# Agendipõhine tehisintellekt (LAC9800)

**Ülevaade**

Ülevaade tehisintellekti teooriast ja praktikast. Intelligentsed agendid -- otsustus- ja tegutsemisvõimelised süsteemid. Tehisintellekti probleemide kirjeldamine agentide seisukohast. Probleemide lahendamine, teadmiste esitamine ja arutlemine. Teadmiste esitamine, planeerimine, teadmistebaasile toetuv tuletusprotsess ja otsustamine. Ebakindel planeerimine ja arutlemine. Õppimine, suhtlemine tajumine ja tegutsemine.

**Sisukord**

## Agendipõhine tehisintellekt

Raamat annab lühiülevaate tehisintellekti valdkonna arengust ja seisukorrast tänapäeval vaadeldes tehisintellektiga seotud probleeme ja nende lahendusi üldises kontekstis, milleks on nn. intelligentsed agendid. Raamatus on 27 peatükki, mis on jagatud kaheksasse ossa, kus käsitletakse:

- tehisintellekti valdkonda ja intelligentseid agente üldiselt;
- probleemide lahendamist otsinguga, otsingu meetodeid ja mängude mängimist;
- teadmisi, nende esitamist ja teadmistel põhinevat arutlemist, tuletuskäike loogikas ja loogilise arutluse süsteeme;
- loogilist tegutsemist, teoreetilist ja praktilist planeerimist ning planeeritud tegutsemist;
- ebakindluse sisse toomist looglisse arutlusse ja planeerimisse ning tõenäosuslike arutlussüsteeme;
- õppimist ja sellega seotud probleeme ning närvi- ja oletusvõrke;
- suhtlemist, taju ja tegutsemist;
- tehisintellekti suhet filosoofiaga ja tehisintellekti tuleviku.

Sissejuhatuses vaadeldakse erinevaid tehisintellekti definitsioone ja erinavaid lähenemisi tehisintellektile, mis nendes definitsioonides kajastuvad.

**Tehisintellekt**

Tehisintellekti valdkonnaks määratletakse püüet mõista ja ehitada intelligentseid olemeid (*entities*). Sealjuures sisaldab tehisintellekt terve hulga alamalasid: üldiseid (tajumine, loogiline arutlus jms.) ning konkreetseid (malemäng, matemaatiliste teoreemide tõestamine, luule kirjutamine jms).

Tehisintellekti definitsioone on antud neljalt erinevalt positsioonilt:

- süsteemid, mis mõtlevad nagu inimene (Haugeland, 1985; Bellman, 1978);
- süsteemid, mis mõtlevad ratsionaalselt (Charniak ja McDermott, 1985; Winston, 1992);
- süsteemid, mis käituvad nagu inimesed (Kurzweil, 1990; Rich ja Knight, 1991);
- süsteemid, mis käituvad ratsionaalselt (Schalkoff, 1990; Luger ja Stubblefield, 1993).

Need jagunevad kahte kategooriasse -- ühed käsitlevad mõtteprotsesse ja arutlemist ning teised käsitlevad käitumist (*behavior*). Samuti mõõdavad need definitsioonid tehisintellekti edu kahe erineva kriteeriumi, kas inimese võimete või ideaalse mõiste -- ratsionaalsuse järgi. Sealjuures loetakse süsteem ratsionaalseks, kui ta teeb õigeid asju s.t. tegutseb õigesti.
Inimesele orienteeritud lähenemisviis on empiiriline teadus, ratsionaalsusele orienteeritud lähenemisviis kombineerib matemaatika ja inseneriteaduse.

Vaatleme neid lähenemisi täpsemalt:

- Inimlik tegutsemine -- tehisintellekti edu mõõdetakse Turing'i testiga, mis pakuti välja Alan Turing'i poolt 1950 kui tehisintellekti operatsionaalne (*operational*) definitsioon. Turing defineeris intelligentse käitumise kui võime saavutada inimese-sarnane tase kõikides tunnetuslikes (*cognitive*) tegevustes, mis on küllaldane küsitleja petmiseks. Test nägi ette tehissüsteemi ja inimese suhtlemise teletaibi vahendusel ja pidi õnnestuma, kui inimene pole suuteline ära arvama, et ta suhtleb tehissüsteemiga. Täielik Turing'i test sisaldab ka võimaluse tehissüsteemi taju kontrollimiseks vaadelda ning kombata samu objekte. Sellise testi läbimiseks peab tehissüsteem omama järgmisi omadusi/võimeid:
    - loomuliku keele töötlemine (vajalik suhtlemiseks);
    - teadmiste esitamine (vajalik suhtluse käigus selguvate teadmiste salvestamiseks);
    - arutlemine (vajalik küsimustele vastamiseks ja järelduste tegemiseks);
    - õppimine (vajalik kohanemiseks uute tingimustega ja korduvate olukordade üldistamiseks);
    - täieliku Turing'i testi läbimiseks lisaks veel: nägemist (objektide tajumiseks) ja motoorikat e.

robootikat (objektide käsitlemiseks).

- Inimlik mõtlemine -- põhineb tunnetuslikul modelleerimisel (*cognitive modelling*) -- tehisintellekti edu mõõdetakse võrrelduna inimese mõtlemis ja/või tunnetusprotsessiga. Selleks on vaja kõigepealt teada, kuidas inimene mõtleb. Seda on võimalik teha kas sisevaatluse või psühholoogilise katse teel. Seejärel püütakse samat protsessi modelleerida arvutil, püüdes saavutada programmi poolt loodavate arutluskäikude sarnasust sama probleemi lahendavate inimeste arutluskäikudega (näit. GPS).
  Tunnetusteadus (*cognitive science*) seob tehisintellektis loodud arvutimudelid ja eksperimentaalpsühholoogia püüdes luua inimmõistuse tegutsemise täpsed ja kontrollitavad teooriad.
- Ratsionaalne mõtlemine -- põhineb mõtlemise seadustel -- tehisintellekti edu mõõdetakse loogika seaduste alusel. Selleks on vaja luua programmid, mis oleksis suutelised loogika notatsioonis esitatud probleemidele leidma lahenduse, kui selline olemas on.
  Tehisintellekti loogikute (*logicist*) koolkond loodab selliste programmide täiustamisega luua mõistuslikke süsteeme. Sellel teel on aga kaks takistust, esiteks pole mitteformaalse teadmise esitamine loogika notatsiooni kasutades lihtne (eriti, kui teadmine pole 100% kindel) ja teiseks on suur vahe probleemi põhimõttelisel ja praktilisel lahendusel.
- Ratsionaalne käitumine -- põhineb ratsionaalsuse mõistel -- tehisintellekti edu mõõdetakse tehissüsteemi edukusega. Tehisintellekti eesmärgina vaadeldakse ratsionaalsete agentide (agent on midagi, mis tajub ja tegutseb) uurimist ja ehitamist.
  Ratsionaalsele käitumise suunal on tehisintellektis kaks eelist. Esiteks on see palju üldisem, kui ratsionaalse mõtlemise suund, kuna korrektne tuletuskäik on vaid üks ratsionaalsuse saavutamise mehhanism, ja teiseks põhineb ta rohkem teaduslikule meetodile, kui inimliku mõtlemise ja -käitumise suunad, kuna ratsionaalsus on selgelt defineeritud ning üldine.
  Kuna täieliku ratsionaalsuse saavutamine võib keerukates keskkondades olla võimatu arvutusvõimsuste puudulikkuse tõttu, vaadeldakse ka piiratud ratsionaalsust -- adekvaatset tegutsemist juhul, kui pole piisavalt aega vajalikuks arutlemiseks.

**Agendid**

Agent on midagi, mida võib vaadelda oma ümbrust andurite (*sensors*) kaudu tajuvana (*perceiving*) ja selles effektorite (*effectors*) kaudu tegutsevana (*acting*).
Ratsionaalne agent on selline agent, mis teeb õigeid tegevusi. Esimeses lähenduses on õiged tegevused need, mille tõttu agent on kõige edukam. Agendi edukust mõõdetakse soorituse mõõduga (*pefrormance measure*). Sellest, kuidas (mõõtpeaks olema agendi suhtes objektiivne) ja kunas (mõõt peaks pika aja jooksul säilitama korrektsuse) agendi edu mõõdetakse sõltub ka agendi tegevuse tulemus.

