Mesterséges Intelligencia (Artificial Intelligence)

Bevezetés (ágens típusok, környezet tulajdonságai)

Ágens: Környezetébe ágyazott (érzékelések, beavatkozások) autonóm rendszer (minimum válasz).
[Bármi ami az **érzékelői** segítségével **érzékeli** a környezetét és **beavatkozó szervei** segítségével **megváltoztatja** azt.]
A beágyazottság, a folyamatos kölcsönhatás a környezettel, folyamatos működés (belső állapot karbantartás).

Racionális ágens: a helyes dolgot teszi.

A következő négy dolgon múlik az, hogy egy adott pillanatban mi a helyes (racionális):
- A siker fokát mérő **teljesítmény mérőszám**.
- Minden, amit az ágens eddig megfigyelt. Ezt teljes észlelési történetnek, avagy **észlelési sorozatnak** nevezzük.
- Amit az ágens a **környezetéről tud.**
- A **cselekvések**, amiket az ágens képes végrehajtani.

Ideális racionális ágens: *minden egyes észlelési sorozathoz a bennük található tények és a beépített tudása alapján minden elvárt dolgot megtesz a teljesítmény mérőszám maximalizálásáért.*

Egy rendszer annak mértékéig autonóm, amennyire a viselkedését saját tapasztalatai határozzák meg.

Tábla-vezérelt-ágens: nem elég jó, mert nem következtet.

Ágens-típusok:

- **Egyszerű reflexszerű ágens**
  feltétel-cselekvés szabályok mindig ugyanazok; mindig az aktuális állapotot vizsgálja.
  ha *az-előző-autó-fékez* **akkor** *kezdj-fékezni*
- **Ágensek, melyek nyomon követik a világot**
  Az egyszerű reflexszerű ágens csak akkor fog működni, ha a helyes döntés meghozható az aktuális észlelés alapján. Különben a vezetőnknek nyilván kell tartania valamiféle **belső állapotot** a végrehajtandó cselekvés kiválasztásához. Megoldás: az ágensnek fenn kell tartania valamiféle belső állapot információt ahhoz, hogy megkülönböztesse a világ azonos észlelési bemenetet generáló, de lényegében különböző állapotait.
- **Cél-orientált ágensek**
  A környezet jelenlegi állapotának ismerete nem mindig elég annak eldöntéséhez, hogy mit tegyünk.
  A jelenlegi állapot leírása mellett az ágensnek valamiféle **cél** információval is rendelkeznie kell, amely leírja a kívánatos helyzeteket.
  Ágens program: összevetheti a lehetséges cselekvések eredményeiről szóló információkkal annak érdekében, hogy a céljához vezető cselekvést meghatározza.
  Ez néha egyszerű, máskor sokkal trükkösebb: **keresés** és **tervkészítés**.
- **Hasznosság-orientált ágensek**
  Egy általánosabb teljesítmény mérték: a világ állapotainak az összehasonlítása, milyen boldoggá tennék az ágenst, ha elérné azokat. „Boldogság” – tudományosan „előnyösebb” egy másikhoz képest, ha magasabb a **hasznossága** az ágens számára.

Környezetek: - Hozzáférhetőség (érzékelhető minden aspektusa melyek a cselekvés kiválasztásához szükségesek),
- Determinisztikusság (a környezet köv. állapotát a jelenlegi állapota és az ágens által választott cselekvések egyértelműen meghat.),
- Epizódszerűség (ágens tapasztalata - epizódokra: ágens észlelései+cselekvései - bontható) ,
- Statikusság (dinamikus – ha ameddig az ágens dönt a környezet megváltozhat),
? - Diszkrét/folytonos elérhetőség (diszkrét, ha létezik az észlelések, cselekvések elkülönülő, világosan definiált véges elemű halmaza)

Problémamegoldás kereséssel

Célvezérelt ágens egy típusa: cselekvés sorozatokat keres, amelyek kívánt állapotokba vezetnek. (Az ágens egy **célt** tud kitűzni maga elé és megpróbálja azt elérni.)
Cél megfogalmazás: a pillanatnyi helyzet alapján.
cél: a világ állapotainak egy olyan halmaza, amelyekben a cél teljesül.
cselekvések: hatására a világ állapotai közötti állapotátmenetek mennek végbe,
az ágensnek nyilvánvalóan azt kell megkeresnie,
hogy mely cselekvések juttatják el egy célállapotba.

Problémák:

Négy lényegileg eltérő probléma típus: egy-állapotú problémák,
több-állapotú problémák,
eshetőségi problémák és
felfedezési problémák.

1. Egy-állapotú probléma

ágens érzékelői: pontosan tudja, melyik állapotban van
(vagyis a világ elérhető a számára)
pontosan tudja, hogy a cselekvése mit eredményez.

2. Több-állapotú probléma

ágens: az ágens ismeri az összes cselekvésének a hatását,
de csak korlátozott mértékben fér hozzá a világ állapotához.
(előfordulhat, hogy egyetlen érzékelővel sem rendelkezik)

3. Eshetőségi probléma

Néha **a tudatlanság megakadályozza** az ágenst egy garantált megoldás megtalálásában.
Nem létezik rögzített cselekvés sorozat, amely ebben az esetben garantálná a megoldást.

4. Felfedezési probléma

Az ágensnek semmilyen információja sincs a cselekvései hatásáról. (térkép nélkül eltévedünk egy idegen országban).
Az ágensnek tehát *kísérleteznie* kell.
Ez a legnehezebb feladat, amivel egy intelligens ágens találkozhat.

Jól definiált probléma:

- A **problématér** (a kezdeti, a közbülsõ és a célállapotok közös reprezentációja).
- Az ágens rendelkezésére álló **lehetséges cselekvések halmaza**
  operátor: egy cselekvés leírása, mely megadja, hogy a cselekvés egy adott állapotban történő alkalmazásának hatására az ágens mely állapotba kerül.
- **Irányító heurisztikák és a pálya cselekvési/számítási költségei.**

Keresés hatékonyság mérése:

1. egyáltalán talál-e megoldást. (teljesség)
2. a megtalált megoldás jó megoldás-e (egy alacsony út költségű megoldás-e)? (legjobb-optimalitás)
3. a megoldás megtalálásához szükséges idő és a memória (idő- és tárigény)

kapcsolódó **keresési költség**

A keresés **összköltsége**: az útköltség és a keresési költség összege.

Keresés: új állapotok **generálása -** az állapot **kifejtése**.
Keresési stratégia: ez határozza meg a kifejtendő állapot kiválasztását.

Csomópontok:

csomópont öt komponensből álló adatszerkezet:
- a csomóponthoz tartozó állapottér állapot
- a keresési fa azon csomópontja, amely ezen csomópontot generálta (**szülő csomópont**)
- a csomópontot generáló operátor
- a gyökér csomóponttól eddig a csomópontig vezető út csomópontjainak a száma (a csomópont **mélysége**)
- a kiinduló állapottól a csomópontig számított út költsége.

(a csomópont nem az állapot!!!
a csomópont egy adatszerkezet, egy állapot a világ egy konfigurációja.
a csomópont rendelkezik mélységgel és szülővel, míg az állapot nem).

Keresési stratégiák

Négy kritérium:

- **teljesség:** a stratégia garantáltan megtalálja a megoldást, amennyiben létezik
- **időigény:** mennyi ideig tart egy megoldás megtalálása
- **tárigény:** a keresés elvégzéséhez mennyi memóriára van szükség
- **optimalitás:** a stratégia megtalálja a legjobb minőségű megoldást, amikor több különböző megoldás létezik

Vak keresési módszerek

Semmilyen információnk nincs az aktuális állapotból a célállapotba vezető út lépésszámáról vagy út költségéről. Ezen keresési algoritmusok csak annyira képesek, hogy meg tudják különböztetni a célállapotokat a nem célállapotoktól.

Keresések:

szélességi keresés (teljes; optimális – ha az útköltség a csomópont mélységének nem csökkenő függvénye)
- először a gyökér csomópontot fejti ki, majd a gyökérből generált csomópontokat fejti ki, majd az ezekből generáltakat.
- a legsekélyebben található megoldást találja meg először.
- legjobb / legrosszabb tulajdonság: teljesség/ exponenciális komplexitású

egyenletes költségű keresés
- a szélességi keresés kiterjesztése
- mindig a legkisebb útköltségű csomópontot fejti ki először, nem pedig a legkisebb mélységűt.

mélységi keresés
- mindig a fa legmélyebben fekvő csomópontját fejti ki először
- ha zsákutcába jut, akkor kifejti az eggyel magasabban szinten lévő csomópontot
- legjobb / legrosszabb tulajdonság: lineáris komplexitású nem teljes

! lineáris tárkomplexitású !

mélységkorlátozott keresés
- a mélységi keresés kiterjesztése (javítása)
- korlát a max. mélységre (ezzel kiküszöböli a mélységi keresés gyengeségét-keresés nagy vagy végtelenül mély keresési fán)

! lineáris tárkomplexitású !

iteratívan mélyülő keresés (mélységkorlátozott keresések egymás után, egyre növekvő mélységkorláttal)
! lineáris tárkomplexitású !

kétirányú keresés
- a keresést egyszerre mind a kiinduló-, mind a célállapotból elindítom előre ill. hátrafelé.
- jó ha reverzibilisek az operátorok, ekkor könnyű meghatározni az elődcsomópontokat.

