TIETOJENKÄSITTELYTEORIAN KÄSITERAKENTEET JA LUONNOLLISEN KIELEN YMMÄRTÄMISEN MALLITTAMINEN[1]

EERO HYVÖNEN
Helsingin Teknillinen korkeakoulu
Digitaalitekniikan laboratorio

1. Johdanto

Perinteisesti on tietokoneita käytetty lähes yksinomaan teknismatemaattiseen laskentaan ja kaupallishallinnolliseen tietojenkäsittelyyn. Niinpä useimmat nykyiset ohjelmointikielet (FORTRAN, ALGOL, COBOL, PASCAL jne.) ja muut suunnittelijan avuksi kehitetyt tietojenkäsittelyteorian käsiterakenteet tukevat yleensä ensisijaisesti matemaattisen ja data-tyyppisen koneläheisen tiedon käsittelyä. Ihmisen symbolimaailman käsittelyyn ne ovat monasti sopimattomia. Suurena esteenä kehittyneiden luonnollisen kielen tietokonemallien toteuttamiselle on puute sellaisista käsitteellisistä välineistä, joilla kielen eri ilmiöitä voitaisiin tietokoneella luontevasti mallittaa. Erinomaista olisi, mikäli kognitiotieteiden parissa työskentelevät tutkijat voisivat yhteistyössä kehittää sellaisia formaaleja lähestymistapoja, joista olisi aiempaa enemmän hyötyä eri paradigmojen edustajille.

Käsityksen esimerkiksi ohjelmointikielten käsiterakenteiden erilaisuudesta saa vertaamalla FORTRANia ja symbolien käsittelyyn tarkoitettua LISP-ohjelmointikieltä toisiinsa: LISP on tekoälytutkimuksen tärkein ohjelmointikieli (Winston ja Horn 1981:13). Sen listarakenteilla voidaan kätevästi esittää vaikkapa seuraava Virtasen perhettä kuvaava symbolinen lauseke:

```
(PERHE-1  ( SUKUNIMI VIRTANEN)
          ( PUHELIN 413557)
          ( MIES MATTI)
          ( VAIMO MAIJA)
          ( LAPSET ((PIKKU-KALLE)(LYYTI))))
```

[1] Lausun lämpimät kiitokseni Ph.D Harri Jäppiselle ja DI Jouko Seppäselle, jotka ovat tehneet arvokkaita huomautuksia tämän käsikirjoituksen aiempiin versioihin.

Vastaavan tiedon esittäminen numeeriseen ohjelmointiin tarkoitetun FORTRANin muuttujilla, taulukoilla, sijoituskäskyillä tms. olisi varsin kömpelöä ja teknistä, vaikka esitettävä asia on ihmiselle yksinkertainen.

Luonteva tiedon ja ohjelmien esitysformalismi saattaa helpottaa käytännön ohjelmointityötä huomattavasti. Formalismin valinta on kuitenkin yhtälailla teoreettinen kuin käytännöllinenkin kysymys: ongelman luonteva esitystapa ja jäsentely on edellytys hyvälle teoreettiselle mallille. Tutkittaessa ymmärryksemme perimmäisiä mekanismeja hämärtyy teorian ja käytännön raja. On olemassa vain teoriakäytäntöä.

Tämän kirjoituksen tarkoituksena on kiinnittää huomiota siihen, miten tärkeä vaihe luonnollisen kielen mallintamisessa on sopivien, ongelmaläheisten käsiterakenteiden suunnittelu ja valinta.

Termiä "tietojenkäsittelyteorian käsiterakenteet" käytetään seuraavissa luvuissa erittäin laajassa, jopa epätäsmällisessä merkityksessä: mm. modulaarinen ohjelmointitekniikka on ymmärretty eräänlaisena käsiterakenteena. Parempaa, käsiteltävien asioiden yhteenkuuluvuutta korostavaa termiä en kuitenkaan keksinyt.

Koska "käsiterakenteet" tässä kirjoituksessa on ymmärretty näin laajasti, jäävät tarkastelut pakosta varsin yleisiksi ja puutteellisiksi. Toivottavasti ajatukset kuitenkin lisäävät kognitiotieteiden edustajien kiinnostusta tietojenkäsittelytieteeseen. Kirjallisuusviitteitä on pyritty antamaan melko runsaasti erityisesti tekoälyn tutkimustulosten tekemiseksi tunnetummiksi Suomessa.

2. Käsiterakenteiden valinnan kriteerejä

2.1. Intuitiivisuus

Järjestelmien intuitiivisuus ja käytön helppous ovat tulleet yhä keskeisemmiksi tietosysteemejä suunniteltaessa. Myös ohjelmointityössä pyritään ennen kaikkea selkeisiin ja yksinkertaisiin (ja siten luotettaviin sekä helposti yllä pidettäviin) ratkaisuihin usein tehokkuuden kustannuksella. Tekoälyn NLU-tutkimuksen (Natural Language Understanding; Barr ja Feigenbaum 1981 ja Bolc 1980) tärkeä tavoite on helppokäyttöisempien, ihmisen lailla toimivien järjestelmien toteuttaminen. Intuitiivisuuden vaatimuksen tulee johdonmukaisesti toteutua myös systeemien toteutuksen tasolla.