Ratsionaalsuse ja kõiketeadvuse (*omniscience*) vahel tuleb vahet teha, sest kõiketeadev agent teab ette oma tegevuse tulemust ja võib vastavalt tegutseda. Ratsionaalsus tegeleb vastavalt tajutule eeldatava eduga.
Ratsionaalsus sõltub järgnevast:

- sooritusе mõõdust, mis määrab edukuse astme,
- kõigest, mida agent senini on tajunud (tajujada -- *percept sequence*),
- kõigest, mida agent teab keskkonnast,
- tegevustest, mida agent sooritab.

Ideaalse ratsionaalse agendi definitsioon:

> *Ideaalne ratsionaalne agent peab iga võimaliku tajujada jaoks tegema seda, mis maksimiseerib tema sooritusе mõõtu, toetudes tajujadale ja sisemistele teadmistele.*

Selliste tegevuste tegemine, mille abil agent kogub tarvilist informatsiooni on ratsionaalsuse tähtis osa.

Kuna agendi käitumine sõltub vaid tema tajujadast antud ajahetkeni, võib Igat agenti kirjeldada kujutusega (funktsiooniga) tajujadadest tegevustesse. Ideaalne kujutus kirjeldab sellisel juhul ideaalset agenti, määratledes, milliseid tegevusi agent peab tegema. Kui agendi teod põhinevad täielikult

sisseehitatud teadmistele nii, et agendi tegevus ei sõltu tema poolt tajutavast, üteldakse, et agendil puudub autonoomia (*autonomy*). Süsteem on autonoomne sedavõrd, kuivõrd selle käitumine on määratud süsteemi oma kogemusega. Intelligentse agendi disainimisel tuleb leida õige vahekord sisseehitatud teadmiste ja õppimisvõime vahel.

Agendi programmi disainimiseks, peab olema hea ettekujutus agendi tajust, tegevustest, sihtidest ja keskkonnast (*PAGE -- Percepts, Actions, Goals, Environment*).

***Agentide struktuur***

Tehisintellekti ülesanne on disainida agendi programm: funktsioon, mis realiseerib kujutuse tajust tegevustesse. Eeldatavalt töötab see programm mingil arvutusseadmel, mida nimetame arhitektuuriks. Arhitektuur võib koosneda arvuti(te)st, eriotstarbelisest riistvarast (andurid ja effektorid) ja tarkvarast, mis sidestab agendi programmi konkreetsest riistvarast lahti. Üldiselt teeb arhitektuur tajutu agendi programmile kättesaadavaks, täidab programmi ja kui tegevusi genereeritakse juhib effektoreid. Kokkuvõttes

*agent = arhitektuur + programm*

Arhitektuur võib ka tervikuna tarkvarast koosneda. Tarkvaraagendid (*software agents*) või tarkbot'id (*softbots*) võivad eksisteerida väga keerukates keskkondades (näiteks trakbot, mis peab oma klienti huvitavad uudised ise leitud uudiste allikatest välja otsima).

Kõik agendi programmid on sama struktuuriga: väliskeskkonna tajumine ja tegevuste genereerimine:

```
function SKELETON-AGENT(percept) returns action
    static:  memory, the agent's memory of the world
    memory <-- UPDATE-MEMORY(memory, percept)
    action <-- CHOOSE-BEST-ACTION(memory)
    memory <-- UPDATE-MEMORY(memory, action)
    return action
end
```

Igal agendi programmil on sisemised andmestruktuurid, mida uuendatakse väliskeskkonna tajumisel ja mida töötlevad otsustusprotseduurid, et genereerida tegevusi, mis arhitektuurile edastatakse. Sihi või soorituse mõõt pole üldistatud agendi programmi osa kuna soorituse mõõtu rakendatakse väliselt agendi tegevuse hindamisel ja ilma seda teadmata saab häid tulemusi saavutada.

Üldjuhul on võimalik kujutust tajutust tegevustesse esitada tabelina see aga omab nelja puudust:
- ka väga lihtsa agendi jaoks peab tabel olema äärmiselt suur;
- tabeli ehitamine võtakse agendi programmi disainieril väga kaua aega;
- agent pole autonoomne (kui keskkond muutub ootamatult siis agent ebaõnnestub);
- kui agent oleks ka õppimisvõimeline võtaks õppimine suure tabeli puhul väga kaua aega.

Järgnevalt vaatleme 4 agendi programmi tüüpi:
- Lihtsad refleksiivsed agendid (*simple reflex agents*);
- Maailma jälgivad agendid (*agents that keep track of the world*);
- Sihipõhised agendid (*goal-based agents*);
- Kasupõhised agendid (*utility-based agents*).

*Lihtsad refleksiivsed agendid.*Töötlus toimub kasutades lihtsaid tingimus-tegevus reegleid (*condition-action rule*, *production*, *if-then rule*), mis võivad olla õpitud või sisseehitatud. Lihtsa refleksiivse agendi programmis on funktsioon CHOOSE-BEST-ACTION realiseeritud otsinguna reeglite hulgal vastavalt reeglite tingimustele ja agendi poolt tajutule ning leitud reegli kasutamine järgneva tegevuse leidmiseks.

*Maailma jälgivad agendid.* Eelnev agent töötab õigesti vaid siis, kui õige otsuse saab teha vaid jooksvalt tajutu põhjal. Kuna andurid ei suuda kujutada täielikku keskkonna olekut, on vaja agendil säilitada sisemist olekuinfot, et eristada keskkonna olekuid. Selline sisemine olekuinfo vajab kahte tüüpi teadmisi

agendi programmis: esiteks kuidas keskkond areneb agendist sõltumatult ja teiseks kuidas agendi enda etegvused mõjutavad keskkonda.

*Sihipõhised agendid.* Teadmistest keskkonna kohta ei piisa alati, et otsustada, mida teha. Lisaks olekuinfole vajab agent ka mingit sihiinfot (*goal information*), mis kirtjeldab soovitavat olukorda. Otsing (*search*) ja planeerimine (*planning*) on tehisintellekti osad, mis tegelevad sihi saavutamiseks vajaliku tegevuste jada leidmisega. Refleksiivse agendi korral on agendi programmi koostamisel juba õiged vastused kõikidele juhtumitele leitud aga sihipõhine agent peab selle ise leidma tegevuse käigus. Refleksiivne agent on selles suhtes äärmiselt effektiivne aga paindumatu, kui sihipõhine agent on äärmiselt painduv (uus käitumine tekib kohe kui antakse ette uued sihid) aga ebaeffektiivne.

Sihipõhise agendi programmis on funktsioon CHOOSE-BEST-ACTION realiseeritud planeerimisega või loogilise arutlusega, mis lähtub agendi poolt tajutust, keskkonna mudelist, sisseehitatud teadmistest ja sihist.

*Kasupõhised agendid.* Sihi omamine pole piisav, et tekitada head käitumist, kuna sihti on võimalik saavutada mitmeti. Sihid ei anna mingit hinnangut sihi saavutamiseks teostatud tegevuste jadale või saavutatud keskkonna olekutele. Keskkonna olekuid võib agendi seisukohalt hinnata nende kasulikkuse poolest agendile. Kasulikkus on funktsioon, mis kujutab keskkonna olekute hulga reaalarvude hulka. Igale keskkonna olekule vastab reaalarv, mis iseloomustab antud keskkonna oleku kasulikkust agendile. Kasulikkus võimaldab otsustada vasturääkivate sihtide ja valida võrdvõimalike vahel.

Kasupõhise agendi programmis on funktsioon CHOOSE-BEST-ACTION realiseeritud planeerimise või loogilise arutlusega, mis lähtub agendi poolt tajutust, keskkonna mudelist, sisseehitatud teadmistest ja sihist ning kasutab tegevuste valikul saavutatavate olekute kasulikust.

Igal ratsionaalsel agendil on olemas ilmutatud kasufunktsioon, mis lubab tal teha ratsionaalseid otsuseid.