A keresési stratégiák összehasonlítása:

Kritérium	Szélességi	Egyenletes költségű	Mélységi	Mélység-korlátozott	Iteratívan mélyülő	Kétirányú (amennyiben alkalmazható)
Idő igény	B^d	b^d	b^m	B^l	b^d	$b^{d/2}$
Tár igény	B^d	b^d	bm	Bl	bd	$b^{d/2}$
Optimális?	Igen	Igen	Nem	Nem	Igen	Igen
Teljes?	Igen	Igen	Nem	Igen, ha $l \geq d$	Igen	Igen

A keresési stratégiák értékelése. d a megoldás mélysége, m a keresési fa maximális mélysége, l a mélység korlát b - elágazási tényező - egy állapot elágazási tényezője, az állapot kifejtéséből keletkező új állapotok száma !

Informált keresési módszerek

Legjobbat-először (Best-First) keresés

Következő kifejtendő csomópont? ⇒ tudás ⇒ **kiértékelő függvény**
- egy csomópont kifejtésének szükségességét leíró szám,
- sorrendezés, a legjobb értékkel rendelkező csomópontot fejti ki legelőször

Heurisztikus függvény: $h(n)$ = az ***n* csomópont állapotából** egy cél-állapotba vezető legolcsóbb út becsült költsége (ezt minimalizálja).

A becsült költség minimalizálása: a Mohó keresés (nem teljes, és nem optimális)

Mohó keresés - az a legjobbat először keresés, amelyik a h heurisztikus függvényt használja a következő

kifejtendő csomópont kiválasztására.

A keresés* (**elfogadható heurisztika -** s*oha nem becsüli felül* a cél eléréséhez szükséges költséget)

adott h függvény,
(optimista, úgy gondolja, hogy a probléma megoldása kisebb költséggel jár, mint amekkora költséget a megoldás valójában igényel).
amennyiben h elfogadható, akkor f(n) soha sem becsüli túl az n csomóponton keresztül vezető legjobb megoldás valódi költségét
f függvényt alkalmazó legjobbat-először keresés, amelyben *h* elfogadható: **A* keresés**

f(n) – a legolcsóbb, az n csomóponton keresztül vezető megoldási út költségének becslője.
g(n) – az n csomópontig megtett út költsége.

maximális-út egyenlőség: $f(n') = max(f(n),g(n')+h(n'))$.

A heurisztikus függvény minősítése: b^* - **effektív elágazási tényező:** az A* által kifejtett összes csomópont száma egy adott problémára N, a megoldás mélysége d, akkor b^* annak a d mélységű kiegyensúlyozott fának az elágazási tényezőjével egyezik meg, amely N csomópontot tartalmazna:

$$N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d.$$

Minden n csomópontra $h_2(n) \geq h_1(n)$, h_2 **dominálja** h_1–et

Az olyan problémát, amelyben az operátorokra kevesebb megkötést teszünk, mint az eredeti problémában, **relaxált problémának** nevezzük.
Nagyon gyakran teljesül, hogy a relaxált probléma pontos megoldásának költsége jó heurisztikus függvénynek (h) bizonyul az eredeti problémára.

EMA (Egyszerűsített memóriakorlátozott A* keresés)*

Az EMA* algoritmus egyszerű, legalábbis vázlatos szinten. Amikor egy követő csomópontot kell legenerálnia és már nem áll rendelkezésére felhasználható memória, akkor helyet kell csinálnia a sorban. Ehhez egy csomópontot törölni kell a sorból. Az ily módon törölt csomópontokat **elfelejtett csomópontoknak** nevezzük. Az algoritmus a nagy f költségű csomópomtokat törli a sorból.

Iteratívan javító algoritmusok

Az alapötlet ennél a megközelítésnél az, hogy egy teljes (de nem jó) konfigurációból indulunk ki és olyan módosításokat hajtunk végre, ami javítja a konfiguráció minőségét.

Az iteratívan javító algoritmusok két nagy csoportba oszthatók. A **hegymászó** (vagy **grádiens módszer**, ha a kiértékelő függvényt költségnek nem pedig minőségnek tekintjük) algoritmusok mindig olyan változtatásokat próbálnak meg végrehajtani, amik javítanak az aktuális állapoton. A **szimulált lehűtési** algoritmusok néha megengednek olyan lépéseket is, amelynek hatására (legalábbis átmenetileg) rosszabb állapotba kerülünk.

Hegymászó keresés

A hegymászó keresés egyszerűen csak egy ciklus, ami mindig javuló értékek felé lép. Az algoritmus nem tart nyilván keresési fát, ezért a csomópontot leíró adatszerkezetnek csak az állapotot és a kiértékelését kell nyilvántartania. Egy fontos finomítás, hogy amikor egynél több legjobb követő csomópont létezik, az algoritmus közülük véletlenszerűen bármelyiket kiválaszthatja. Ez a megközelítés három ismert problémával küszködik. A lokális maximumban megragad, ezen a ”véletlen újraindítású” verziója segít !

Szimulált lehűtés

Ahelyett, hogy ismét véletlenszerűen újraindítanánk a keresést, amikor az algoritmus egy lokális maximumban ragad, megengedhetjük a keresési algoritmusnak, hogy néhány lefelé vezető lépést tegyen, hogy elmeneküljön a lokális maximumból. Durván ez az alapgondolata a **szimulált lehűtés**nek. A szimulált lehűtés legbelső ciklusa nagyon hasonlít a hegymászáshoz. A *legjobb* lépés megtétele helyett azonban egy *véletlen* lépést tesz. Ha a lépés javítja a helyzetet, akkor az mindig végrehajtásra kerül. Ellenkező esetben az algoritmus a lépést csak valamilyen, 1-nél kisebb valószínűséggel teszi meg. A valószínűség exponenciálisan csökken a lépés "rosszaságával" – azzal a delta E mennyiséggel, amivel a kiértékelő függvény értéke romlott.

A logikusan gondolkozó ágens - a tudásbázisú ágens
Reprezentáció, következtetés és logika

tudásbázis -a világot leíró tények egy halmaza
mondatok - tudás darabkák, majd logikában "igazi" mondatok lesznek
tudás reprezentációs nyelv

tudásreprezentáció - a reprezentációs nyelv két aspektusa:
- a nyelv **szintaktikája:** a lehetséges konfigurációja az összes létrehozható mondatnak.
- a **szemantika** meghatározza a világ tényeit, amikre a mondatok vonatkoznak.
A szemantikával minden mondat állít valamit a világról. Szemantikával azt is mondatjuk, hogy ha a mondat az ágens valamely fizikai konfiguráció által reprezentált, akkor az ágens **hiszi** a hozzátartozó mondatot.

Új mondatokat akarunk létrehozni, amelyek szükségszerűen igazak, mivel a régi mondatok is igazak. Ezt a mondatok közötti kapcsolatot **vonzat**nak nevezzük, ami tükrözi azt a kapcsolatot, hogy egy tény következik a másikból.
A vonzat reláció az tudásbázis *TB* és az α mondat között: *TB* vonzata α : $TB \models \alpha$

Következtetési eljárás (két dolgot tehet):
ha adott egy *TB*
- létrehozhat új α mondatokat, amelyek vonzata a TB-nak.
- ha adott egy TB és egy másik α mondat, akkor megállapíthatja hogy α vonzata-e a TB-nak.

Azt a következtetési eljárást, amely csak olyan mondatokat hoz létre, amelyek vonzatai más mondatoknak, **igazságtartó**nak nevezzük.
Egy igazságtartó következtetési eljárás operációinak sorozatát **bizonyítás**nak nevezzük.

Egy következtetési eljárás teljes, ha minden vonzat mondathoz képes találni egy bizonyítást.

Az igazság függ mind a mondat interpretációjától, mind a világ aktuális állapotától.

Egy mondat érvényes avagy természetszerűleg igaz akkor és csakis akkor, ha minden világban minden lehetséges interpretációja igaz, függetlenül attól, hogy mit szándékozott jelenteni és függetlenül az univerzum leírt dolgainak állásától.

Egy mondat **kielégíthető** akkor és csakis akkor, ha létezik valamely interpretációja, amely valamely világban igaz.

Egy logika - teljes, ha minden igaz állítás bebizonyítható benne.
- **eldönthető,** ha minden állításának mind az igaz, mind a hamis voltát algoritmikusan ki lehet deríteni.

Az egyszerű logika eldönthető (indok: pl. az igazság tábla módszerével). Az elsőrendű logika csak félig eldönthető.

