leftrightarrow \neg \mathrm{female}(x)).$$

Jos taas valittiin funktiosymboli gender, niin mikään alkio ei tämän valinnan nojalla voi olla yhtä aikaa sekä mies että nainen. Mutta:

– Perusjoukkoon lisättiin samalla uudet alkiot "$\mathrm{miehuus}$" ja "$\mathrm{naiseus}$".
  Nämä uudet alkiot eivät ole ihmisiä, vaan ihmisiin liittyviä *abstrakteja käsitteitä*, jotka on nyt *reifioitu* eli tuotu konkreettisiksi alkioiksi.

– Funktiosymbolia gender vastaavalla funktiolla pitää olla arvo *jokaisella* alkiolla, siis myös näillä uusilla alkioilla.
  Tarvitaan siis uusi arvo "$\mathrm{määrittelemätön}$" ja sille vakiosymboli $\mathrm{undefined}$.
– Tarvitaan siis vielä esimerkiksi lause "sukupuoli on määritelty vain ihmisille"

$$\forall x. \neg \mathrm{human}(x) \Leftrightarrow \mathrm{gender}(x) = \mathrm{undefined}$$

rajaamaan määrittelemättömän arvon käyttöä.
– Joudutaan siis lisäämään myös predikaatti "(olla) ihminen".
– Funktiosymbolin gender valinta johtaisi siis ainakin yhtä monimutkaisiin tietämyksenesityslauseisiin kuin predikaattisymbolien male ja female käyttö.
  Se ei siis näytä kannattavan.

Jo tästä pikku pulmasta nähdään yksi (loogisen/symbolisen) tietämyksenesittämisen vaikeus:

Vaikka halutaan esittää vain jokin näennäisen yksinkertainen tiedonpalanen, voi siitä seurata monenlaisia

– siihen liittyvän *tausta*tietämyksen esitystarpeita
– vaikutuksia muiden tiedonpalasten esitykseen.

Silloin tietämyskannan koko paisuu ja sen sisäiset riippuvuudet mutkistuvat.

Tätä vaikeutta on yritetty lieventää eri tavoin:

– On määritelty *yleisiä ontologioita (general ontologies)* [RN95, luku 8.4]: rakenteisia (hierarkisia) "käsitekirjastoja", joihin uusia tiedonpalasia sovitellaan itse perusrakennetta muuttamatta.
  (Vertaa valmiit oliokirjastot.)
– On ryhdytty keräämään ja esittämään suuria määriä arkipäivän taustatietämystä (CYC).
– On kehitetty sellaisia logiikoita, joiden päättelyaskeleet vähentävät tarvetta esittää kaikkea tietoa eksplisiittisesti.
  Esimerkiksi luvussa 5 mainitut epämonotoniset logiikat mahdollistavat sen, että vain tavallisuudesta poikkeava tietämys kirjataan näkyviin. Jos jotakin tietoa ei ole kirjattu, niin päättelyssä

sen voidaan silloin olettaa olevan kuten tavallisesti.
  - On keskitytty koko todellisuuden sijasta sen yksinkertaisempiin osiin ja pelkistetty niitä edelleen.
- "Mikään olio ei ole oma jälkeläisensä."

$$\forall x. \neg \operatorname{descendant}(x, x)$$

Mehän emme missään vaiheessa kieltäneet kehiä sukupuissa. . .

Jälleen on vaikeaa tietää, milloin käsitteiden riippuvuuksia on kuvailtu – eli milloin sovellusaluetta on *aksiomatisoitu* – tarpeeksi.

Vasta kun aksiomatisointia käytetään päättelyyn, selviää salliiko se kaikkiin mielenkiintoisiin kysymyksiin vastaamisen.

Luvun 5 "päättelyn varovaisuusperiaatteen" mukaanhan vain se saadaan päätellä luvallisesti mikä ei missään tapauksessa voisi olla toisin silloin kun päättelyn lähtöoletukset ovat voimassa. Jos siis aksiomatisointi jättää jotakin avoimeksi, niin päättely ei etene.

Esimerkiksi aksioomaa

$$\forall x. \neg \operatorname{father}(x) = \operatorname{mother}(x)$$

eli "isä ja äiti ovat eri oliot" ei enää tarvitse lisätä:

Se ei ole *riippumaton (independent)* muista aksioomista, vaan se voitaisiin myös päätellä aksioomista (33)–(35): isä on mies, äiti taas nainen, eikä kukaan ole yhtä aikaa sekä mies että nainen.