***Keskkonnad***

Keskkondi võib iseloomustada järgmiselt:

- täielikult tajutavad (*accessible* vs. *inaccessible*) -- kui agendi andurid annavad agendile juurdepääsu kogu keskkonna olekule ütleme, et keskkond on täielikult tajutav; selline keskkond võimaldab agendil loobuda keskkonna olekut peegeldavast olekuinfost;
- ennustatav (*deterministic* vs. *nondeterministic*) -- kui keskkonna järgmine olek on täielikult määratud praeguse oleku ja agendi poolt valitud tegevus(t)ega ütleme, et keskkond on ennustatav; täielikult tajutav ja ennustatav keskkond võimaldab agendil loobuda määramatusest (*uncertainty*); kui keskkond pole täielikult tajutav võib ta paista ennustamatuna (eriti kui keskkond on keerukas) seega võib keskkonda vaadelda agendi seisukohast ennustatavana või ennustamatuna;
- tükeldatav (*episodic* vs. *nonepisodic*) -- tükeldatavate keskkondade puhul võib agendi kogemused esitada tükkidena, mis koosnevad agendi tajust ja tegevusest; tegevuse kvaliteet sõltub vaid antud tükist ja järgnevad tükid ei sõltu eelnevates tehtud tegevustest; tükeldatavates keskkondades tegutsevad agendid ei pea ette mõtlema;
- staatiline või dünaamiline (*static* vs. *dynamic*) -- kui keskkond muutub ka siis kui agent tegeleb otsustamisega on keskkond dünaamiline; staatilised keskkonnad võimaldavad agendil mitte arvestada aega; kui keskkond ei muutu agendi otsustamisprotsessi ajal aga agendi sooritus mõõt muutub ütleme, et keskkond on pooldünaamiline;
- diskreetne või pidev (*discrete* vs. *continuous*) -- kui on olemas piiratud arv selgelt defineeritud tajuelemente ja tegevusi ütleme, et keskkond on diskreetne.

Kõige keerulisem juhutum on mitte-täielikult tajutav, ennustamatu, tükeldamatu, dünaamiline ja pidev keskkond.

Agente võib testida ja uurida keskkonna programmidega, mis simuleerivad agendi jaoks keskkonda ning samuti hindavad agendi sooritust.

Keskkonna programm haldab keskkonna mudelit, mida ta kasutab agentidele tajutava info andmiseks ning mida ta muudab vastavalt agentide tegevustele. Lisaks haldab keskkonna programm ka agentide soorituse mõõte.

Tavaliselt on agent disainitud töötamaks keskkondade klassis -- hulgas erinevates keskkondades. Et agenti testida on vaja keskkondade generaatorit, sellisel juhul huvitab meid agendi keskmine sooritus hinnang mingi keskkondade klassi jaoks. Tähtis on jälgida, et agendi programm ei omaks otsest juurdepääsu keskkonna programmi poolt hallatavale olekuinfole. Agendi programm peab enda olekuinfo looma vaid tajutu põhjal.

Järgnevalt vaadeldakse raamatus erinevaid tehisintellekti alla kuuluvaid valdkondi lähtudes esitatud agendi mudelist.

**Probleemide lahendamine**

Probleemide lahendamist käsitledes vaadeldakse probleeme lahendavat (*problem-solving*) agenti. Probleeme lahendavad agendid otsustavad, mida teha, leides tegevuste jadasid, mis viivad soovitavatesse olekutesse.

Esimeseks etapiks probleemi lahendamisel on, praegusele olukorrale põhinev, sihi formuleerimine. Sihti vaatleme kui hulka keskkonna olekuid, milles siht on saavutatud. Tegevusi võib vaadelda keskkonna olekute vahel üleminekuid põhjustavatena. Teiseks etapiks probleemi lahendamisel on probleemi formuleerimine -- otsustamine, milliseid tegevusi ja olekuid arvestada. Järgnevaks etapiks on sellise tegevuste jada otsimine, mis viiks sihile -- otsingualgoritmi sisendiks on probleemi kirjeldus ja tulemuseks lahendus tegevuste jada kujul. Kui lahendus on leitud, tuleb leitud tegevused sooritada, seda nimetetakse täitmise faasiks.

Probleemide tüübid:

- üksiku oleku (*single-state*) probleemid -- keskkond on täielikult tajutav ja ennustatav, seega saab agent täpselt välja arvutada, millises olekus keskkond on peale igat tegevuste jada;
- mitme oleku (*multiple-state*) probleemid -- keskkond pole täielikult tajutav aga on ennustatav, seega peab agent arvutama keskkonna olekute hulkadega; samuti võib käsitleda olukorda, kus keskkond pole ennustatav;
- tingimuslikud (*contingency*) probleemid -- kui keskkond pole ennustatav, peab agent välja arvutama mitte tegevuste jada vaid tegevuste puu, milles on tingimused, mida tuleb täitmise käigus kontrollida; tingimuslikud probleemid võimaldavad disainida agente, mis asuvad tegutsema enne sihti garanteeriva plaani leidmist ja planeerivad tegevuse käigus;
- uurimisprobleemid (*exploration problems*) -- kui agendil puudub informatsioon oma tegevuste mõjust keskkonnale (keskkonna mudel), peab agent sooritama katseid uurides keskkonda (selle olekuid ja oma tegevuste mõju keskkonnale); selline tegutsemine menutab otsingut, aga see on otsing tegelikus keskkonnas mitte keskkonna mudelis.

Probleem on info, mida agent kasutab selleks, et otsustada mida teha. Probleemi definitsiooni elementideks on olekud ja tegevused. Et neid formaliseerida on vaja:

- algolekut (*initial state*), milles agent teab ennast olevat;
- agendi poolt sooritavate tegevuste hulka, tegevuse kirjeldust kujutusena olekute hulgast olekute hulka nimetatakse operaatoriks (*operator*), järgnevusfunktsioon (*successor function*) näitab iga oleku jaoks, milliseid olekuid sellest alustades võib saavutada suvalise (ühe) tegevusega;
- sihitesti (*goal test*), mis võimaldab ära tunda sihtoleku; siht võib olla kirjeldet kas sihtolekute hulgana või sihtoleku omadusena;
- tee maksumuse (*path cost*) funktsiooni, mis seob tegevuste jadaga mingi hinna/kaalu.

Lihtsa probleeme lahendava agendi programmis on funktsioon CHOOSE-BEST-ACTION realiseeritud

sihtide formuleerimisega, probleemi formuleerimisega ning lahenduse otsimisega või eelnevalt leitud lahenduse täitmisega, mis lähtub agendi poolt tajutust, keskkonna mudelist ja sisseehitatud teadmistest.

***Probleemide lahendamine otsinguga***

Otsinguatlgoritmi väljund on tee algolekust sihtolekusse. Probleemiruum (*problem space*) on kõikide olekute hulk, millesse võib suvalise tegevuste jadaga algolekust jõuda. Mitme oleku probleemide puhul on probleemiruumi korral tegemist olekute hulkade hulgaga.

Otsingu effektiivsust saab mõõta leitud tee maksumusega ja otsingu maksumusega (otsinguks vajalik aeg ja mälumaht). Otsingu koguhind on leitud tee maksumuse (*online cost*) ja otsingu maksumuse (*offline cost*) summa. Et otsingu koguhinda dünaamiliste keskkondade puhul maksimeerida peab agent jagama oma ressursse otsingu (planeerimise) ja tegutsemise (plaani täitmise) vahel.

Olekute ja tegevuste esituse valikul on tähtis mitteoluliste detailide kõrvaldamine -- abstraheerimine (*abstraction*). Ilma võimeta ehitada kasulike abstraktsioone pole intelligentsete agentide tegutsemine reaalses maailmas võimalik.