Összefoglalva, elmondható, hogy **a logika a következőkből áll**:

- **Egy formális rendszer** a dolgok állapotainak leírásra, amely tartalmazza:
  - a nyelv **szintaktikáját**, amely leírja, hogy hogyan készítsünk mondatokat
  - a nyelv **szemantikáját,** amely kifejezi a mondatoknak a dolgok állapotával levő kapcsolatát meghatározó szisztematikus kényszereket
- **A bizonyítás elmélet** – szabályok egy halmaza, amely mondatok egy halmaza által maga után vont vonzat kikövetkeztetésére alkalmas.

Kétféle logika: az ítélet logika és az elsőrendű logika

Az ítélet logikában szimbólumok reprezentálnak teljes tényeket; például, *D*-nek lehet egy interpretációja, hogy a „Wumpus halott”, amely lehet hogy igaz, vagy lehet, hogy nem igaz állítás.
A szimbólumokat a **kötőszavakkal** kombinálhatjuk, hogy összetettebb jelentéssel bíró mondatokat hozzunk létre.

Az **elsőrendű logika** a világ reprezentációját **objektumok** és objektumokra épülő **predikátumok** (például, objektumok tulajdonságai vagy objektumok kapcsolatai) formájában valósítja meg, használva **kötőszavakat** és **kvantorokat**, amely megengedi, hogy rögtön bármiről az univerzumban megfogalmazhassunk mondatokat.

Ítélet kalkulus (egyszerű logika)

Szimbólumok: *igaz* és *hamis* **logikai konstansok**,
ítélet szimbólumok, mint a *P* és a *Q*,
a **logikai kötőszók** a $\wedge,\vee,\Leftrightarrow$(ekvivalencia, igaz:=> és <=),$\rightarrow$,és $\neg$, és a zárójelek, ().

Precedencia sorrendje (a legmagasabbtól a legalacsonyabb fele): $\neg$, $\wedge$, $\vee$, $\rightarrow$ és $\Leftrightarrow$.

Modell - bármely világ, amelyben egy mondat igaz egy bizonyos interpretációban.

Az ítélet logika következtetési szabályai

Deduktív következtetõ lépések (jól definiált formulák kombinálási módszerei)

Modus Ponens $\frac{A,\ A \rightarrow B}{B}$
(Implikáció eliminálása)

AND eliminálása $\frac{A_1 \wedge A_2 \wedge ... \wedge A_n}{A_i}$

AND bevezetése $\frac{A_1, A_2, ..., A_n}{A_1 \wedge A_2 \wedge ... \wedge A_n}$

OR bevezetése $\frac{A_i}{A_1 \vee A_2 \vee ... \vee A_n}$

Dupla negálás eliminálása $\frac{\neg\neg A}{A}$

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

Rezolúció
$$\frac{A \vee B,\ \neg B \vee C}{A \vee C}$$

Fontos még: $(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$

Logika monotonitása: ha $TB_1 \models X$ akkor $(TB_1 \cup TB_2) \models X$
Horn klózok: mondatoknak egy hasznos osztálya, amelyre létezik polinomiális idejű következtetési eljárás.
Horn klóz: $P_1 \wedge P_2 ... \wedge ... P_n \rightarrow Q$

A következtetés igazságtábla módszere teljes.

Predikátum(szituáció) kalkulus (elsőrendű logika)

Elsőrendű logika:
- a világot **objektumok** alkotják, amelyek másik objektumoktól megkülönböztető saját azonosítókkal és **tulajdonságokkal** rendelkező dolgok.
- ezen objektumok között különböző **relációk** létezhetnek. A relációk közül néhány **függvény** – olyan reláció, ahol csak egy „értéke" van egy adott „bemenet" esetén.

termek: *BalLába*(*Apja*(*János*)), ...
Egy **term** egy objektumra vonatkozó logikai kifejezés. (Konstans szimbólumok tehát a legegyszerűbb termek).
atomi mondatok: *Bátyja*(*Apja*(*János*)), *Házas*(*Apja*(*Richárd*), *Anyja*(*János*)), ..
Tényeket fejeznek ki. Egy atomi mondatot egy predikátum szimbólum és az őt követő zárójelezett listán található termek listája alkotja. Az atomi mondatoknak lehetnek összetett termek is az argumentumai.
összetett mondatok: *Bátyja*(*Apja*(*János*)) $\rightarrow$ *Házas*(*Apja*(*Richárd*), *Anyja*(*János*))
Használhatunk **logikai összekötő szimbólumokat** a még összetettebb mondatok építésére.

kvantorok:

Tulajdonságok, amelyek objektumok egész gyűjteményeire vonatkoznak, ahelyett hogy megneveznénk minden objektumot a nevével.
Az elsőrendű logika két standard kvantort tartalmaz:

univerzális kvantor ($\forall$) $\forall x\ Macska(x) \rightarrow Emlős(x)$
egzisztenciális kvantor ($\exists$)

Az egyediség kvantor: $\exists!$

$\exists!\ x\ Király(x)$

„Létezik egy egyedi objektum *x*, amely kielégíti a *Király(x)*-et" vagy kevésbé formálisan „létezik pontosan egy király".

Következtetési lépések kibõvítése

Modus Ponens:

$$\frac{E_1 \qquad E_1 \rightarrow E_2}{E_2} \qquad \frac{P(A) \qquad \forall x\ P(x) \rightarrow Q(x)}{Q(A)}$$

(ilyen kvantifikált implikáció általában egy korábbi indukció eredménye)

Univerzális kvantor eliminálása:

$$\frac{\forall x\ P(x, A)}{P(B, A)}$$

Egzisztenciális kvantor eliminálása:

$$\frac{\exists x\ Q(x, A)}{Q(B,A)}$$

feltéve, hogy B-nek másutt nincs szerepe a tudásbázisban!

B az ún. Skolem konstans, tehát bizonyos tulajdonságokkal rendelkezõ, pl. emberszerű, de a feladatban önálló léttel nem rendelkezõ objektum.

Egzisztenciális kvantor bevezetése:

$$\frac{P(B,A)}{\exists x\ P(x, A)}$$

Rezolúció:

$$\frac{P(x, A) \vee Q(x) \qquad \neg Q(B) \vee R(x)}{P(B, A) \vee R(B)}$$

A rezolúciós stratégiák:

- egységpreferencia: először azokat a következtetéseket választjuk ki, amelyek rövid mondatokat hoznak létre.
  Pl. egységmondat($\neg$P), P $\vee$ Q => Q
- támogató halmaz: az eljárásmondatok egy halmazának kiválasztásával kezdődik, amelyet támogató halamznak nevezünk, aztán egy halmazbeli és egy nem halmazbeli elemet kombinálunk össze és az eredmény hozzáadjuk a halmazhoz
- bemeneti rezolúció: egy mondatot(illetve ennek a mindig alakitott verzióját) kombinálunk más mondatokkal.
- bennfoglalás: kizárja a keresésből azokat a mondatokat, melyek benne foglaltatnak más TB-beli mondatokban.

Teljes: **minden igaz állítás belátható** (Gödel, 1930, egzisztenciális bizonyítás)
Félig eldönthetõ: **hamis állítás hamis volta nem mutatható ki !**

Általánosított Modus Ponens:

ÉS bevezetése + Univerzális kvantor eliminálása + Modus Ponens
==> Általánosított Modus Ponens

$$\frac{P1(x1),\ P2(x2),\ ...,\ Pn(xn) \qquad P1(y1) \wedge P2(y2) \wedge ...Pn(yn) \rightarrow Q(y1,y2,...,yn)}{Q(x1,x2,...,xn)}$$

Horn klózzá alakítás:

Horn klóz: **konjunkció → egyetlen atom**

Konverzió Horn-klózokká: egzisztenciális kvantor eliminálása + AND eliminálása

Modus Ponens lépés nem képes felhasználni a tudásbázis összes állítását, vagy a tudásbázis létrehozásánál mesterséges megkötéseket kell alkalmazni!
(Modus Ponens alapú bizonyítás nem teljes, de az elsőrendű logika teljes)

Transzformáció klóz formára:

1. Implikációt eltüntetni: $A \rightarrow B = \neg A \vee B$

2. Negálást az atomi formulák szintjére áthelyezni.
   $\neg (A \vee B) = \neg A \wedge \neg B$, $\neg \forall x\, P(x) = \exists x\, \neg P(x)$

3. Egzisztenciális kvantorokat eltüntetni.
   Skolemizálás (egzisztenciális kvantorok eliminálási folyamata)
   $\exists\, x\ Owns(Nono, x) \wedge Missile(x)$ ===> $Owns(Nono, M1)$
   $Missile(M1)$
   Every person has a heart (Minden embernek van szíve).
   $\forall x\ Person(x) \rightarrow \exists\, y\ Heart(y) \wedge Has(x, y)$
   $\forall x\ Person(x) \rightarrow Heart(H1) \wedge Has(x, H1)$

   feltéve, hogy *H1* (egy fiktív szív) sehol sem szerepel a tudásbázisban

   $\forall x\ Person(x) \rightarrow Heart(f(x)) \wedge Has(x, f(x))$

   feltéve, hogy $f(x)$ (minden x-hez egy fiktív szív) sehol sem szerepel a tudásbázisban

4. Ha szükséges, a változókat átnevezni.
   $\forall x\, P(x) \vee \forall x\, Q(x)$ ----------> $\forall x\, P(x) \vee \forall y\, Q(y)$

5. Univerzális kvantorokat balra kihelyezni.
   $....\forall x....\forall y.. = \forall x \forall y\ ....x...y..$

6. Diszjunkciókat literál szintjére áthelyezni.
   $(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$
   Ami most megvan, az **a konjunktív normál forma**

7. Konjunkciókat eltüntetni.
   Bontás diszjunktív klózokra
8. Ha szükséges, a változókat átnevezni.
9. Univerzális kvantorokat elhagyni.