Suunnittelun ja ohjelmoinnin ongelmien voimakas korostuminen laitteistoteknologian kehittyessä 1960- ja 70-luvulla on johtanut siihen, että yläarajan kielen ymmärtämisen mallien täydellisyydelle asettaa viime kädessä oh-

jelmistojen kompleksisuus. Lopulta mallin rakenteiden moninaisuus kuitenkin ylittää suunnittelijan suunnittelutaidon ja ohjelmoijan ohjelmointitaidon. Näitä kahta kriittistä rajakohtaa voidaan siirtää eteenpäin kiinnittämällä huomiota formaalien rakenteiden luontevuuteen ihmisen eikä esimerkiksi tietokoneen kannalta. Pyrkimyksenä on kehittää sellaisia korkean tason käsiterakenteita, jotka ovat havainnollisia ja sisältävät vain sovelluskohteeseen liittyviä käsitteitä.

Yleensä hyvät intuitiiviset formalismit perustuvat johonkin analogiaan kuvattavan reaalimaailman seikan kanssa. Tällöin kuvaus reaalimaailmasta formaaleille rakenteille on luonteva. Esimerkiksi luonnollisen kielen lause on yksinkertaista esittää listana, kun taas shakkilaudan formaaliksi esitykseksi sopii taulukko. Varsin tavallisia ovat graafiset analogiat, sillä ihminen pystyy monasti ajattelemaan tehokkaammin kaksiulotteisten kuvien avulla kuin yksiulotteisilla merkkijonoilla. Lisäksi monet reaalimaailman asiat ovat jo perusluonteeltaankin moniulotteisia.

Oikea esitysformalismi jäsentää ongelman luontevasti ja tukee samalla tarvittavia operaatioita, jolloin monimutkaisempien systeemien suunnittelu ja ohjelmointityö käy mahdolliseksi. Osa ongelman ratkaisua voidaan sisällyttää käytettäviin rakenteisiin, jolloin ohjelmat yksinkertaistuvat.

2.2. Käytännön ohjelmointi

Käytännön ohjelmointitekniset näkökohdat eivät ole toisarvoisen tärkeitä kielen teoreettisten mallien kannalta, vaan niiden pitäisi paljolti yhtyä teoreettisiin tarkasteluihin. Ohjelmointitekniikan valinnallahan (esim. modulaarisuus ja ohjelman muu strukturointi) pyritään mahdollisimman mielekkääseen kuvattavan asian jäsentelyyn, mikä on tavoitteena myös teoreettisella tasolla.

Ongelman formaali ratkaisu ja sellaiset yksittäiseen toteutukseen liittyvät seikat kuin käytettävä tietokone tai ohjelmointikieli on kuitenkin osattava pitää erillään. Helposti aletaan ratkaista ongelmia ajatellen tiedostamattomasti esimerkiksi ohjelmointikielen käsittein, jotka sellaisenaan eivät välttämättä heijasta ongelman teoreettista luonnetta. Näin saatetaan kyllä päätyä toimivaan ja tehokkaaseen ohjelmaan, mutta formaalilla tasolla tällaiset kertakäyttöiset ratkaisut ovat helposti epätyydyttäviä eikä niitä voida myöhemmin esimerkiksi yleistää tai muunnella.

Eräs osoitus formaalien ratkaisumenetelmien ja ohjelmointimenetelmien lähentymisestä on kiinnostus ongelmaläheisten korkean tason ohjelmointikielten kehittelyyn. Esimerkiksi PLANNER-kieli (Winograd 1972) sisälsi impli-

siittisiä, käyttäjälle näkymättömiä päättelymekanismeja. Tällöin otettiin askel kohti deklaratiivisia kieliä (ks. Winogradin (1979) tulevaisuuden ohjelmointikieliä käsitteleviä spekulaatioita), joilla kirjoitetuissa ohjelmissa ei tarvitsisi ratkaista ongelmia algoritmisesti peräkkäisillä ohjelma-askelilla, vaan ainoastaan kuvata lopputulokselle asetettavat ehdot riittävän tarkasti.

Ohjelmointikielen ja korkean tason formalismien välinen käsitteellinen ero on usein hämärä. Formalismien toteutukset voidaan helposti mieltää ohjelmointikielinä tai niiden laajennuksina. Esimerkiksi LISPissä kaikki käyttäjän määrittelemät korkean tason funktiot ovat käyttäjän kannalta aivan samanlaisia kuin systeemifunktiotkin. LISP-pohjaisessa KRL-kielessä (Knowledge Representation Language; Bobrow ja Winograd 1977b ja 1979) joka on kenties kehittynein tekoälyn ohjelmointikieli, tapahtuvat sekä ohjelmointi että ongelman ratkaisu ongelmaläheisesti, viitekehys-käsitteeseen (Minsky 1975 ja Metzing 1980) perustuen.

2.3. Laskennan kompleksisuus

Ymmärryksen teoreettisessa mallintamisessa tulee ottaa huomioon kognitiivisten kykyjemme tehokkuus tai tehottomuus eri tilanteissa. Miten on esimerkiksi mahdollista, että meille saattaa tuottaa vaikeuksia muistaa yksinkertainen puhelinnumero, kun samalla sen ihmisen kasvot, jolle aikoo soittaa, erottaisi "miljoonista"? Tietokonetoteutuksen tehokkuustarkastelut eivät ole toisarvoisia kielen ymmärtämistä tutkfttaessa, sillä ongelmien kompleksisuus ja niiden muut teoreettiset ominaisuudet eivät ole pelkästään tietokonetekniikan ratkaisulle asettamia reunaehtoja. Niiden tutkimus kuuluu tietojenkäsittelyteoriaan paljolti siksi, että niitä tässä ympäristössä vaivattomimmin voidaan tutkia. (Helppolukuinen johdatus tietojenkäsittelyteoriaan on esimerkiksi Arbib et al. 1981.)