Probleemi õige formuleerimine (olekute ja operaatorite valik) mõjutab oluliselt otsinguruumi suurust. Otsimisel rakendatakse operaatoreid jooksvale olekule ja genereeritakse uute olekute hulk. Seda protsessi nimetatakse oleku laiendamiseks (*expanding*). Valik, millist olekut esimesena laiendada on määratud otsingu strateegiaga. Otsinguprotsessi võib vaadelda kui olekute ruumile kujutatud otsingupuu ehitamist. Otsingupuu juureks on algolek. Igal sammul valib otsingualgoritm ühe otsingupuu lehtedest, et seda laiendada. Laiendamist ootavate sõlmede hulka nimetame rajaks (*fringe* või *frontier*).

Õige otsingustrateegia otsimisel tuleb arvestada, kas:

- täielikus (*completeness*) -- kas strateegia leiab lahenduse, kui see on olemas;
- ajaline keerukus (*time complexity*) -- kui kaua võtab aega lahenduse leidmine (keerukus sõltub olekuruumi hargnemisfaktorist ja madalaima lahenduse sügavusest);
- mahuline keerukus (*space complexity*) -- kui palju mälu on vaja otsingu läbiviimiseks;
- optimaalsus (*optimality*) -- kas strateegia leiab võimalikest lahendusist parima.

On olemas kuus otsingustrateegiat, mida võib nimetada mitteinformeeritud otsingu (*uninformed search*) meetoditeks. Mitteinformeeritud otsingut nimetatakse vahel ka pimedaks (*blind*) otsinguks. Strateegiad, mis hindavad otsingu käigus laiendamiseks valitavat sõlme lisainformatsiooni alusel nimetatakse informeeritud otsingu (*informed search*) või heuristilise (*heuristic*) otsingu meetoditeks.

Mitteinformeeritud otsingu meetodeid eristatakse sõlmede laiendamise järjekorra järgi:

- laiuti otsing (*breadth-first search*) laiendab madalaimat sõlme otsingupuus esimesena, ta on täielik, optimaalne ühikulise maksumusega operaatorite suhtes ja tema ajaline ning mahuline keerukus on O(bühtlase hinnaga otsing (*uniform cost search*) laiendab odavaimat lehte otsingupuus esimesena, ta on täielik, optimaalne ka kui operaatorite maksumused on erinevad ja tema keerukus on sama mis laiuti otsingul;
- sügavuti otsing (*depth-first search*) laiendab sügavaimat sõlme otsingupuus esimesena, ta pole ei täielik ega optimaalne ja tema ajaline keerukus on O(bpiiratud sügavusega otsing (*depth-limited search*) seab sügavuti otsingule sügavuspiiri, kui sügavuspiir juhtub olema sama suur, kui madalaim lahend, siis on ajaline ja mahuline keerukus minimaalsed;
- iteratiivselt süvenev otsing (*iterative deepening search*) kasutab piiratud sügavusega otsingut järjest suureneva sügavuspiiriga, kuni lahendus leitakse, ta on täielik, optimaalne ja tema ajaline keerukus on O(bkahesuunaline otsing (*bidirectional search*) on otsing üheaegselt alates lähteolekust sihi poole ja alates sihtolekust lähteoleku poole. Kahesuunalise otsingu otsingu ajaline keerukus on O(2*b*
Kitsenduste rahuldamise probleem (*constraint satisfaction problem*), mis rahuldab lisaks põhiprobleemile veel mingeid struktuurilisi omadusi. Kitsenduste rahuldamise probleemis on olekud defineeritud muutujatena (*variables*) ja sihitest määrab kitsenduste (*constraints*) hulga, mis peab nende väärtustel

kehtima. Kitsendused võivad olla absoluutsed (lahend ei tohi neid rikkuda) ja eelistuslikud (nende abil saab leida eelistatud lahendi). Igal muutujal kitsenduste rahuldamise probleemis on väärtuste hulk (*domain*), mis võib olla diskreetne või pidev. Diskreetsete ja lõplike väärtuste hulkade korral võib kitsendusi esitada lubatud väärtuste kombinatsioonide loeteluga.

Sügavuti otsingule võib kitsenduste rahuldamise probleemi puhul lisada järglaste genereerimise sammu ette kontroll, kas ühtegi kitsendust pole rikutud -- tulemuseks on tagasiastuv otsing (*backtracking search*), mis pöördub otsingupuus tagasi, et teist teed proovida.

Et vältida lahendust mitte sisaldavate otsingupuu harude katsetamist, tuleb lisada otsingualgoritmile ette vaatamine (*forward checking*) -- iga kord, kui muutujale antakse väärtus, kustutatakse väärtust veel mitte omavate muutujate väärtuste hulkadest kõik need väärtused, mis on antud väärtusega vastuolus; kui mingi muutuja väärtuste hulk saab tühjaks, pöördutakse otsingupuus tagasi. Ette vaatamine on kaarte kokkusobivuse (*arc consistency*) kontrollimise erijuhtum -- olek on kaar-kokkusobiv, kui igal muutujal on väärtus, mis ei riku ühtegi selle muutuja kitsendust. Kaarte kokkusobivust saab kontrollida eemaldades väärtusi, mis mingit kitsendust rikuvad (see omakorda võib esile kutsuda mingite uute kitsenduste rikkumise ja viib kitsenduste levitamisele (*constraint propagation*), kus valikuvõimalusi vähendatakse). Kaarte kokkusobivuse kontrolli kasutatakse sageli otsingule eelneva sammuna.

Heuristilised otsingud on:

- Parim-enne otsing (*best-first search*) on üldistatud otsing, milles (mingi mõõdu järgi) vähima maksumusega sõlmed laiendatakse esimesena:
  - ahne otsing (*greedy search*) vaatleb lahendusele lähemat sõlme ja minimiseerib eeldatavat sihile jõudmise maksumust h(n), ahne otsing pole ei optimaalne ega täielik;
  - minimaalse maksumusega tee: A* otsing (*minimizing the total path cost: A* search*) vaatleb lühimat lahendust läbi antud sõlme; minimiseerib f(n) = g(n) + h(n) (kombineerides ühtlase hinnaga otsingu ja ahne otsingu) kui on garanteeritud, et h(n) ei hinda kunagi üle (*addmissible heuristic*) ning korduvaid olekuid hallatakse; A* otsing on täielik, optimaalne ja optimaalsete otsingualgoritmide seas optimaalselt effektiivne, samas on tema mahuline keerukus eksponentsiaalne kui heuristilise funktsiooni viga kasvab kiiremini tegeliku tee maksumuse logaritmist:

    $$|h(n)-h^*(n)| \leq O(\log h^*(n))$$

    kus $h^*(n)$ on n-ist sihile jõudmise tegelik maksumus.
- Piiratud mahuga otsing (*memory bounded search*)
  - Iteratiivselt süvenev A* otsing (*Iterative deepening A* search (IDA*)*) on sügavuti otsing, mis kasutab lahenduse maksumuse piiri suurendamist;
  - Lihtsustatud piiratud mahuga otsing (*simplified memory-bounded A* (SMA*)*) kasutab rohkem mälu, kui IDA* (kasutab kogu kättesaadavat mälu) vältides korduvaid olekuid (kui rohkem mälu pole võimalik kasutada kustutab mälust kõrgema hinnaga sõlmed); on täielik, kui kättesaadav mälu võimaldab esitada madalaimat optimaalset lahendust;
- Keelatud otsingu (*tabu search*) algoritmid, mis säilitavad olekute hulka, kuhu ei tohi uuesti sattuda;

Heuristiliste algoritmide ajaline keerukus sõltub heuristilise funktsiooni kvaliteedist. Sageli on lõdvendatud probleemi (*relaxed problem*) täpse lahenduse hind heaks heuristikaks algse probleemi jaoks. Kitsenduste rahuldamise probleemi korral on heaks heuristikaks kõige kitsendatuma muutuja (*most-constrained-variable*) (vähima võimalike väärtuste hulgaga) valimine ja sellele kõige vähem kitsendava väärtuse (*least-constraining-value*) valimine.

Iteratiivse parandamise algoritmid (*iterative improvement algorithms*) -- kõik olekud on laotatud pinnale, mille iga punkti kõrgus vastab antud punktile vastava oleku hinnale.