A szituáció kalkulus

A változások leírásának egy bizonyos módja az elsőrendű logikában. Ez úgy tekinti a világot, hogy az **szituációk** sorozatából áll, amelynek mindegyike egy „pillanat felvétel" a világ állapotáról.
Minden relációt vagy tulajdonságot amely időben változhat, a hozzátartozó predikátumhoz történő extra szituáció argumentum hozzáadása segítségével kezelünk. A szituáció argumentum mindig az utolsó és a szituáció konstansokat S_i jelöli.
A **hatás axióma** szerepe leírni, hogy egy adott cselekvésnek milyen hatása van a világ általa megváltoztatott tulajdonságára.
A **keret axiómák** azt írják le, hogy hogyan marad a világ változatlan (a változás ellenkezőjeként). Együttesen a hatás axiómák és a keret axiómák egy teljes leírását adják, hogyan fejlődik a világ az ágens cselekvéseinek hatására.

Utód-állapot axióma: Összekombináljuk a hatás axiómákat és a keret axiómákat egyetlen axiómába, amely leírja hogyan számítsuk a *Birtokol* predikátumot a következő lépésben, ha adott az értéke a pillanatnyi lépésben. Egy ilyen axióma szükséges minden egyes predikátumhoz, amely változhat az idők során. Egy utód-állapot axiómának fel kell sorolnia, minden lehetséges módját a predikátum igazzá válásának és minden módot, amikor hamissá válik

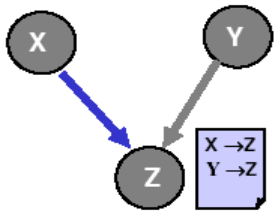
Igaz utána ⇔ [bármely cselekvés, amely igazzá tette
∨ már igaz volt és nem volt olyan cselekvés, ami hamissá tette volna]

Valószínűségi következtető rendszerek

A valószínűségi eloszlás tömör megadása -> **valószínűségi háló (belief network)**

A valószínűségi háló → egy gráf:

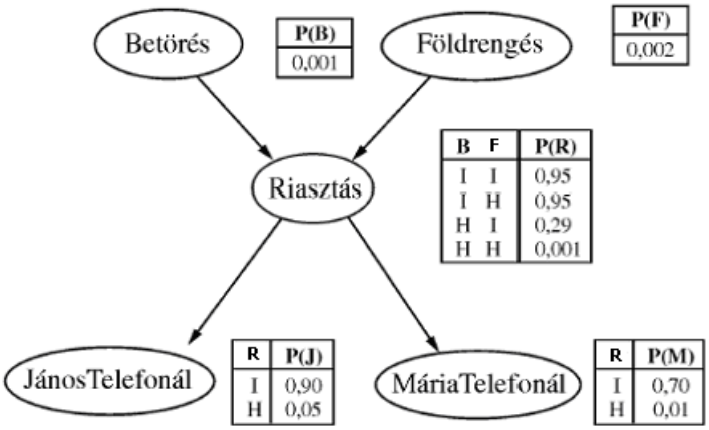
1. Csomópontok: valószínűségi változók egy halmaza
2. Csomópont párok: irányított élek halmaza:
   Az *X* csomópontot a *Z* csomóponttal összekötő nyíl → az *X-nek közvetlen befolyása* van a *Z*-ra
3. Minden csomópont: feltételes valószínűségi tábla → szülők hatása a csomópontra
4. A gráf nem tartalmaz irányított kört (= irányított, körmentes gráf = DAG).



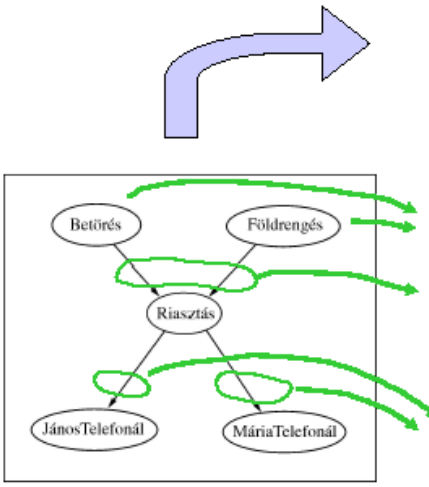
Valószínűségi változó - egy állítás a problémáról

Feltételes valószínűségi táblázat – FVT
- minden egyes csomópontra
- Egy sor a táblázatban az egyes csomóponti értékek feltételes valószínűsége az adott sorhoz tartozó szülői feltétel esetén.
- Szülői feltétel: a szülő csomópontok értékeinek egy lehetséges kombinációja (egyfajta elemi esemény).

Példa:



Általánosságban, ha a változóink binárisak, és n db bináris szülőm van, akkor 2^n-1 valószínűség adható meg. (maximum-worst case)



Együttes eloszlás: 5 változó

$= 2^5-1 = 31$

Háló:

2 x 1 a priori

+

1 x 4 szülői feltétel

+

2 x 2 szülői feltétel

$= 2 + 4 + 4 = 10$

Valószínűségi hálók építése:

- együttes valószínűségi eloszlás = feltételes valószínűségek szorzata

$$P(x_1, \ldots, x_n) = P(x_n \mid x_{n-1}, \ldots, x_1)\, P(x_{n-1}, \ldots, x_1)$$

ezek, a P(AB)=P(A|B)*P(B) alapján.

Érdemes úgy sorszámozni, hogy ez igaz legyen: (sokat lehet egyszerüsíteni)

$$P(Xi \mid Szülők(Xi), Ősök(Xi)) = P(Xi \mid Szülők(Xi))$$

Az általános eljárás egy háló fokozatos megépítésére:

1. Határozzuk meg a tárgytartományt leíró változók halmazát.
(azon eseményeket, melyek a leírásban szerepelnek és egymásra hatással vannak)
2. Határozzunk meg egy sorrendet.
3. Ameddig maradt még érintetlen változó:

a) Válasszuk a következő *Xi* változót és adjunk egy csomópontot a hálóhoz.

A helyes sorrend a csomópontok hozzáadásánál:
először az „alapvető okokat” adjuk a hálóhoz,
majd a változókat, amiket befolyásolnak,
és ezt addig folytatjuk, amíg el nem érjük
a „leveleket”, amiknek már nincs közvetlen okozati hatása
más változókra.

b) Legyen a *Szülők*(*Xi*) a csomópontok azon minimális halmaza, amik már szerepelnek a hálóban és a feltételes függetlenségtulajdonságát teljesítik.
ha minden csomópont feltételesen független a sorrendben őt megelőzőktől, feltéve, hogy a szülők értékei ismertek.
Példa:

$$P(MáriaTel. \mid JánosTel., Riasztás, Földrengés, Betörés) = P(MáriaTel. \mid Riasztás)$$

c) Definiáljuk *Xi* feltételes valószínűségi tábláját.

* Mivel mindegyik csomópontot csak korábbi csomóponthoz csatlakoztathatunk = a háló körmentes lesz.
* Egy másik fontos tulajdonsága a valószínűségi hálóknak, hogy nem tartalmaznak redundáns valószínűségi értékeket, kivéve esetleg egy bejegyzést soronként a feltételes valószínűségi táblában.