6. Kun sovellusalue näyttää riittävästi aksiomatisoidulta, voidaan tietämyskantaan KB syöttää yksittäisiä havaintoja.

   Esimerkissämme nämä ovat esimerkiksi muotoa

$$\operatorname{mother}(\text{'King Edward VII'}) = \text{'Queen Victoria'}\,.$$

   Havainnot tuovat näin mukanaan uusia vakioita nimeämään havaittuja olentoja.

Näin saadaan kuvattua staattinen, muuttumaton maailma.

Vaikka luvussa 5.2 sanottiinkin, ettei predikaattilogiikka tarjoa *valmiina* menetelmiä dynaamisen, muuttuvan maailman kuvaamiseen, ei mikään estä aksiomatisoimasta sellaista *käsin*.

Eräs luonteva tapa on *tilannekalkyyli (situation calculus)*:

- Jokaiseen ajasta riippuvaan perusilmaukseen (predikaatti- tai funktiosymboliin) lisätään uusi (viimeinen) argumentti.
- Tämä uusi argumentti edustaa sitä *tilannetta (situation)* jossa ilmauksen merkitystä tutkitaan.

  Predikaatin totuus
  funktion osoittama alkio
  riippuu nyt (myös) tilanteesta.

Esimerkki ajasta riippuvasta vakiosta (0-paikkaisesta funktiosta) on "$\text{Suomen presidentti}$".

Määritelläänkin se siis 1-paikkaisena funktiona "$\text{Suomen presidentti ajanhetkellä } t$", jolloin

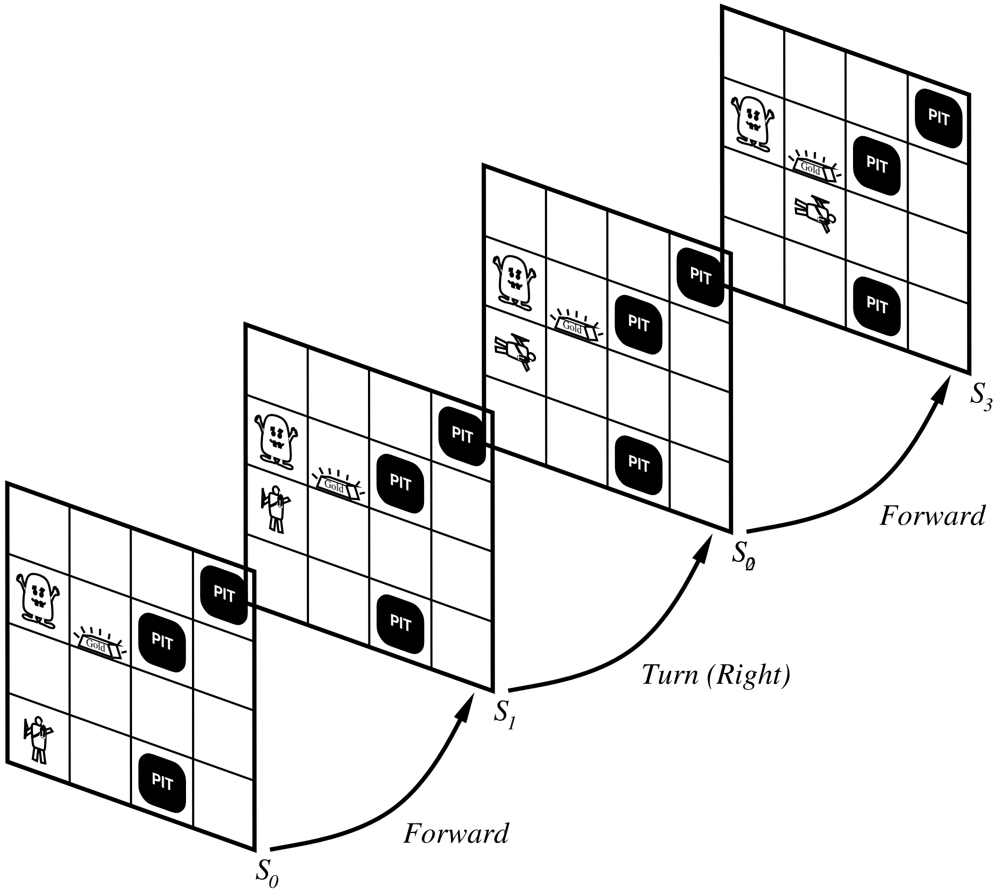
$$\text{president-of-Finland}(\text{'13th Nov. 1998'}) = \text{'M. Ahtisaari'}$$
$$\text{president-of-Finland}(\text{'13th Nov. 2001'}) = \text{'T. Halonen'}$$

kuvaa viranhaltijan vaihtumista joskus tilanteiden '13th Nov. 1998' ja '13th Nov. 2001' välillä.