- mäkke ronimine (*hill-climbing*) -- algoritm peatub leides lokaalse maksimumi; algoritm on sunnitud platool (kus hinnafunktsioon on konstantne) juhuslikult liikuma; pinna harjad võivad põhjustada algoritmi tsüklilist võnkumist.

- gradientlaskumine (*gradient descent*) -- funktsiooniks on hind
- simuleeritud külmutamine (*simulated annealing*) -- kui algoritm on sattunud lokaalsele maksimumile lubatakse tal liikuda mingi aeg allamäge; selliste "halbade" sammude tõenäosus sõltub algoritmi töötamise ajast (väheneb); kui seda tõenäosust vähendada küllalt aeglaselt, siis leiab algoritm globaalse optimumi.

***Mängude mängimine***

Mänguprogrammidel on tegemist tingimuslike (*contingency*) probleemidega kuna vastane toob sisse ebakindluse (*uncertainty*). Suur keerukus toob sisse uut tüüpi ebakindluse -- kuna puudub võimalus (ajalise ja mahulise keerukuse tõttu) arvutada välja suvalise käigu täpseid tagajärgi. Lisaks on mängudes aeg tavaliselt piiratud.

Mängu kui probleemi võib formaliseerida koosnevana:

- algolekust (*initial state*) -- seis mängulaual ja alustaja;
- operaatorite (*operators*) hulgas -- lubatud käigud;
- lõputestist (*terminal test*) -- määrab kunas mäng on läbi; olekuid kus mäng on läbi nimetatakse lõpuolekuiks (*terminal states*);
- kasufunktsioonist (*utility function*) -- annab mängu tulemusele numbrilise esituse (kes võitis ja kui palju).

Mängija peab leidma strateegia (*strategy*), mis viib sellisesse lõpuolekusse, kus tema on võitja hoolimata sellest, mis vastane teeb.

Kahe mängija täieliku infoga mängudes võib minimax algoritm, eeldusel, et vastane mängib perfektselt, nummerdades kogu mängupuu määrata parima käigu. Minimax otsus maksimeerib kasu eeldusel, et vastane mängib perfektselt et seda minimeerida.

Mittetäiuslike otsuste (*imperfect decisions*) tegemiseks asendatakse kasufunktsioon hinnangu funktsiooniga (annab antud positsiooni kasulikuse hinnagu) ja lõputest katkestustestiga (lihtsaim on otsingupuu sügavusest sõltuv). Katkestamist saab rakendada vaid rahulikel (*quiescent*) positsioonidel, kus väärtus lähemas tulevikus oluliselt ei muutu. Sellise rahuliku positsiooni otsimist nimetatakse rahu otsinguks (*quiescence search*). Horisondi probleem (*horizon problem*) ilmneb, kui vastane võib teha lõplikult vältimatu käigu, mis tekitab palju kahju aga mille toimumine vahepealsete takistavate käikude tõttu nihkub "horisondi taha".

Otsingupuu haru vaatluse alt välja jätmist nimetatakse kärpimiseks (*pruning*). Alfa-beeta kärpimine rakendatuna minimax puule saab sama lahenduse, mis minmax aga jätab kõrvale harud, mis tulemust ei mõjuta. On näidatud, et alfa-beeta algoritm on automaatselt tuletatav minimax algoritmist rakendades üldisi programmide transformeerimise tehnikaid.

Mitmed mängud simuleerivad reaalsust tuues sisse juhuslikkuse (näiteks täringuvisked). Sellise mängu puu peab sisaldama lisaks MAX ja MIN sõlmedele ka juhuslike sõlmi. Minimax väärtuse asemel saame nüüd opereerida vaid keskmise või oodatud väärtusega (*expected value*).

Sihile suunatud arutelu (*goal-directed reasoning*) või planeerimine (*planning*) võimaldab otsingut üldse vältida.

Agent, millel on sihid ja mis otsib lahendusi sihile jõudmiseks on parem kui selline agent, mis vaid reageerib oma keskkonnale.

**Teadmised ja arutlemine**

Laiendame agentide võimeid, lisades neile võime üldiselt loogiliselt arutleda. Loogiline, teadmistepõhine agent alustab mingite teadmistega maailma ja oma tegevuste kohta, kasutab loogilist arutlemist maailma

kirjelduse säilitamiseks vastavalt tajutule ja järeldab tegevused, mis viivad sihile.

Intelligentsed agendid vajavad teadmisi keskkonna kohta, et jõuda headele otsustele. Teadmised sisalduvad agendis teadmiste esitamise (*knowledge representation*) keele lausetena (*sentences*), mis on salvestatud teadmistebaasi (*knowledge base*). Teadmispõhine agent koosneb teadmistebaasist ja tuletusmehhanismist (*inference mechanism*). Teadmispõhine agent tegutseb salvestades lauseid keskkonna kohta oma teadmistebaasi, kasutab tuletusmehhanismi, et tuletada uusi lauseid ja kasutab neid, et võtta vastu otsuseid, mida teha.

Teadmiset esitamise keel on määratud süntaksi ja semantikaga, mis määravad lausete struktuuri ja selle kuidas laused on seotud faktidega keskkonnas. Lause interpretatsioon (*interpretation*) on fakt, millele see lause viitab. Kui lause viitab faktile, mis kuulub keskkonda, siis see lause on tõene.

Teadmisi tegevustest ja nende mõjust võib esitada olukorraarvutusena (*situation calculus*). See teadmine lubab agendil keskkonda jälgida ja tuletada planeeritud tegevuste mõju.

On võimalik kirjutada diagnostikareegleid (*diagnostic rules*), mis leiavad tajutust eeldusi keskkonna kohta ja põhjusreegleid (*causal rules*), miskirjeldavad kuidas keskkonna seisund põhjustab tajutut. Viimased on paindlikumad aga tuletuskäigus raskemad kasutada.

***Teadmiste baasi ehitamine***

Teadmiste baasi ehitamist nimetatakse teadmiste ehitamiseks (*knowledge engineering*). Teadmiste kogumist nimetatakse (*knowledge acquisition*). Sellised mõisted, nagu aeg, muudatus, objekt, aine/materjal (*substance*), sündmus, tegevus, raha, mõõt (*measure*), jne. tulevad ette igas valdkonnas (*domain*). Kõige üldisemate mõistete esitamist nimetatakse (*ontological engineering*).

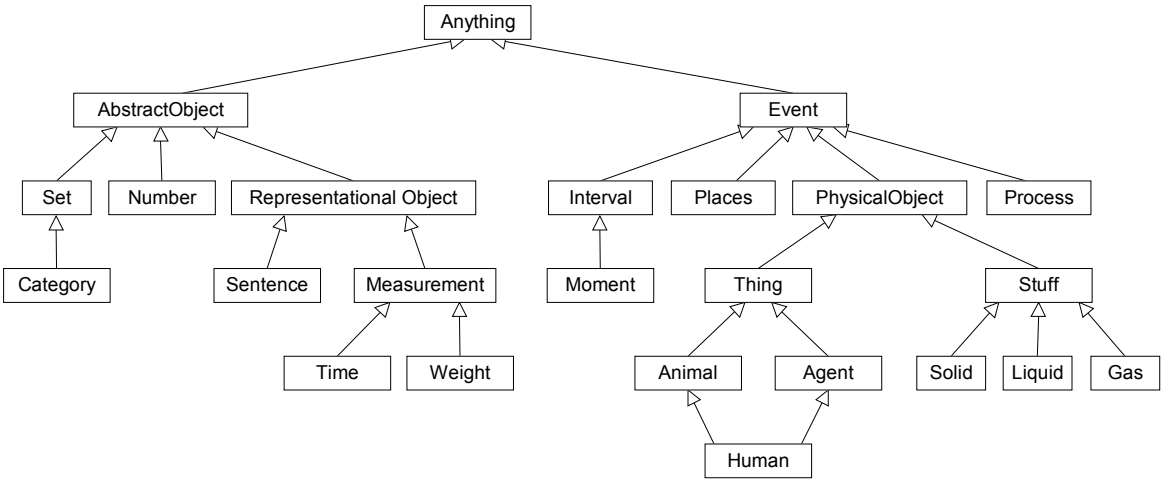
Hea teadmiste esitamise keel peab olema väljendusvõimeline (*expressive*), ühene (*unambiguous*), kontekstist sõltumatu (*context-insensitive*) ja effektiivne. Teadmistebaas peab lisaks olema veel selge (*clear*) ja korrektne (*correct*).