Az együttes valószínűségi eloszlásfüggvény számítása:
- minden esemény valószínűsége felbontható az FVT megfelelő elemeinek a szorzatára

Példa:

Esemény: a riasztó megszólal, de sem betörés, sem földrengés nem volt, azonban János is és Mária is telefonál:

$$\begin{aligned} P(J\ M\ R\ \neg B\ \neg F) &= P(J|\ M\ R\ \neg B\ \neg F)\ P(M\ R\ \neg B\ \neg F) \\ &= P(J|R)\ P(M|R\ \neg B\ \neg F)\ P(R\ \neg B\ \neg F) \\ &= P(J|R)\ P(M|R)\ P(R|\neg B\ \neg F)\ P(\neg B)\ P(\neg F) \\ &= .90 \times .70 \times .001 \times .999 \times .998 \\ &= .00062 \end{aligned}$$

A valószínűségi hálók négy eltérő természetű következtetésre képesek:

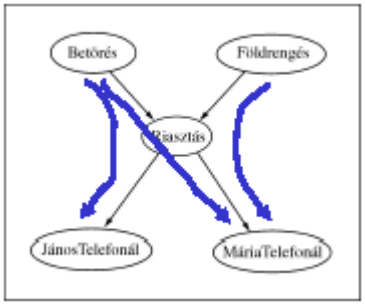
1. **Diagnosztikai következtetés** (hatásról az okra)
Ha adott a *JánosTelefonál,* akkor kiszámíthatjuk,
hogy *P*(*Betörés*|*JánosTelefonál*)=0,016.

2. **Okozati következtetés** (okról a hatásra)
Ha adott a *Betörés,* akkor kiszámíthatjuk,
hogy *P*(*JánosTelefonál*|*Betörés*)=0,67.

3. **Okok közötti következtetés** (következtetés egy közös hatás okai között)
Ha adott a *Riasztás* akkor *P*(*Betörés*| *Riasztás*)=0.376.
Ha azonban hozzávesszük azt a tényt is, hogy a *Földrengés* igaz, akkor a P(*Betörés*|*Riasztás*∧*Földrengés*) = 0.003.
Bár a betörések és a földrengések függetlenek, az egyik jelenléte a másik valószínűségét csökkenti. Ez a fajta következtetési mód a **kimagyarázás.**
4. **Kevert következtetések** (a fentiek kombinált használata).
Ha a *JánosTelefonál* okozat igaz és a *Földrengés* ok hamis, akkor *P*(*Riasztásl*|*JánosTelefonál*∧ ¬*Földrengés*)=0,03.
Ez a diagnosztikai és okozati következtetés együttes felhasználása.
Hasonlóan,
P(*Betörés*|*JánosTelefoná*∧¬*Földrengés*)=0,017.
Ez a diagnosztikai és az okok közötti következtetés kombinálása.

Egy algoritmus lekérdezések megválaszolására

- egyszeresen összekötött,
bármely két csomópont között legfeljebb egyetlen egy irányított út



- többszörösen összekötött
bármely két csomópont között több irányított út

Egyszeresen összekötött fa ,gráf (polytree) esetén létezik lineáris komplexitású algoritmus.
Többszörösen összekötött fa ,gráf esetén nem létezik zárt alakú (rekurzív) algoritmus.
(vagy megugrik a komplexitás, vagy csak közelítő módszerek vannak)

HIÁNY !!!

A bizonytalan következtetés tárgyalása

Ahogy a tudásbázishoz hozzáadott állítások
megváltoztatják az abból levonható következtetéseket,
ugyanúgy a valószínűség is megváltozik, amikor több
tény birtokába jutunk.

Minden valószínűségi kijelentésnek hivatkoznia kell azokra a
tényekre, amelyek alapján az adott valószínűség az
állításhoz lett rendelve.
Amint egy ágens új észlelések birtokába jut, ezeknek
figyelembevételével módosítja a valószínűségek becslését.

Mielőtt tények birtokába jutunk:
előzetes, a **priori** (prior), feltétel nélküli valószínűség.

Tények birtokában:
utólagos, a **posteriori**, feltételes valószínűség.

Példa.: (feltételes valószínűség számolására)

	Fogfájás	*¬Fogfájás*
Fogszuvasodás	0.04	0.06
¬Fogszuvasodás	0.01	0.89

$$P(Fogszuvasodás) = 0.06 + 0.04 = 0.11.$$

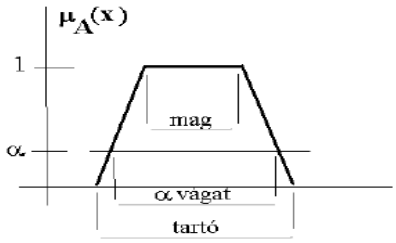
$$P(Fogsz.|Fogfájás) = \frac{P(Fogsz. \wedge Fogfájás)}{P(Fogfájás)} = \frac{0.04}{0.04 + 0.01} = 0.8$$

Fuzzy logika

Fuzzy **változó** – **egy szó** pl. ibolyák **kékek**, rózsák nem kékek, kobalt üveg nagyon kék.

Fuzzy halmazok értelmezése

fuzzy halmaz: $A = \{ (x, \mu_A(x)), x \in X, \mu_A(x) \in [0,1] \}$
X - univerzum
$\mu_A(x)$ - tagsági függvény



Természetes nyelvtől fuzzy logika felé:
- címkék (fuzzy halmazok univerzum felett)
- módosító szócskák: nagyon .., kissé .., egészen .., többé-kevésbé, ...
- negálás, logikai összekötő szavak: nem, és, vagy, ...
- fuzzy feltételes kijelentések: Sok -> Olcsó
- fuzzy algoritmusok:
  értékadás: X és Y kicsi(k) és csúnya(k)
  feltételes kifejezések: ha X nagy, akkor növeljük kicsit az Y -t
  feltétel nélküli kijelentések: csökkentsük Y -t egy kicsit

A fuzzy logika idenpotens => $A \vee A = A$
de: $\neg A \vee A \neq$ igaz, $A \wedge \neg A \neq$ hamis !

Fuzzy logikában (Zadeh) az „akkor és csakis akkor” operátor eredetileg:
$t(p \rightarrow q) = \max(1-t(p), t(q))$

Fuzzy aritmetika:

$[a,b,a',b'] + [c,d,c',d'] = [a+c,b+d,a'+c',b'+d']$

hasonlóan kivonás, szorzás, osztás

Nyelvi módosító szócskák (hedges): $X = \mu_A(..)$

nagyon	$= X^2$
kevésbé	$= 0{,}75\ X$
nagyon	$= 1{,}25\ X$
valamivel kevesebb	$= X^{1.75}$
valamivel több	$= X^{0.75}$
többé-kevésbé = kevésbé (valamivel több)	$= 0.75\ X^{0.75}$

Műveletek (alapválasztás)

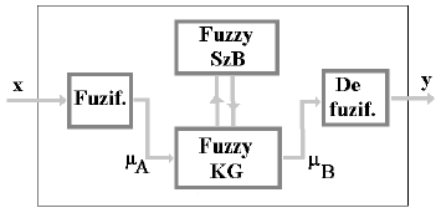
$\mu_{A \wedge B}(x, y) = \min(\mu_A(x), \mu_B(y))$

$\mu_{A \vee B}(x, y) = \max(\mu_A(x), \mu_B(y))$

$\mu_{\neg A}(x) = 1 - \mu_A(x)$

Fuzzy következtetés

Fuzzy következtető (feldolgozó) rendszer
- Fuzzy SzB - Fuzzy Szabály Bázis
- Fuzif. - Fuzifikálás
- Defuzif.- Defuzifikálás
- Fuzzy KG - Fuzzy Következtető Gép



A következtetés felépítése:
Numerikus adat -> fuzifikálás fuzzy halmazzá -> következtetés fuzzy szabályokkal ->
-> defuzifikálás numerikus adattá -> kimeneti adat

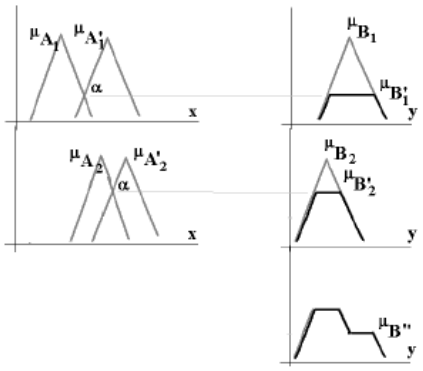
Szabályok szuperponálása:

A_1 -> B_1
A'_1

A_2 -> B_2
A'_2

B'_1, B'_2

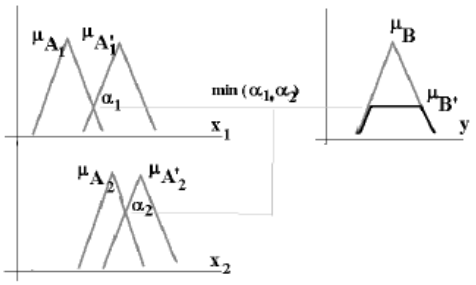
B'



Szabályfeltétel bonyolítása:

$A_1 \wedge A_2$ -> B
A'_1, A'_2

B'



Tervkészítés

A terv egy cselekvéssorozat, mely cselekvések egymásutánja.

A tervkészítés lehet - **progresszív** – a kiinduló szituáció felől próbál meg eljutni a célszituációig.
- **regresszív** – visszafelé keresés a célállapot felől a kiindulási állapot felé.