Algoritmiteoria ja laskennan teoria (Aho et al. 1974) ovat ongelmien ja ratkaisujen teoreettisia ominaisuuksia käsitteleviä tutkimusaloja. Näiden alojen lähtökohtana on se tosiasia, että ongelman esitys- ja käsittelytavasta riippuen voi saman pulman ratkaisu vaatia hyvin erilaisia suoritusaikoja ja tietomääriä (muistia) samalla tiedonkäsittelijällä. Esimerkkinä olkoon henkilö, joka ei muista ystävänsä puhelinnumeroa ja joutuu siten turvautumaan puhelinluetteloon. Mikäli luettelon laatija ei olisi käyttänyt aakkosjärjestykseen perustuvaa tiedonesitysformalismia ja me emme tuntisi aakkosjärjestykseen perustuvaa hakumenettelyä, olisi tietyn puhelinnumeron löytäminen luettelosta lähes mahdotonta.

2.4. Psykologinen mielekkyys

Suurena periaatteellisena vaikeutena psykologisesti mielekkäiden kielen tietokoneanalogioiden toteuttamiselle on, etteivät nykyiset von Neumann -tyyppiset konearkkitehtuurit (Hill ja Peterson 1973) vastaa ihmisaivoja toiminnaltaan. Tietokoneet eivät (yleensä) käsittele tietoa rinnakkaismuodossa kuten ihmisaivot vaan peräkkäisesti. Lisäksi tietokoneen muistin rakenne ja operaatiot perustuvat erillisiin muistipaikkoihin ja osoitteisiin eivätkä assosiatiivisuuteen, jossa tieto itse toimii osoitteena kuten aivoissa.

Psykologinen mielekkyys tietokoneanalogioissa ei kuitenkaan ole mieletön tavoite: vaikka tietokoneet ja aivot perusratkaisuiltaan ovat erilaisia, voivat ne silti toimia korkeammilla tasoilla samoilla periaatteilla. Sekventiaaliset tietokoneet voivat tunnetusti simuloida rinnakkaislaskentaa mutta myös päinvastoin. Esimerkiksi looginen päättely korkealla tasolla saattaa edetä peräkkäisesti askel askeleelta kohti lopputulosta, vaikka aivosolut luonnostaan käsittelevätkin tietoa rinnakkaisesti.

2.5. Teoreettinen laskennallinen voima

Tinkimällä ohjelmistojen vasteaika- ja muistitilavaatimuksista voidaan tietokoneella simuloida laskennan teorian mukaan periaatteessa melkein mitä tahansa toimintaa. Normaalien ohjelmointikielien laskennallinen voima vastaa ns. Turingin konetta, jolla on voitu osoittaa voitavan ratkaista kaikki algoritmisesti ratkaistavissa ja pääteltävissä olevat ongelmat, mikäli aikaa ja muistia on riittävästi. Sovittaessa jostakin korkeamman tason formalismista on kuitenkin ensin varmistettava, että sen laskennallinen voima on käyttötarkoitukseensa nähden riittävä. Esimerkiksi yhteysvapaat kieliopit havaittiin aikanaan riittämättömiksi kuvaamaan luonnollisen kielen lauserakenteita. Toisaalta liian yleinen formalismi saattaa johtaa tarpeettoman yleisten ja siten hitaiden algoritmien käyttöön.

Yleisen formalismin ei kuitenkaan välttämättä tarvitse merkitä monimutkaista tai hidasta toteutusta, tarjoaahan se voimakkaampia keinoja ongelmien ratkaisemiseksi. Esimerkiksi yhteysriippuvilla semanttisilla tarkistuksilla voidaan tehostaa yhteysriippumatonta lauseenjäsennystä.

2.6. Yleispätevyys

Formalismin suunnittelussa pitäisi pyrkiä riittävään yleisyyteen tuleviakin tarpeita ajatellen. Liiallinen yleisyys on pienempi paha kuin liian ahdas näkemys varsinkin teoreettisissa tarkasteluissa. Kielen mallittamisessa on toistaiseksi pyritty ehkä liikaa nousemaan tyvestä puuhun ja unohdettu, että puuhun voi kiivetä myös toisella tavalla; nimittäin latvasta käsin katkaisemalla ensin koko puu. Varoittavana esimerkkinä siitä, mihin liian matalan tason tavoitteiden asettaminen voi johtaa, tarjoaa generatiivinen kielioppi: vaikka tämä formalismi onkin ollut käyttökelpoinen syntaksitason tarkastelussa, se on osoittautunut riittämättömäksi mallitettaessa kielen semantiikkaa ja pragmatiikkaa.

Tietojenkäsittelytermejä käyttäen voidaan sanoa, että käsiterakenteiden valinnoissa on painotettu bottom-up -strategiaa (vrt. induktio). On pidetty liian selvänä, että käsiterakenteet voidaan myöhemmin yleistää kokonaisvaltaisemmaksi formalismiksi. Top-down -näkemyksen (vrt. deduktio) mukaan taas pitäisi kielen problematiikkaa lähestyä korkeammilta tasoilta lähtien eli tutkimalla ensin kognitiivisia prosesseja yleensä. Kielen ymmärtäminenhän on vain eräs kognitiivisuuden esiintymismuoto (vrt. Goodwinin ja Heinin (1980) kognitiotieteiden paradigmojen tarkastelut). Tärkeää olisi käyttää kaikilla kielen mallintamisen tasoilla hyväksi kaikki mahdollinen tieto siitä, mihin kullakin tasolla lopulta ollaan pyrkimässä.