Teadmiste esitamine mingi valdkonna (*domain*) kohta toimub mitmes järgus. Esimene mitteformaalne järk sisaldab otsuseid, milliseid objekte ja suhteid objektide vahel on vaja esitada (kirjeldatakse ontoloogia (*ontology*)). Ontoloogia on valdkonna mõistete loetelu, millega otsustatakse mis on olemas aga ei otsustata nende asjade omadusi ja seoseid. Siis valitakse sõnastik ja kasutatakse seda valdkonna üldiste teadmiste kodeerimiseks. Peale konkreetsete probleemiisendite (*problem instances*) kodeerimist saab kasutada automaatseid tuletusprotseduure nende lahendamiseks.

Igas piisavalt keerukas valdkonnas tuleb ühtlustada (*unify*) erinevate alade teadmised, kuna probleemide lahendamine võib puudutada mitmeid teadmiste alasid üheaegselt. Üldine ontoloogia peab olema rakendatav enam-vähem igas erivaldkonnas (kui lisada valdkonnale iseloomulikud aksioomid).

Üldisse ontoloogiasse kuuluvad:

- Kategooriad (*categories*) -- kategooriad võib defineerida kui hulgad, mis sisaldavad samade omadustega objekte; kategooriad ise on samuti objektid ja neid võib siduda üldisesse klassifikatsioonihierarhiasse (*taxonomic hierarchy*).




Kategooriaid saab esitada unaarsete predikaatidena. Kategooriad on objektistatud (*reify, reification*) (objektistamine on predikaadi või funktsiooni tegemine keele objektiks), see lubab teha otsustusi kategooria enda kohta. Kategooriad organiseerivad ja lihtsustavad teadmistebaasi pärivuse (*inheritance*) abil. Kategooriasse kuuluvad objektid pärivad kategooria omadused ja alamklassi suhe organiseerib kategooriad klassifikatsioonihierarhiasse (*taxonomy* või *taxonomic hierarchy*). Kaks kategooriat on kattumatud (*disjoin*) kui neil pole ühiseid liikmeid. Kattumatud täielikult dekomponeerivad (*exhaustive decomposition*) on tükeldus (*partition*).

- Mõõdud (*measures*) -- mitmed omadused seovad objekte mingit tüüpi suurustega (mass, vanus, hind, ...), mida nimetame mõõtudeks. Mõõdud on seotud mõõtühikutega.
  Eeldame, et universium sisaldab abstraktseid "mõõduobjekte", mille nimeks on ühikut esitava funktsiooni ja numbri kombinatsioon. Mõõdul ei pea sugugi olema numbrilist väärtust, mõõdu tähtsaim omadus on tema järjestatavus (monotoonsed suhted mõõtude vahel on kvalitatiivse füüsika (*qualitative physics*) aluseks).
- Liitobjektid (*composite objects*) -- sageli kuuluvad objektid kategooriatesse oma sisestruktuuri tõttu. Objekt, millel on osa (või osi) on liitobjekt. Liitobjektide kategooriaid iseloomustab nende objektide struktuur (osad ja nendevahelised suhted). Üldistatud skeem (*schema*) või skript (*script*) võib defineerida suvalise liitobjekti struktuuri. Osadest koosneva objekti mõningaid omadusi, võib vaadelda tulenevatena osade omadusist. Kasulikud on ka liitobjektid, millel pole struktuuri (mille struktuuri ei vaadelda).
- Aeg (*time*), ruum (*space*) ja muutus (*change*) -- et käsitleda tegevusi ja sündmusi, millel on erinev kestvus ja mis võivad samaaegselt toimuda, tuleb laiendada ontoloogiat ajaga. Universium on pidev nii ajalises kui ruumilises mõõtmes, ajad, kohad ja objektid on selle osad.
  Kuna olukorraarvutus ei võimalda vaadelda paralleelseid tegevusi/sündmusi, mille kestvused on erinevad ja mille mõju sõltub kestvusest, toome sisse sündmusarvutuse (*event calculus*) -- vaatleme universiumi ruumilist ja ajalist mõõdet. Sündmus (*event*) on tükike sellist universiumit. Sündmusel võib olla alamsündmusi (*subevents*). Intervall (*interval*) on sündmus, mis sisaldab alamsündmusina kõiki teatud aja jooksul toimunud sündmusi. Intervallid on universiumi ajalised sektsioonid. Kui olukorraarvutuses teatud fakt on tõene mingis olukorras, siis sündmusarvutuses toimub sündmus alati mingis intervallis. Nagu kõiki muid objekte võib ka sündmusi kategoriseerida. Ajaintervallid on tükeldatud momentideks (*moments*), mille kestvus on null, ja laiendatud (*extended*) intervallideks. Kohad (*places*) on nagu intervallidki aegruumi lõigud.
- Sündmused (*events*) ja protsessid (*processes*) -- sündmused on samuti objektid ja nagu kõiki objekte võib neid ka kategoriseerida. Sündmused toimuvad mingil ajal ja mingis kohas. Protsessid on sündmused, mis on pidevad ja homogeensed.
  Siiani vaatlesime diskreetseid sündmusi (*discrete events*), milledel on struktuur. Sündmused, mille iga alamintervall on sama sündmus, kuuluvad vedelate sündmuste (*liquid event*) või protsesside (*process*) kategooriasse. Protsessid võivad kirjeldada nii pidevat muutust kui ka pidevat muutumatust e. olekut (*state*).

- Füüsilised objektid (*physical objects*) -- olles olemas nii ruumis kui ajas on füüsilised objektid sarnased sündmustele.
  Sündmusarvutus võimaldab objektidel omada erinevatel aegadel erinevaid omadusi.
- Ained/materjalid (*substances*) -- on olemas teatav osa reaalsusest, mida ei saa eristatavateks objektideks jagada, seda nimetame üldnimega aine (*stuff*). Aine jagamisel osadeks saame samuti aine (iga osa on omakorda aine). Aine ja objektide erinevus on sama, mis vedelate sündmuste ja tahkete sündmuste vahel (vedelaid sündmusi on nimetatud ka ajaaineks (*temporal substances*) ja aineid vastavalt ruumiaineteks (*spatial substance*)).
  On olemas sisemised (*intrinsic*) omadused, mis kuuluvad objekti ainele ja säilivad objekti jagamisel, mitte objektile kui tervikule ja välimised (*extrinsic*) omadused, mis kuuluvad objektile ja ei säili objekti jagamisel. Vaid sisemiste omadustega objektide klass on aine.
- Mõttelised objektid (*mental objects*) ja oletused/uskumused (*beliefs*) -- agendil on sageli vaja arutleda omaenda ning teiste uskumuste üle. Meie ontoloogias on laused otseselt esitatud ja usutud agentide poolt.
  Suhted agendi ja propositsionaalsete sündmuste/objektide vahel milesse agent usub on propositsioonilised arvamused (*propositional attitudes*). Võimalust asendada ühte termi teisega nimetatakse viiteläbipaistvuseks (*referntial transparency*), esimest-järku loogikas on iga suhe viiteläbipaistev. Suhet "usub" tuleks defineerida suhtena, mille teine argument on läbipaistmatu (*opaque*) -- selles argumendis pole võimalik asendada ühte termi teisega ilma, et mõte muutuks. Mõtteliste objektide süntaksi teoorias (*syntactic theory*) esitatakse mõttelised objektid esituskeele märgijadadena.

Teadmispõhiste süsteemide ehitamisel on eelis programmeerimise ees: teadmisinsener peab keskenduma vaid sellele, mis on tõene valdkonna kohta, selle asemel et lahendada probleeme ja nende lahendusprotsesse kodeerida. Samu teadmisi võib sageli kasutada erineval viisil. Teadmiste silumine on lihtsam, kui programmide silumine (kuna teadmistebaasi laused ei sõltu sellisel määral kontekstist kui programmeerimiskeele laused -- nad on deklaratiivsed ja isesisalduvad (*self-contained*)).