A keresés alapú problémamegoldók alapelemei/bajai:

Cselekvések leírása:
követő állapotgenerálással - a cselekvés végrehajtása a generált állapotokból látszik

Állapotok leírása: minden állapot leírása teljes.
Állapotleírások használata: **csak** a követő állapotok generálásakor, heurisztikus függvények kiértékelésekor és a célállapot vizsgálatakor.

Célok leírása: a célról csak a célállapot vizsgálatára és heurisztikus függvény kiértékelésére használt eljárások szolgáltatnak információt.

Tervek leírása: megoldás = cselekvéssorozat,
A cselekvések végrehajtási sorrendje kötött, a kezdeti állapottól indulnak (vagy a célállapotból).

Ha az ágens tudatában van annak hogy mit tartalmaz a **cél**, (ismeri a cél tartalmát képező predikátumokat) és tudja, hogy egy predikátum hatására egy célpredikátum teljesül, akkor olyan tervet fog készíteni amelyben az adott célpredikátumhoz tartozó predikátum szerepel.

Tervkészítés szituáció(predikátum) kalkulusban:
Bár elméletileg minden tervezés leírható predikátum kalkulusban,
mégis a gyakorlatban - exponenciális komplexitása
- az eredményről kapott tervről csak azt tudhatjuk biztosan, hogy a cél állapothoz vezet.
miatt nem használjuk.

A tervkészítés gyakorlatiassá tétele:

1. Korlátozzuk a probléma leírására használt nyelvet!
   - korlátozott nyelvvel kevesebb olyan megoldás adódik, amelyeket végig kell keresnünk.

2. Használjunk a megoldás keresésére speciális tervkészítő eljárást az általános célú tételbizonyító helyett!

STRIPS – Stanford Research Institute Problem Solver

A tervkészítés alapvető reprezentációi:
- Állapotok és célok reprezentációja - Függvénymentes rögzített literálok konjukciói
  Pl. Helyszín(Otthon) ∧ Van(Tej) ∧ Van(Banán) ∧ … ∧ Van(Fúró)
  De a célok tartalmazhatnak változókat is:
  Pl. Helyszín(x) ∧ Árul(x,Tej)
- Cselekvések reprezentációja – STRIPS operátorok
  Op(Cselekvés*: JobbCipőFel*
  Előfeltétel*: JobbZokniFelhúzva*,
  Következmény*: JobbCipőFelhúzva)*

Az eredmény:
A kezdeti szituáció, pl.:
Helyszín(*Otthon*)*, Út*(*Otthon, Élelmiszerbolt*)*, …*
a *Megy*(*Élelmiszerbolt*) alkalmazásával **az eredmény**
Helyszín(*Élelmiszerbolt*)*, Út*(*Otthon,Élelmiszerbolt*)*, …*

Szituációtér és tervtér

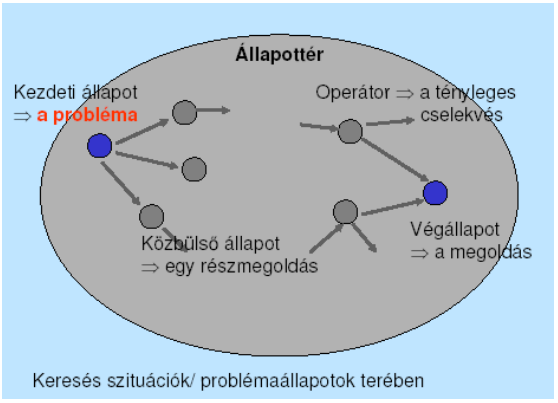
A kiinduló állapotból operátorokat alkalmazunk egyesével, egészen addig, míg az aktuális állapot nem tartalmazza a célállapot összes literálját.

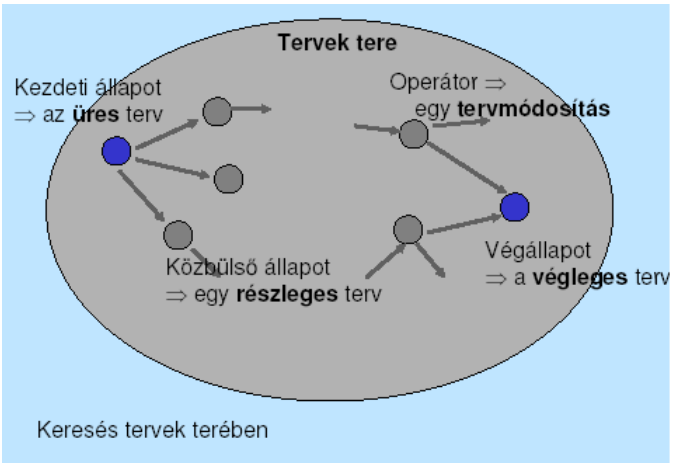
- **Szituációtérbeli tervkészítő:** a lehetséges szituációk terében keres, progresszív tervkészítő, mert a kiinduló szituáció felől próbál eljutni a cél szituációig.

  Megközelítés hátránya - rendkívül nagy elágazási tényező
  - óriási méretű keresési tér

Megoldások az elágazási tényező csökkentésére:

a) regresszív tervkészítés
b) váltsunk absztrakciót és keresési teret, menjünk oda ahol kisebb a komplexitás:

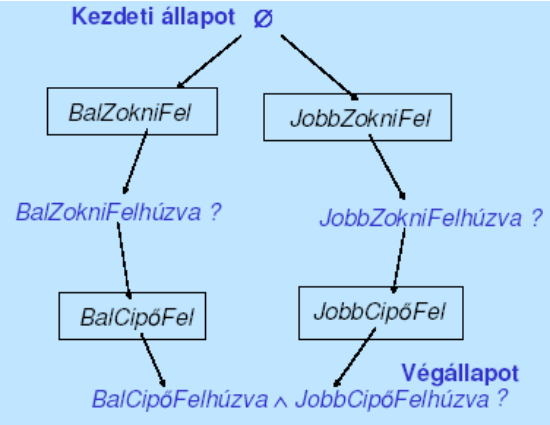




A keresést a szituációk tere helyett a lehetséges tervek terében végezzük el.

- Indulás: egy egyszerű, befejezetlen terv, amit **részleges tervnek** nevezünk (és ami kezdetben „**üres”**).
- A részleges terv **bővítése** újabb elemekkel, míg el nem érünk egy **teljes tervet**, amely már az egész problémát megoldja.
- A keresési folyamat operátorai:
  - finomító operátor - **???**
  - ami pedig nem finomító, az módosító operátor - hibásan megadott terveket javítják ki
- A megoldás az utolsó lépés során keletkező terv, a hozzá vezető út pedig érdektelen.

Tervreprezentáció (példa)



- fontos: figyeljünk az operátorok sorrendjére ha meg van szabva,
  ill. az evidenciák betartására(előbb fel kell húzni a zoknit, és csak utána jöhet a cipő)
- több egyenértékű szekvenciális terv lehetséges

A legkisebb megkötés elve:
mindig csak az adott pillanatban fontos dolgokat kell eldönteni,
és az összes többi döntés későbbre halasztható.

A legkisebb megkötést alkalmazó tervkészítőnek a lépések sorrendjét nem kell rögzítenie.

A részben rendezett tervkészítők képesek olyan tervek kezelésére, melyekben bizonyos lépések végrehajtási sorrendje egymáshoz képest kötött, míg másoké nem rögzített.

A teljesen rendezett tervkészítők ezzel szemben csak olyan tervek kezelésére képesek, amelyek egy egyszerű lépéssorozatból állnak.

Egy **terv** formálisan:
1. A terv **lépései:** Minden lépés a problémához kapcsolódó egy-egy operátor.
2. A **lépések sorrendjére vonatkozó kényszerek**:
Si < ***Sj*** - *Si* megelőzi *Sj*-t, vagyis *Si* végrehajtásának valamennyivel *Sj* végrehajtása előtt kell megtörténnie, **de nem kell szükségszerűen közvetlenül** megelőznie azt.

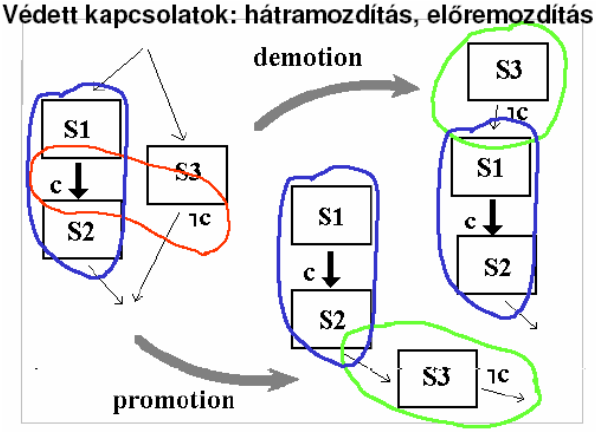
3. **A változó értékek kötéseire vonatkozó kényszerek:**
Az értékkényszerek ***v* = *x*** alakúak, ahol *v* egy adott lépésben szereplő változó, és *x* egy konstans, vagy egy másik változó.