Peruskysymys kielen ymmärtämisen mallintamisessa on, millaisen kokonaisvaltaisen formalismin puitteissa kielen eri tasot voitaisiin kuvata niin eksplisiittisesti, että tietokoneanalogia voitaisiin toteuttaa. Aluksi on epäilemättä rajoituttava jonkin osa-alueen kuten morfologian tai syntaksin mallintamiseen, jotta formalismien käyttökelpoisuus tulisi todella punnittua. Kokonaisvaltaisuuteen pitäisi kuitenkin pyrkiä entistä ponnekkaammin. Englantilainen runoilija Philip Sidney on sanonut: "Joka kurkottaa tähtiin voi olla varma, ettei saavuta päämääräänsä, mutta voi myös olla varma siitä, että yltää korkeammalle kuin se, joka kurkottaa pensaaseen".

Yhteenvetona voidaan todeta, että luonnollisen kielen ymmärtämisen käsiterakenteiden valinnassa ja suunnittelussa tulisi arvioida erikseen ainakin niiden (1) luontevuus ja tehokkuus suunnitteluvälineenä ihmisen kannalta, (2) psykologinen, lingvistinen yms. mielekkyys, (3) teoreettinen laskennallinen voima, (4) laskennallinen kompleksisuus, (5) formalismin yleispätevyys ja (6) luontevuus ja tehokkuus tietokoneen toiminnan kannalta.

3. Nykyisten NLU-järjestelmien käsiterakenteesta

Tiedon esitysmuotojen (knowledge representation) tutkimuksen (Bars ja Feigenbaum 1981, Findler 1980 ja Metzing 1980) on sanottu olevan tämän hetken aktiivisin tekoälyn tutkimusalue. Tekoälyssä tiedon esittämisellä ei tarkoiteta pelkästään käytettäviä passiivisia (tieto)rakenteita vaan samalla myös niitä tulkitsevia proseduureja. Eihän tietosanakirjakaan ole kuin pino paperia ilman lukijaa eikä puhelinluettelolla ole merkitystä, ellei sen käyttäjillä ole tiedossaan aakkosellista hakumenettelyä.

3.1. Proseduraalien ja deklaratiivinen tieto

1970-luvun kehittyneimpien, tietopohjaisten (knowledge-based) kielenymmärtämisjärjestelmien kehittämisen myötä on proseduraaliset formalismit otettu yleiseen käyttöön. Winogradin (1972) SHRLDU-järjestelmä sekä Woodsin (Woods et al. 1972) LUNAR-systeemin proseduraalinen semantiikka, jossa lauseet viime kädessä muunnettiin eksplisiittisiksi proseduureiksi, olivat pioneeritöitä proseduraalisen tiedonesityksen tutkimuksessa. Deklaratiiviset, passiiviset rakenteet eivät Winogradin ja Woodsin tietopohjaisissa mikromaailmoissa olleet yhtä joustavasti sovellettavissa eri tilanteisiin kuin proseduurit. Proseduraalisen tiedonesityksen tärkein etu on tehokkuus, joka saadaan aikaan sovellutuskohtaisella heuristiikalla.

Formaalien esitysten proseduraalisten käsitteiden pitäisi olla sopusoinnussa tiedon esitystavan kanssa. Eräs peruste tälle on, että proseduurit, jotka prosessoivat tietoa, ovat viime kädessä itsekin tietoa. LISP-ohjelmoinnissa tämä dualistinen näkemys on realisoitu esittämällä sekä funktiot että data syntaksiltaan samanlaisina symbolisina lausekkeina. Korkeamman tason käsiterakenteissa eivät proseduraalinen ja deklaratiivinen tiedonesitys kuitenkaan läheskään aina ole yhteensopivia, vaikka ne olisivatkin LISP-pohjaisia. Hyvin erityyppisiä formaaleja esitystapoja on löydettävissä yksittäisten järjestelmien sisältä. Teoreettisesti tällaiset käsiterakenteiltaan hajanaiset ratkaisut eivät ole täysin tyydyttäviä: tieteenfilosofian Occamin veitsi ei ole suopea monimutkaisille rakenteille. Suhteessa liian rajoittuneisiin formalismeihin saattaa runsas rakenteiden käyttö kuitenkin olla käyttökelpoista.

Esimerkin menestyneestä kirjavien struktuurien järjestelmästä tarjoaa Woodsin LUNAR-systeemi, joka iästään huolimatta muistuttaa paljon nykyisiä järjestelmiä.

Aluksi päätellään LUNARissa sanojen morfologinen muoto, mikä on lähes triviaalia englannissa. (LUNARissa on muuten tekoälyn historian suurimpia toteutettuja leksikkoja eli 3500 lekseemiä. Yleensä leksikkojen koko on kertaluokkaa pienempi (ks. Cercone ja Mercer 1980). Lekseemit esitetään LUNAR-systeemissä, kuten useimmissa muissakin järjestelmissä, irrallisina LISP-atomeina. Mitään eksplisiittistä "sanakirjaa" ei ole, vaan lekseemien hauissa turvaudutaan LISP-järjestelmän ylläpitämiin objektilistoihin.