Teadmuspõhistes süsteemides kasutatavad loogilise arutluse süsteemid:

- Loogilise programmeerimise süsteemid ja teoreemide tõestajad (*theorem provers*)
- Produktsioonisüsteemid (*production systems*)
- Semantilised võrgud (*semantic networks*)
- Deskriptiivne loogika (*descriptive logics*)

**Loogiline tegutsemine**

Olles vaadelnud probleeme lahendavaid agente, mis on võimelised planeerima oma tegevusi arutledes tegevuste jadade tagajärgede üle ja näidanud, et teadmispõhised agendid on võimelised valima oma tegevusi põhinedes jooksva oleku ning tegevuste tagajärgede ilmutatud ja loogilisel esitusel. Kui need kaks asja ühendada saame planeerivad agendid. Kõige üldisemal tasemel on planeerimine sama mis probleemide lahendamine, seda võib vaadelda kui lahenduse otsimist olukordade ruumi asemel plaanide ruumis.

Planeerimisalgoritme võib vaadelda kui eriotstarbelisi teoreemide tõestajaid, mis arutlevad kasutades tegevusi kirjeldavaid aksioome. Keerukate probleemide korral tekib vajadus jagada need alamsihtideks (*subgoals*), millede lahendusi saab kombineerida terve probleemi lahenduseks.

Planeerivad agendid kasutavad ette vaatamist (*lookahead*), et leida tegevusi, mis aitavad sihti saavutada. Planeerivad agendid erinevad probleeme lahendavatest agentidest olekute, tegevuste, sihtide ja plaanide paindlikuma esituse poolest.

Keerukate valdkondade puhul on teostamatu (*not feasible*) otsida üle kogu olukordade ruumi. Selle asemel otsitakse plaanide ruumis, alustades minimaalsest plaanist ja laiendades seda kuni lahendus on leitud (see on effektiivne probleemide puhul, mille alamplaanid ei mõjuta teineteist).

Hiliseima sidumise põhimõte (*principle of least commitment*) väidab, et planeerija (või suvaline otsingualgoritm) peaks vältima otsuseid enne kui nende tegemiseks on head põhjendused. Osalise järjestuse kitsendused (*partial-ordering constraints*) ja väärtustamata (*uninstantiated*) muutujad võimaldavad hiliseima sidumise põhimõtet järgida.

Põhjuslikud seosed (*causal link*) on kasulik andmestruktuur sammude eesmärgi salvestamiseks. Iga põhjuslik seos kehtestab kaitstud intervalli, mille vältel tingimust (*condition*) ei või mingi teise sammu poolt kustutada. Põhjuslikud seosed võimaldavad plaani osas lahendamatuid vastuolusid leida ja seega vähendavad kasutut otsimist.

Hierarhiline dekompositsioon (*hierarchical decomposition*) lubab mitte-primitiivseid operaatoreid, mida saab dekomponeerida primitiivsemateks sammudeks, plaani lülitada. Hierarhiline dekompositsioon on effektiivseim, kui tema abil saab kärpida (*prune*) otsinguruumi. Kärpimine õnnestub, kui kehtib kas alaspidise lahendi omadus (*downward solution property*) (s.t. iga abstraktse lahendi võib dekomponeerida primitiivseks lahendiks) või ülespidise lahendi omadus (*upward solution property*) (s.t. vastuolulised abstraktsed plaanid ei oma primitiivseid lahendeid).

Paljud tegevused kulutavad ressursse nagu raha, kütus või toore. Neid on mugav käsitleda numbriliste mõõtudena. Tegevused võivad ressursse luua ja kulutada ja poolikuid plaane on kerge ning effektiivne kontrollida ressurssidele seaotud piirangute suhtes enne edasist täpsustamist. Aeg on üks kõige tähtsamaid ressursse. Välja arvatud mõned erandid, saab aega käsitleda samade üldistatud mehhanismidega, kui muid ressursse.

Standartsed planeerimisalgoritmid eeldavad täielikku ja korrektset infot. Paljud valdkonnad rikuvad seda eeldust. Mittetäieliku info täiendamiseks vajalikule tasemele saab kasutada tajutegevusi. Tingimuslikud plaanid (*conditional plans*) sisaldavad erinevaid alamplaane erinevate kontekstide jaoks sõltuvalt juurde saadud infost.

Mittekorrektne info võib vii tegevuste ja plaanide eelduste mitterahuldatusele. Täitmise jälgimine (*execution monitoring*) leiab plaani edukaks täitmiseks vajalike eelduste rikkumised. Tegevuste jälgimine (*action monitoring*) leiab tegevused, mis ebaõnnestuvad.

Lihtne üleplaneeriv agent (*replanning agent*) kasutab täitmise jälgimist ja tükeldab (*splices*) alamplaane kui vaja. Täielikum lähenemine plaanide täitmisele sisaldab plaanide inkrementaalset muutmist vastavalt tingimuste muutumisele keskkonnas.

**Ebakindel teadmine ja arutlemine**

Eelnevad osad katsid loogiliste agentide uurimise ja ehitamise. Nende puhul kasutati esimest-järku loogikat faktide esitamiseks.

Keeruliste dünaamiliste või mitte-täielikult tajutavatate keskkondade puhul on ebakindlus vältimatu. See tähendab, et paljud deduktiivse tuletuse (*deductive inference*) lihtsustused ei kehti enam.

Et toime tulla ebakindlate teadmistega kasutatakse tõenäosusteoorial põhinevaid vahendeid. Kuna tegevused ei taga enam kindlalt tulemusi, peavad agendid kasutama kasulikuse teooriat sihtide soovitavuse ja nende saavutamise tõenäosuse võrdlemiseks. Tõenäosused väljendavad agendi võimetust jõuda kindla otsuseni lause tõeväärtuse suhtes ja võtavad kokku agendi oletused/uskumused (*beliefs*). Tõenäosuse aksioomid määravad kitsendused tõenäosuste väidetele omistamisele. Kui agent rikub neid aksioome, käitub mõnedes olukordades irratsionaalselt.

Oletusvõrgud (*belief networks*) on vahendiks ebakindlate teadmiste esitamisel ja nende kasutamisel arutlustes.Sõlmede vahelised ühendused väljendavad valdkonna kvalitatiivseid aspekte ja tingitud tõenäosuse tabelid väljendavad valdkonna kvantitatiivseid aspekte. Oletusvõrgud võivad arutleda põhjuslikult (*causally*), diagnoosivalt (*diagnostically*), kombineeritult või põhjuslikuste vahel (*intercausally*).

Et väiksema arvutuskuluga saada tõeliste tõenäosustega hinnag, on võimalik kasutada ka lähendamistehnikaid (*approximation techniques*), kaasaarvatud stohhastilist simuleerimist (*stochastic simulation*).

Keerukate, ebakindlatel teadmistel põhinevate, otsuste tegemisel, kus tegevuste kasulikus agendile sõltub otsuste jadast, tagastavad planeerimisalgoritmid mitte tegevuste jada vaid poliitika (*policy*), mis on hulk reegleid, mis seovad iga oleku jaoks olukorrad tegevustega.

**Õppimine**

Seni eeldati, et kogu "intelligentsus" on agenti sisse ehitatud. Kui agendi disaineril pole täielike teadmisi keskkonna kohta on õppimine agendi ainus võimalus saada teada seda, mida on vaja edukaks tegutsemiseks keskkonnas. Tajutut peab õppiv agent kasutama mitte ainult tegutsemiseks vaid ka oma tegutsemisvõime parandamiseks.

Õppivas agendis võime eristada õppivat osa (vastutab edukuse parandamise eest), tegutsevat osa (vastutab tegevuste valiku eest -- siiani vaadeldud agentide ainus element), hindav osa (teavitab õppivat osa agendi tegutsemise edukusest) ja probleemide generaator (vastutab selliste tegevuste, mis annavad uut informatsiooni e. uurimistegevuste genereerimise eest). Õppimine, kus agent saab tagasiside kaudu teada, milline oleks pidanud olema õige tegevus, nimetatakse juhendatud õppimiseks (*supervised learning*). Õppimine, kus agent saab teada tagasiside kaudu teada, milline hinnag tema tegevusele anti (aga mitte õiget tegevust), nimetatakse kinnistavaks õppimiseks (*reinforcement learning*). Õppimine, kus agent ei saa mingit tagasisidet tegevuste õigsuse kohta nimetatakse juhendamata õppimiseks (*unsupervised learning*).