4. **Okozati kapcsolatok: *Si* --- *c* ----> *Sj***
olvasata "*Si* előállítja *c*-t *Sj* számára"
az egyes lépések tervbeli rendeltetését rögzítik :
Si rendeltetése, hogy előállítsa *Sj* számára a *c* előfeltételt.

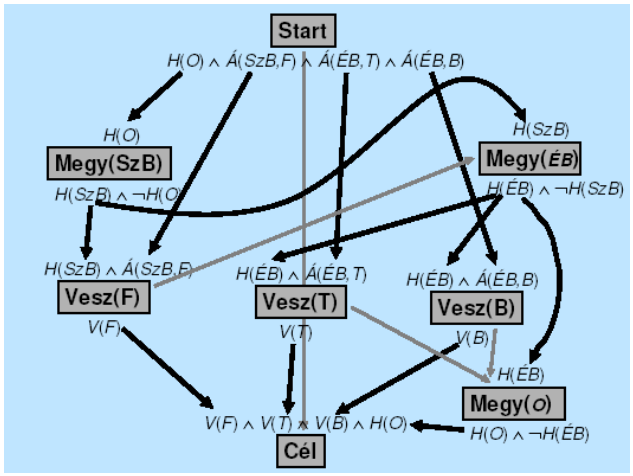
Példa: - a megjelenített cipőhúzós probléma **kezdeti terv**e
Terv(Lépések*:*
{***S1***:*Op*(Cselekvés*:* ***Start***),
S2:*Op*(Cselekvés*:* ***Cél***,
Előfeltétel:*JobbCipőFelhúzva ∧ BalCipőFelhúzva*)}
Sorrendezés*:* { }
Kötések*:* { }
Kapcsolatok*:* { })

A lényeg: a megoldás egy teljes és konzisztens terv legyen.
- **teljes** - minden lépés minden előfeltétele teljesül
  és ha minden lépése olyan elemi operátor, melyet az ágens közvetlenül végre tud hajtani.
- **konzisztens** - ha nem tartalmaz ellentmondó sorbarendezési vagy kötési kényszereket.
  (x=A,x=B értékkötés egyszerre)

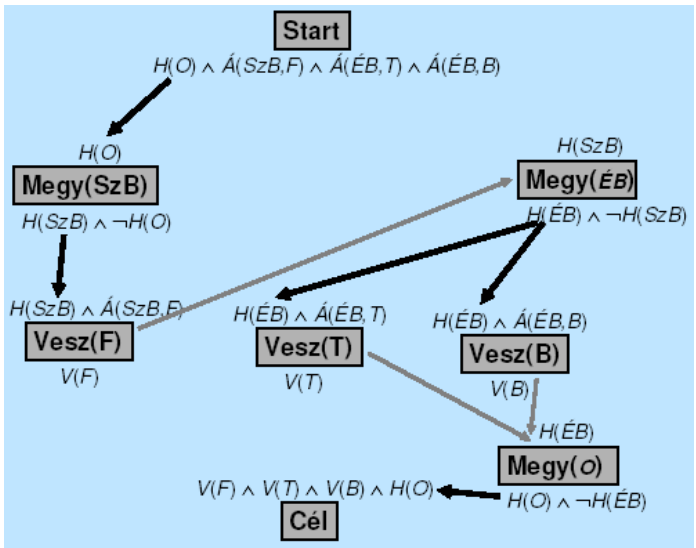
Védett kapcsolatok: hátramozdítás, előremozdítás



Előtte:



Utána:



A kettő ábra közti átmenet levezetése: „eloadas-12b-tervkeszites-2005.ppt” – előadás fólia

Összefoglaló:

Tervkészítő ágens:
- az állapotokat, cselekvéseket, célokat
  és a terveket rugalmasabb módon írja le és használja.

A STRIPS nyelv: cselekvés = előfeltételek + következmények.

Ereje nagyrészt megegyezik a szituációs kalkuluséval.

Bonyolult tárgytartományban nem célszerű a szituációtérben való keresés, ehelyett a tervek terében keresünk.

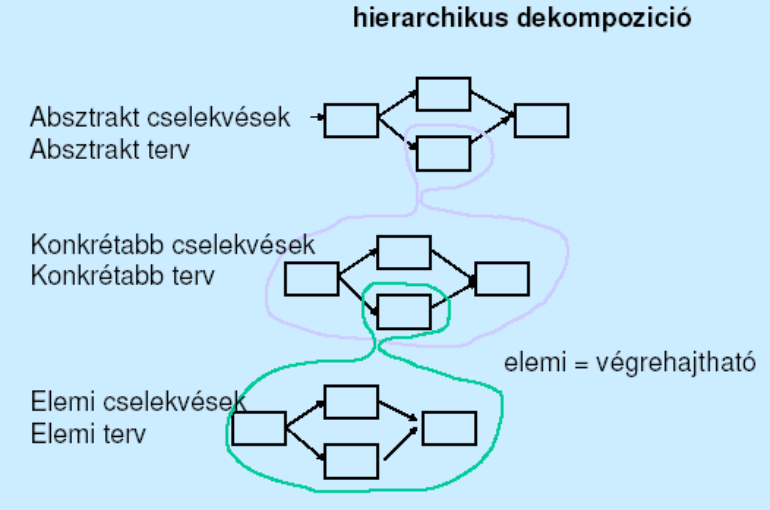
Ha a résztervek nincsenek egymással kölcsönhatásban, akkor ez a módszer hatékony.

A legkisebb megkötés elve alapján a tervkészítő elkerülheti a döntések meghozatalát, amíg erre valami jó ok nem támad.

Az okozati kapcsolatok segítségével a feloldhatatlan ellentmondások hamar felfedezhetők a részleges tervekben, így a meddő keresés elkerülhető.

A Részben Rendezett Tervkészítés algoritmus bizonyíthatóan helyes és teljes tervkészítő.

A tervkészítés során egy absztrakt terv áll elő, amely kevés lépésből áll, de ezek fizikailag nem végrehajthatók.
A megoldás: a hierarchikus dekompozíció - kell készíteni az absztrakt terv alapján egy ágens számára végrehajtható konkrétabb tervet, mely elemi tervekből áll.
(ismétlés) Egy terv teljes - minden lépés minden előfeltétele teljesül
és ha minden lépése olyan elemi operátor, melyet az ágens közvetlenül végre tud hajtani.



A hierarchikus tervkészítés ötlete:

1. A STRIPS nyelvbővítése:
   összetett (nem elemi) operátorok leírása
2. A tervkészítő algoritmus módosítása:
   képes legyen összetett operátorokat
   a felbontásukkal helyettesíteni

Maradjon helyes, teljes és konzisztens és továbbra is érvényesüljön a legkisebb megkötés elve.

Általánosságban hogy mi az elemi és mi az összetett operátor az csak az operátorokat végrehajtó ágens ismeretében pontosítható.

Példa:

Az építési vállalkozó esetében a Beépít(Parketta) elemi operátor, mert csak fel kell bérelni a feladatra egy megfelelő munkást.

A parkettásnak viszont a Beépít(Parketta) összetett operátor, számára pl. a Bever(Szög) az elemi operátor.

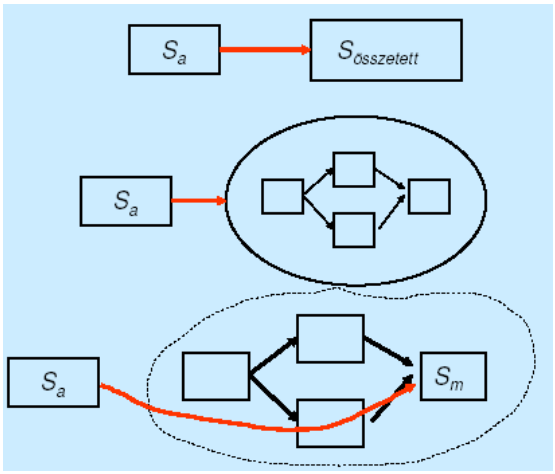
A tervkészítő módosítása:

Megoldás-vizsgálat:

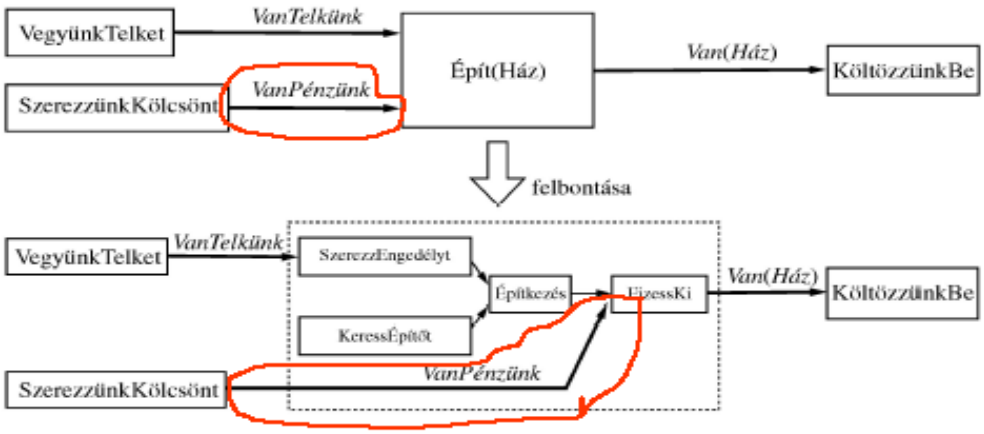
van-e még ki nem elégített el_feltétel?
van-e még nem elemi lépés a tervben?