Syntaktisen analyysin suorittaa LUNARissa monipuolinen, verkkopohjainen ATN-kielioppi ja -jäsentelijä. Jäsennyksen tuloksena saadaan puu-rakenne. ATN-kieliopit ja -jäsentelijät (Bates 1978) ovat sittemmin muodostuneet eräänlaiseksi referenssiksi, johon uusia menetelmiä yleensä verrataan.

Viimeisessä, semanttisessa vaiheessa sovellutuskohtaiset produktiosäännöt muokkaavat jäsennyspuusta proseduurin (procedural semantics), joka tulkittuna osaa käsitellä tietokannassa predikaattimuodóssa esitettyä tietoa.

3.2. Modulaarisuus ja kielen tasot

LUNAR-systeemille on tunnusomaista paitsi kokonaisuuden kannalta kirjava käsiterakenteiden käyttö, niin myös tätä vastaava ohjelmiston modulaarisuus (morfologinen vaihe, syntaktinen vaihe jne.). Eri moduleissa (eli kielen eri tasoilla) on käytetty omia käsiterakenteita.

Modulaarinen suunnittelu oli 1970-luvun keskeisimpiä innovaatioita ohjelmistojen suunnittelussa. Sen soveltamisessa kielen ymmärtämisen mallittamiseen yllä esitetyllä tavalla on kuitenkin yksi paha haitta: modulaarisen ajattelun tärkeä perusprinsiippi on, ettei moduleilla ole parhaassa tapauksessa muuta interaktiota keskenään kuin syötteet ja lopputulos (vrt. vastaavuus käytettävän funktionaalisen LISP-ohjelmoinnin kanssa). Kuitenkin esimerkiksi syntaktisen analyysin yksikäsitteinen suorittaminen vaatii yleisessä tapauksessa laskennan aikaisia semanttisia tarkistuksia. Edelleen semanttiset tarkistukset yksinkertaistavat ja tehostavat jäsentelyä huomattavasti.

Eräänä hankaluutena jäsentelijöiden toteutuksissa on ollut juuri epädeterminstisyyden ja monikäsitteisyyksien takia tarvittava takaisinjäljitys (back-tracking), joka hidastaa algoritmeja. Marcus (1980) on esittänyt ratkaisuksi determinististä jäsentelijää, joka edellä olevia sanoja tai konstituentteja testaamalla kykenee aina tekemään oikean valinnan mahdollisista jäsennyksen jatko-operaatioista.

Edellä kritisoitiin kielen käsittelyn jakamista puhtaasti modulaarisiin, kielen perinteisiä tasoja vastaaviin vaiheisiin. Modulaarista suunnit-

telua sinänsä ei kritisoitu.) Ihmisaivoissa kielen tasot eivät ole erillisiä, vaan vaikuttavat vahvasti toisiinsa johtaen tehokkaaseen top-down-bottom-up -järjestelmään. Tähän mitä ilmeisimpään seikkaan ei kielitieteessä ole kiinnitetty toistaiseksi paljoa huomiota. Kielitieteen perinteiset eri tasojen mallit on nähtävä paljolti ajatusrakennelmina, jotka tukevat tutkijoiden loogista päättelyä, kielen tietoista ymmärtämistä. Kielen luonnollisen ymmärtämisen tai tajuamisen mallittaminen vaatii osittain erilaisia käsiterakenteita kuin kielen kuvaaminen edellä mainitussa analyyttisessä mielessä. Tämä seikka on varmaan osaltaan vaikuttanut siihen, ettei tekoälyn tutkijoiden parissa ole käytetty kielitieteen luomia analyyttisiä malleja niin paljon kuin voisi luulla.

Yksi käytännön syy modulaarisuuden ja siihen läheisesti liittyvään "mahdollisimman myöhäisen sitoutumisen periaatteen" suosioon on, että näiden avulla voidaan parantaa ohjelmien siirrettävyyttä osaksi jotain toista järjestelmää. Esimerkiksi erottamalla toisistaan lauseiden jäsennys syväsijarakenteille ja sovellutuskohtainen semantiikka, voidaan suunnitella yleiskäyttöisiä jäsentelijöitä, joihin myöhemmin on helpompaa liittää sovelluskohtaisia spesifejä "häntäpäitä". Modulaarinen ohjelmointitekniikka (ja funktionaalinen ajattelu) houkuttelee kuitenkin helposti modulaarisuuteen myös formalismeissa, mikä voi johtaa teoreettiseen epäyhtenäisyyteen.

Kaikki luonnollisen kielen järjestelmät eivät perustu kielen perinteisten tasojen hyväksikäyttöön. Lingvistiikan traditioita uhmaavat mm. sanaorientoituneet jäsentelijät (Riesbeck 1974, Riesbeck ja Schank 1976 ja Riegev ja Small 1981), jotka tuottavat suoraan semanttisia rakenteita pintalauseista. Ne perustuvat yksittäisten sanamerkitysten, päätteiden ja välimerkkienkin proseduraalisiin määrittelyihin (word experts). Yksi motivaatio tälle lähestymistavalle on englannin kielelle tyypillinen homonymia; esimerkiksi take-verbille luettelee Websterin sanakirja 19 merkitystä. Verbien ja substantiivien toisistaan erottaminen ilman kontekstia ei useinkaan onnistu. Suomen kielelle on sanakeskeinen jäsennystapa varteenotettava vaihtoehto rikkaan morfologian ja tämän mahdollistaman väljän syntaksin johdosta.