Agendi üldine disain määrab ära info, mida peab õppima. Kui agendi tegutsev osa põhineb otsustuspuudele, võib õppiv osa juhendatud õppimise korral tuletada otsustuspuid näidetest. Seda nimetatakse induktiivseks õppimiseks ja sealjuures kasutatakse Ockham'i habemenoa (*Ockham's razor*) nimelist põhimõtet, mis soovitab valida lihtsaima hüpoteesi, mis rahuldab kõiki vaatlusi.

Loogiliste teooriate õppimiseks on kaks varianti: parima jooksva hüpoteesi (*current-best-hypothesis*) meetod, mis säilitab ja muudab ühte hüpoteesi, ja versioonide ruumi (*version space*) meetod, mis säilitab kõiki kooskõlas olevaid hüpoteese. Mõlemaid mõjutab näidetes olev müra.

Närvivõrkude korral tehakse vahet kahte tüüpi võrkude vahel: päripidiste (*feed-forward*), milles on kõik ühendused samasuunalised (kihilistes päripidistes võrkudes on kõik ühendused külgnevate kihtide vahelised) ja korduvate (*recurrent*) vahel, milles ühenduste topoloogia on vaba. Korduvad võrgud on sümmeetriliste kaaludega kahesuunaliste ühendustega Hopfield'i võrgud, mille kõik elemendid on kas sisend- või väljundelemendid ja aktivatsioonifunktsioon on märgifunktsioon ja sümmeetriliste kaaludega ja varjatud elementidega (pole ei sisend- ega väljundelemendid) Boltzmann'i masinad, mis kasutavad stohhastilist aktivatsioonifunktsiooni. Hopfield'i võrgud töötavad assotsiatiivse mäluna ja kui võrgus on N elementi on Hopfield'i võrk võimeline salvestama 0.138N näidet. Närvivõrgu struktuur mõjutab oluliselt tema õppimisvõimet. Päripidine närvivõrk, milles on kaks varjatud taset võib õppida realiseerima suvalist funktsiooni. Hea närvivõrgu struktuuri leidmine on siiani uurimisprobleemiks, mille ühe lahendusena on pakutud geneetilist algoritmi.

Mitmetasemeliste närvivõrkude korral kasutatakse õppimiseks tagasilevi (*back-propagation*) algoritmi, kus vaadeldud viga levitatakse väljunditest alates kiht-kihilt kuni sisendkihini. Tagasilevi algoritmi võib vaadelda, kui gradientlaskumist elementide vaheliste ühenduste kaalude ruumis minimiseerides väljundi viga (kasutades vea pinna (*error surface*) gradienti).

Kinnistavat õppimise korral on kaks peamist vaadeldud disaini mudelipõhine, mis kasutab mudelit M ja kasufunktsiooni (*utility function*) U ning mudelivaba lähenemine, mis kasutab tegevuse väärtuse funktsiooni Q. Oleku kasulikus on praeguse hetke ja lõpetamise vahel saavutatud heade tulemuste oodatud summa.

Kasulikuseid saab õppida kolmel moel:

Vähimkeskruutude meetodil (*LMS -- least mean square*), mis kasutab antud oleku puhul vaadeldavate heade tulemuste summat kui otsest tõendit kasulikuse õppimiseks. Meetod kasutab mudelit vaid tegevuste valikuks.

Adaptiivne dünaamiline programmeerimine (*ADP -- adaptive dynamic programming*), mis kasutab väärtuse või poliitika (*policy*) iteratsiooni algoritmi, et mudeli lähenduse alusel arvutada oleku täpsed kasulikused. Meetod kasutab optimaalselt keskkonna struktuurist põhjustet olekute kasulikustel määratud lokaalseid kitsendusi.

Ajaline erinavus (*TD -- temporal difference*), mis uuendab kasulikuse hinnaguid vastavalt järgnevatele olekutele ja mida võib vaadelda adaptiivse dünaamilise programmeerimise lähendusena, mis ei vaja õppimisprotsessi jaoks mudelit. Mudeli kasutamine pseudokogemuste genereerimiseks võib aga õppimist kiirendada.

Tegevuse väärtuse funktsioonid või Q-funktsioonid on õpitavad adaptiivse dünaamilise programmeerimise või ajalise erinevuse meetodiga. Viimase puhul ei vaja õppimine ega tegevuste valik mudelit, mis lihtsustab oluliselt õppimist aga vähendab võimet õppida keerukates keskkondades.

Kui õppiv agent on vastutav tegevuste valiku eest, kui ta õpib, peab ta kaaluma nende tegevuste hinnagulist väärtust võrreldes võimaluseg õppida uut ja kasuliku infot. Uurimisprobleemi (*exploration problem*) täpne lahendus on teostamatu (*infeasible*) aga lihtsate heuristikate abil saab saavutada häid tulemusi.

Geneetilised algoritmid saavutavad kinnistava õppimise kasutades kinnistamist (*reinforcement*) suurendamaks edukate funktsioonide arvu programmide populatsioonis. Üldistamine saavutatakse geneetiliste algoritmide puhul mutatsioonide ja ristamise teel.

Taustteadmised (*background knowledge*) võimaldavad õppida palju kiiremini, kui puhas induktsioon. Teadmispõhise induktiivse õppimise (*knowledge-based inductive learning*) korral kombineeritakse taustteadmised ja uus hüpotees, et näiteid seletada. Näitepõhine õppimine (*explanation-based learning*) üldistab näiteid neid seletades ja seletust üldistades.

**Suhtlemine, tajumine ja tegutsemine**

Agentide rühm võib olla edukam nii individuaalselt kui ka kollektiivselt, kui nad suheldes edastavad oma arvamused ja sihid teistele. Suhtlemine on üldiselt tahtlik informatsiooni vahetamine, mis toimub kokkulepitud süsteemi kuuluvate märkide tekitamisel ja tajumisel.

On olemas kaks suhtlusmudelit: sõnumipõhine (*encoded message*), kus sõnumi mõte sõltub vaid sõnumist ja situatsiooniline (*situated language*), kus sõnumi mõte sõltub ka olukorrast, milles sõnum eksisteerib. Telepaatilise (*telepathic*) suhtlusega on tegemist, kui agentidel on sama sisemine teadmiste esitamise keel ja üks agent võib anda edasi teadmisi oma teadmistebaasist kujul, mida teine agent saab otse lisada oma teadmistebaasi. Tavaliselt puudub võimalus omada samat sisemist teadmiste esitamise keelt ja selletõttu on vaja suhelda läbi (suhtlus)keele, mis võib erineda agentide sisemistest teadmiste esitamise keeltest.

Taju saab alguse anduritest (*sensors*). Andurid seadmed, mis võivad muuta agendi olekut keskkonna oleku muutumisel. Andurite poolt antud informatsiooni interpreteerimise puhul tuleb nendest tuletada keskkonna muudatus, kahjuks pole selline tuletus ühene ja tulemuse saamiseks on vaja kasutada taustteadmisi.

Robot defineeritakse, kui aktiivne kunstlik agent, mille keskkond on füüsiline maailm. Põhiliselt vaadeldakse autonoomseid roboteid. Effektorid on seadmed, misroboti juhtimise all mõjutavad keskkonda. Effektoreid kasutatakse kahte moodi: muutmaks roboti asukohta ja asendit keskkonnas (*locomotion*) ning muutmaks teiste objektide asukohta ja asendit keskkonnas (*manipulation*). Kui roboti kõik vabadusastmed on juhitavad, on tegu holonoomilise (*holonomic*) robotiga.

**Kirjandus**

S. Russell, P. Norvig, **Artificial Intelligence**: A Modern Approach, 1995, pp. 932.