Ajalle on tarjolla monenlaisia eri malleja, jotka sopivat erilaisiin tiedonesitystilanteisiin:

- Teollisen prosessin ohjausjärjestelmässä aika lienee jatkuva käsite, koska selluloosan keitinliemessä tapahtuu koko ajan jotakin.
- Tilannekalkyylissä taas *toimija(i)n teot* kuljettavat aikaa valitsemalla seuraavan tilanteen monista mahdollisista vaihtoehdoista.
  Esimerkkinä luvun 5.1 Wumpus-esimerkin mukainen kuva 60.

Tilannekalkyylissä on siis valittu "sarjakuva-aika", jossa "ruudusta" eli tilanteesta toiseen edetään "sankarin" eli toimijan tekojen tahdissa.



Kuva 60: Wumpus-pelitilanteita [RN95, kuva 7.3].

Tilannekalkyylissä käytetään erityistä *tulosfunktiota*, jonka edellinen argumentti on toimijan teko ja jälkimmäinen se tilanne missä se tehdään, ja joka palauttaa arvonaan tekoa seuraavan tilanteen. Siis kuvassa 60

$$\mathrm{result}(\mathrm{forward}, S_0) = S_1$$
$$\mathrm{result}(\mathrm{turn}(\mathrm{right}), S_1) = S_2$$
$$\mathrm{result}(\mathrm{forward}, S_2) = S_3 \text{ ja niin edelleen.}$$

Näin tilannekalkyyli reifioi abstraktin käsitteen 'tilanne' jonka avulla muutokset esitetään.

Tekojen aikaansaamat muutokset kuvaillaan *vaikutusaksioomilla (effect axioms)* kuten

"jos tavara x on läsnä tilanteessa s ja kannettavissa, niin teko 'tartu' aiheuttaa sen, että toimija pitää sitä kädessään":

$$\forall x, s.\, \mathrm{present}(x, s) \land \mathrm{portable}(x) \Rightarrow \mathrm{holding}(x, \mathrm{result}(\mathrm{grab}, s))$$

Siinä tarvittavat apukäsitteet ovat:

$$\mathrm{portable}(\mathrm{gold})$$
$$\forall s.\, \mathrm{at\_gold}(s) \Rightarrow \mathrm{present}(\mathrm{gold}, s) \quad (36)$$
$$\forall s, b, u, c, t.\, \mathrm{percept}([s, b, \mathrm{glitter}, u, c], t) \Rightarrow \mathrm{at\_gold}(t) \quad (37)$$

Tämä tarpeettoman monimutkaisen näköinen muotoilu on valittu siten, että *koko* tietämyksen kuvaileminen olisi helpompaa:

- Sama aksiooma kuvaa kaikkien tarttumistekojen toimintaa, tartuttiinpa sitten kultaan tai mihin muuhun tahansa.
- Kaavan (36) "välikäsite" $\mathrm{present}(\text{esine}, \text{tilanne})$ otettiin käyttöön siltä varalta, että toimijalla olisi muitakin tapoja havaita kulta kuin kaavan (37) kimalluksen näkeminen.
- Kaava (37) tutkii puolestaan kullan kimallusta toimijan tilanteessa t saamasta havainnosta havaintovektorista.
  Sen kirjamme [RN95] esittää Prolog-tyylisenä listana [hakasuluissa].
  Samalla nähdään luvun 2.2 heijastetoimijoiden tietämyskannan KB sisältämien sääntöjen yleinen muoto:
  Implikaatiot eli "jos. . . niin. . . " -säännöt

  $$\forall t.\, \mathrm{at\_gold}(t) \Rightarrow \mathrm{action}(\mathrm{grab}, t)$$

  jotka ehdottavat aina seuraavaa tekoa $\mathrm{action}(\text{teko}, \text{tilanne})$ suoraan havainnon pohjalta miettimättä asiaa sen tarkemmin.