3.3. Tiedon ja taidon esittämisestä tekoälyjärjestelmissä

Yhtenäistä lähestymistapaa ei tiedonesityksessä (knowledge representation) tekoälyjärjestelmissä ole havaittavissa. Erityyppisiä proseduraalisia ja deklaratiivisia esitystapoja yhdistellään varsin vapaasti eklektiseen tyyliin. Yleisimmin kätettyjä tiedonesitysmuotoja (joihin liittyy myös taito tiedon käyttämiseksi) voidaan ryhmitellä seuraavasti (Barr ja Feigbaum 1981):

1) Tila-avaruusesitys ja -haku
2) Logiikka
3) Proseduraaliset esitykset
4) Semanttiset verkot
5) Produktiosysteemit
6) Viitekehykset
7) Erityiset sovellutuskohtaiset tiedonesitysformalismit

Seuraavassa esitetään lyhyet luonnehdinnat näistä lähestymistavoista.

Tila-avaruudet erilaisine hakuineen (esim. Nilsson 1980) olivat ehkä ensimmäinen laajasti tekoälyjärjestelmissä käytetty tiedonesitystapa. Sitä käytetään etenkin ongelmanratkonnassa ja peliohjelmissa, joissa tarkasteltavan maailman tila ja sen muutokset on luontevasti esitettävissä tilaverkkoina (esim. tammilauta ja pelinappuloiden säännönmukaiset siirrot).

Logiikkapohjaisten esitysten viehätys perustuu paljolti niiden ehdottomaan oikeellisuuteen. Muilla formaaleille esityksillä ei rakenteiden konsistenssiin ja muodollisesti oikeisiin päätelmiin aina päästä. Uusien totuuksien johtaminen vanhoista voidaan logiikan avulla toteuttaa mekaanisesti (vrt. resoluutiotekniikka, Nilsson 1980).

Logiikkapohjaiset formalismit eivät anna tehokkaita apuvälineitä päätellä, mikä tieto milloinkin liittyy käsiteltäviin asioihin ja miten (vrt. semanttisten verkkojen assosiatiiviset kaaret ja viitekehykset). Suuntaamaton looginen päättely vähänkin suuremmilla tietomäärillä johtaa tehottomuuteen.

Proseduraaliset tiedonesitykset ovat usein suoraviivaisin tapa ohjelmoida "järkeä" luonnollista kieltä ymmärtäviin järjestelmiin. Proseduurien avulla voidaan esimerkiksi aakkosjärjestys määritellä spesifinä ohjelmana, joka vertaa kirjaimia vastaavia binaarilukuja toisiinsa. Päinvastainen tapa tälle on määrittää aakkosjärjestys joukkona faktoja

(a b, b c, c d, ...,ä ö)

ja tätä predikaattijoukkoa tulkitsevana yleiskäyttöisenä loogisena päättelijänä. Edelleen voitaisiin nominilauseke esittää proseduurina, joka tietää miten käsitellä attribuutteja, taivutuspäätteitä jne. Tärkein proseduraalisen lähestymistavan etu on mahdollisuus suunnata heuristisesti päättelyprosesseja, mikä johtaa tehokkaisiin ratkaisuihin.

Semanttisten verkkojen käsitteen otti ensimmäisenä käyttöön Quillian (1968), joskin hieman samankaltaisia lähestymistapoja oli esitetty jo aiemmin 60-luvulla. Quillian käytti verkkoja assosiatiivisen muistin eksplisiittisenä mallina.

Semanttiset verkot (Findler 1979) koostuvat solmuista sekä näitä yhdistävistä suunnatuista kaarista. Solmut kuvaavat tyypillisesti reaalimaailman objekteja, käsitteitä ja tilanteita, kun taas solmuja yhdistävien nimettyjen kaarien avulla esitetään solmujen välisiä relaatioita. Solmujen katsotaan yleensä (esim. Maida ja Shapiro 1980) esittävän käsitteiden jne. intensiota; solmua ympäröivään verkkoon liittävien kaarien avulla taas kuvataan käsitteiden jne. ekstensiota.

Semanttiset verkot ovat varsin havainnollinen formalismi suunnittelijan kannalta. Ihminen hahmottaa kaksiulotteisia rakenteita selvästi tehokkaammin kuin yksiulotteisia; yksi kuvahan vastaa tuhatta sanaa. Hahmontunnistuksen ja digitaalisen kuvankäsittelyn tutkijat sanoisivat kyllä mieluummin: yksi kuva vastaa kymmentä tuhatta sanaa. (Vrt. kuvallisen informaation käsittelyä mm. kieliopillisen menetelmin (Pavdilis 1977).)

Toistaiseksi ei verkkoesityksille ole voitu luoda yhtenäistä semantiikkaa kuten esimerkiksi logiikan merkinnöille. Eri systeemit käyttävät varsin sekavalla ja monesti epäjohdonmukaisella tavalla erilaisia relaatioita ja käsitteitä verkoissaan. Voimakkaasti arvostelivat Woods (1975b) ja myöhemmin samassa hengessä Brachmann (1977) käytettyjä verkkonotaatioita niiden käsitteellisistä epätarkkuuksista. Brachmann esittää samalla ehdotuksen verkkoformalismin kanonisoimiseksi primitiivisten rakenteiden avulla.

Loogisten operaatioiden, eritoten kvanttoreiden esittäminen on ollut eräs verkkoformalismin ongelma. Schubertin predikaattikalkyylipohjainen esitystapa (Schubert 1976, Schubert et al. 1979) tarjoaa usein mainitun lähestymistavan näihin pulmiin.

Produktiosysteemit (Davis ja King 1975) ovat varsin paljon käytetty käsiterakenne. Produktiosysteemi koostuu periaatteessa joukosta ehto - proseduuri -pareja. Näiden ehto-osia testataan jonkin menetelmän sanelemassa järjestyksessä. Mikäli jonkin produktion ehto-osa toteutuu, voidaan vastaava proseduuri suorittaa, jolloin päästään uuteen tilaan. Produktiosysteemit on havaittu luontevaksi tavaksi kontrolloida deklaratiivisen ja proseduraalisen tiedon interaktioita. Tällä hetkellä produktiosysteemien tutkimus painottaa produktioiden suorituksen ohjauskysymyksiä (vrt. assosiatiivisuuden esittämispulmat logiikkapohjaisissa systeemeissä) ja tutkii oppivia järjestelmiä.

Edellä kuvatut tiedonesitystavat ovat luonteeltaan melko yleiskäyttöisiä: niitä on käytetty hyväksi varsin erityyppisissä tekoälyn sovellutuksis-

sa. Semanttiset primitiivit sen sijaan ovat eräs luonnollisen kielen ymmärtämisen mallittamisessa käytetty lähestymistapa, jota voidaan luonnehtia tällä alueelle spesifinä formalismina.

Semanttisten primitiivien käyttöä on kritisoitu ja puolustettu monin tavoin (Wilks 1977). On sanottu, ettei niiden avulla voida kuvata asioita riittävän vivahteikkaasti. Toisaalta kanonisten primitiivien käyttö johtaa helposti tehottomuuteen, koska korkeankin tason päättely joudutaan usein suorittamaan alkeisyksiköiden avulla. Schankin ja hänen oppilaidensa toteuttamat lukuisat keskustelevat ohjelmat ja Wilksin tutkimukset (esim. Wilks 1975) kuitenkin osaltaan todistavat, että primitiiveihin perustuvilla käsiterakenteilla voidaan toteuttaa korkeatasoisia luonnollista kieltä ymmärtäviä järjestelmiä.

Viitekehykset (Minsky 1975 ja Metzing 1980) on uusin tässä käsiteltävä tiedonesitysformalismi. Sen piirteitä ollaan parhaillaan voimakkaasti kehittämässä. Eräs kehysrakenteiden ehkä harvemmin korostettu etu formalismina on, että niitä voidaan käyttää hyväksi suhteellisen yhtenäisesti kielen ymmärtämisen eri tasoilla morfologiasta tekstilingvistiikkaan (ks. esim. Bobrow et al. 1977a). Lisäksi proseduraalinen tieto voidaan melko luontevasti sisällyttää kehyksiin (procedural attachment). Kehysten ohjelmointia varten on kehitetty korkean tason ohjelmointikieliä (mm. FRL eli Frame Representation Language (Roberts ja Goldstein 1977a ja 1977b) sekä jo aiemmin mainittu KRL). Nämä kielet ovat tavallaan LISPin laajennuksia, jossa symbolisen lausekkeen käsite on korvattu monipuolisemmin strukturoidulla kehyksellä. (Tämä on kuitenkin varsin karkea yleistys varsinkin KRL:n osalta.)

Yleensä on tiedon esitysmuodoista puhuttaessa oltu kiinnostuneita siitä, millaisessa muodossa olisi tiedon tulkinnan kannalta parasta esittää jäsennettyjä lauseita ja tietoa maailmasta. Melko vähän huomiota on kiinnitetty siihen, millaisen formalismin puitteissa myös kielen käsittely voitaisiin esittää yhtenäisesti sen lopputuloksen ja tulkinnan kanssa. Eikö samoja mekanismeja, joita joka tapauksessa tarvitaan esimerkiksi maailmanmallin perusteella kysymyksiin vastattaessa, käytetä myös kielen ymmärtämisvaiheessa? Osittain tähän problematiikkaan liittyen on Helsingin teknillisen korkeakoulun digitaalitekniikan laboratoriossa alettu kehittää algebrallisten verkkokielioppien (Ehrig 1978) pohjalta suomen kielen jäsentelijää, joka generoisi melko yhtenäisen formalismin puitteissa käskylauseista suoraan semanttisia verkkoja ja osaisi aktivoida näitä vastavat proseduurit. Työ on kuitenkin vasta alullaan.

4. Loppulause

Ilman voimakkaita rajoituksia on kielen ymmärtämisen mallintaminen tavattoman vaikea tehtävä, sillä viime kädessä siinä on kysymys mielen mallintamisesta. Ei ole ihme, että näin laajaa ongelmakenttää on lähestytty käyttäen apuna eri tieteenalojen kuten lingvistiikan, tietojenkäsittelyteorian (tekoäly) ja psykologian paradigmoja. Valitettavasti vain ovat eri aloilla suoritetut tutkimukset erilaisista lähestymistavoista johtuen monesti niin yhteensopimattomia, että ne ovat jopa hyödyttömiä naapuritieteiden kannalta katsottuna. Vastaavaa foorumia kuin USA:n Cognitive Science Society tarvittaisiin varmaan myös Suomessa raja-aitojen kaatamiseksi ja hedelmällisen yhteistyön aikaansaamiseksi. Tutkijoiden tulisi kamppailla Alexander Dumas nuoremman kolmen muskettisoturin tavoin "kaikki yhden ja yksi kaikkien puolesta" yhä parempien kielen ja mielen mallien formuloimiseksi.

Kirjallisuus

Aho, A., J. Jopcroft ja J. Ullman 1974. The design and analysis of computer algorithms. Massachusetts: Addison-Wesley.

Arbib, M., A. Kfoury ja R. Moll 1981. A basis for theoretical computer science. New York: Springer-Verlag.

Bates, M. 1978. The theory and practice of augmented transition network grammars, teoksessa Bolc (toim.) 1978.

Barr, A. ja E. Feigenbaum 1981. The handbook of artificial intelligence 1. London: Pitman Books.

Bolc, L. (toim.) 1978. Natural language communication with computers: Lecture notes in computer science 63. Berlin: Springer-Verlag.

Bolc, L. (toim.) 1980. Representation and processing of natural language. Berlin: Carl Hanser Verlag.

Bobrow, D. ja A. Collins (toim.) 1975. Representation and understanding: Studies in cognitive science. New York: Academic Press.

Bobrow, D., R. Kaplan, M. Kay, D. Norman, H. Thompson, ja T. Winograd 1977a. GUS, a frame driven dialog system, Artificial Intelligence 8.

Bobrow, D. ja T. Winograd 1977b. An overview of KRL, a knowledge representation language. Cognitive Science 1.

Bobrow, D. ja T. Winograd 1979. KRL -- another perspective. Cognitive Science 3.

Brachmann, R. 1977. What's in a concept: structural foundations for semantic networks, Int. J. Man-Machine Studies 9.

Cercone, N ja R. Mercer 1980. Design of lexicons in some natural language systems, ALLC Journal (GB) 1.

Charniak, E., C. Riesbeck ja D. McDermott 1980. Artificial intelligence programming. Hillsdale, New Jersey: Lawrence Erlbaum.

Davis, R. ja J. King 1975. An overview of production systems. Stanford University, Computer Science Dept., Memo AIM-271.

Ehrig, H. 1978. Introduction to the algebraic theory of graph grammars. Bericht nr. 78-28, Technische Universität Berlin, Fachbereich Informatik.

Findler, N. (toim.) 1979. Associative networks. New York: Academic Press.

Goodwin, J. ja U. Hein 1980. Artificial intelligence and the study of language. Research report LiTH-MAT-R-80-36, Software System Research Center, Linköping University.

Hill, F. ja G. Peterson 1973. Digital systems, hardware organization and design. New York: John Wiley & Sons.

Maida, A. ja S. Shapiro 1980. Intensional concepts in propositional semantic networks. Technical report 171, Dept. of Comp. Science, State University of New York at Buffalo.

Marcus, M. 1980. A theory of syntactic recognition for natural language. Cambridge, Massachusetts: The MIT Press.

Minsky, M. 1968. Semantic information processing. Cambridge, Massachusetts: The MIT Press.

Minsky, M. 1975. A framework for representing knowledge, teoksessa P. Winston (toim.), The psychology of computer vision, New York: McGraw-Hill.

Metzing, D. (toim.) 1980. Frame conceptions and text understanding. Berlin: Walter de Gruyter.

Nilsson, N. 1980. Principles of artificial intelligence. Palo Alto, California: Tioga Publishing Company.

Pavlidis, T. 1977. Structural pattern recognition. Berlin: Springer Verlag.

Quillian, M. 1968. Semantic Memory, teoksessa P. Winston (toim.), The psychology of computer vision, New York: McGraw-Hill.

Riesbeck, C. 1974, Computational understanding: analysis of sentences and context. Ph.D theses, Stanford University, Computer Science Dept.

Riesbeck, C. ja R. Schank 1976. Comprehension by computer: expectation-based analysis of sentences in context. Research report 78, Yale University.

Rieger, C. ja S. Small 1981. Parsing and comprehending with word experts (A theory and its realization). Report TR-1039, NSG 7253, University of Maryland.

Roberts, B. ja I. Goldstein 1977a. The FRL primer. MIT memo 408.

Roberts, B. ja I. Goldstein 1977b. The FRL manual. MIT memo 409.

Rozenberg, R. 1980. Approaching discourse computationally: a review, teoksessa Bolc (toim.) 1980.

Schank, R. 1972. Conceptual dependency, Cognitive Psychology 3.

Schubert, L. 1976. Extending the expressive power of semantic networks, Artificial Intelligence 7.

Schubert, L., R. Goebel ja N. Cercone 1979. The structure and organization of a semantic net for comprehension and inference, teoksessa Findler (toim.) 1979.

Small, S. ja C. Rieger 1981. Toward a theory of distributed word expert natural language parsing, IEEE transactions on systems, man and cybernetics, Vol. smc-11, No 1.

Wilks, Y. 1975. An intelligent analyzer and understander of English. CACM 18.

Wilks, Y. 1977. Good and bad arguments about semantic primitives. D.A.I. research report No. 42, University of Edinburgh.

Winograd, T. 1972. Understanding natural language. New York: Academic Press.

Winograd, T. 1979. Beyond programming languages. CACM 7.

Winston, P. ja B. Horn 1981. LISP. Reading, Massachusetts: Addison Wesley.

Woods, W., R. Kaplan ja B. Nash-Webber 1972. The LUNAR sciences natural language information system: final report. BBN report No. 2378, Bolt, Beranek and Newman Inc., Cambridge, Massachusetts.

Woods, W. 1975a. Syntax, semantics and speech. BBN report No. 3067, Bolt, Beranek and Newman Inc., Cambridge, Massachusetts.

Woods, W. 1975b. What's in a link: foundations for semantic networks, teoksessa Bobrow ja Collins (toim.) 1975.