Finomítás:
egy részcél megválasztása
- egy operátor megválasztása a részcélhoz
egy összetett lépés megválasztása
- egy tervfelbontás megválasztása az összetett lépéshez

Sorrendezés:



A felbontásnál a felbontott összetett operátorhoz tartozó kapcsolat logikus bevonása a felbontott tevbe, lásd ábra:



Közelítő hierarchia: Az elemi terv szintjén az operátornak minden előfeltételét és következményét figyelembe vesszük, magasabb szinteken a tervkészítő elhanyagol ezek közül egyeseket.

Egy operátor előfeltételei **fontossági szintjük** szerint, pl.:

Op(Cselekvés: *Vesz*(*x*),
Következmény: *Van*(*x*) ∧ ¬*Van*(*Pénz*),
Előfeltétel: **1: *Árusít*(*bolt*,*x*) ∧**
2: *Helyszín*(*bolt*) ∧
3: *Van*(*Pénz*)

A kisebb számokkal címkézett előfeltételek fontosabbak. A tervkészítő a problémát először csak az 1. fontossági szintű előfeltételek figyelembe vételével oldja meg. Ha az 1. szinten sikerült megoldást találni, a tervet a 2. fontossági szintnek megfelelő előfeltételekkel bővítjük, stb.

Tervkészítők kényszerei:

Erőforráskényszerek – mértékek használata

- számszerű mértékek bevezetése a tervkészítés folyamatába
- a mértékeket speciális függvényekkel állítjuk elő a megfelelő objektumok alapján
  példa: BenzinSzint = Mennyiség(BenzinAzAutóban) = Liter(25)
- a mértékeket használó tervkészítők általában megkövetelik, hogy a mértékeket előre deklaráljuk

Időzítési kényszerek

Hiány !!!

<u>Feltételes tervkészítés</u> (valami információ mindig hiányzik)

- eshetőségi tervkészítés néven is ismert.

A feltételes tervkészítés hiányos információk birtokában
készít feltételes terveket, amelyek az összes lehetséges,
előforduló helyzetre vagy eshetőségre tartalmaznak megfelelő cselekvéseket.

Az ágens a terv adott pontjain megjelenő érzékelő cselekvés
végrehajtása alapján dönti el, hogy éppen melyik feltétel
teljesül, és a tervének melyik részét hajtsa végre.

Példa:
A bevásárló ágens például beiktathatna a tervbe egy
megfigyelést a megvásárolandó termék árával kapcsolatban,
és ennek alapján dönthetné el, nem túl drága-e

Példa – előadás ppt:

Ha a célunk egy fölfújt gumikerék, melynek leírása:
Fölszerelve(*x*) ∧ *Fölfújva*(*x*),
és a kezdeti feltételek
Fölfújva(*Pótkerék*) ∧ *NemLyukas*(*Pótkerék*) ∧ *Leszerelve*(*Pótkerék*) ∧ *Fölszerelve*(*Gumi1*) ∧
Lapos(*Gumi1*),

De hiányos információnk van, a lapos gumi lehet lyukas, de lehet az is, hogy csak régen volt felfújva.
A feltételes lépés leírása:

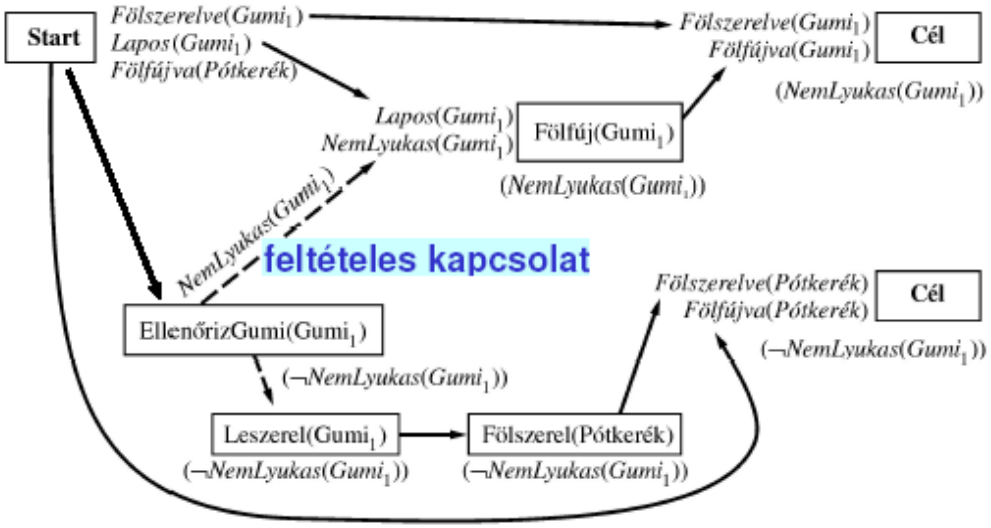
Ha (<*Feltétel*>,<*Következmény*>,<*Egyébként*>).
Ettől kezdve a gumiszerelős terv a következő lépésekből áll:
Ha NemLyukas(*Gumi1*),
[*Fölfúj*(*Gumi1*)],

[*Leszerel*(*Gumi1*),*Fölszerel*(*Pótkerék*)])

Az ágensnek úgy kell tevékenykednie, hogy ezzel biztosítsa a megfigyelések megfelelő eredményét.
Például egy gumi lyukas voltának eldöntésére legyen a *EllenőrizGumi*(*x*) olyan cselekvés, amely előállítja az *x* gumi pillanatnyi állapotát.
Ez egy érzékelő cselekvés.

A feltételes tervek előállítása nagyban hasonlít a korábbi egyszerű tervkészítőre.
A legjelentősebb különbség a tervlépések **kontextusának** a megjelenése.

Egy lépés kontextusa egyszerűen a lépés végrehajtásához szükséges feltételek összessége, amely lényegében azt az ágat határozza meg a tervben, ahol a lépés elhelyezkedik.
A *Fölfúj*(*Gumi1*) kontextusa a *NemLyukas*(*Gumi1*).
Ha egy lépéshez egy adott kontextus tartozott, ezt a terv következő lépései is öröklik.
Példa:



Végrehajtásfelügyelet

Az előbbi egyszerű tervkészítő ágense a tervet "csukott szemmel" hajtja végre, a soron következő cselekvés kiválasztása során nem használja föl a megfigyeléseit (az előírt érzékelést kivéve).
Nagyon sérülékeny stratégia!
A terv végrehajtása közben bekövetkező események nyomon követésével az ágens fel tudja fedezni, ha valamit elhibáz.
Ezután **újratervezést** hajthat végre, hogy a megváltozott helyzetből az eredeti célt elérje.

A tervek meghiúsulásának okait (meg lehet-e előre jósolni a lehetőségeket):

Korlátos határozatlanság: a cselekvéseknek lehetnek váratlan következményei, de ezek lehetséges hatásai fölsorolhatók, és belefoglalhatók a cselekvés leírásába.

Nem korlátos határozatlanság: a váratlan kimenetelek túl sokan vannak ahhoz, hogy teljes számban fölsorolhassuk őket.
Bonyolult és/vagy dinamikus tartományokban: autóvezetés, gazdasági tervezés, hadvezetés, …
A lehetséges következményeknek legfeljebb egy részére tudunk tervezni, és tudnunk kell új tervet készíteni, ha a valóságos helyzet nem úgy alakul, ahogy vártuk.

Értékelés(összegzés) - valós környezetben föllépő problémák

Ágens képes a viselkedését a tartományok és célok explicit reprezentációinak megfelelően alakítani.
Részben rendezett tervkészítéssel kihasználhatja a probléma dekompozició előnyeit, bonyolult tartományokat

is tud kezelni az exponenciális komplexitás káros hatásai nélkül.

Képes olyan tartományokat is kezelni, amelyeken feltételes következmények, keletkező és megszűnő objektumok és elágazások is előfordulnak. Tárolt terveket is használhat egyes részcélok elérésére.
Képes a tartományleírásban megjelenő hibák kezelésére, tud olyan feltételes terveket készíteni, amelyek beszerzik a szükséges hiányzó információkat.
Tudja kezelni a dinamikusan változó világ eseményeit fokozatosan kijavítva terveit, ahogy kielégítetlen célokra vagy hibákra bukkan.

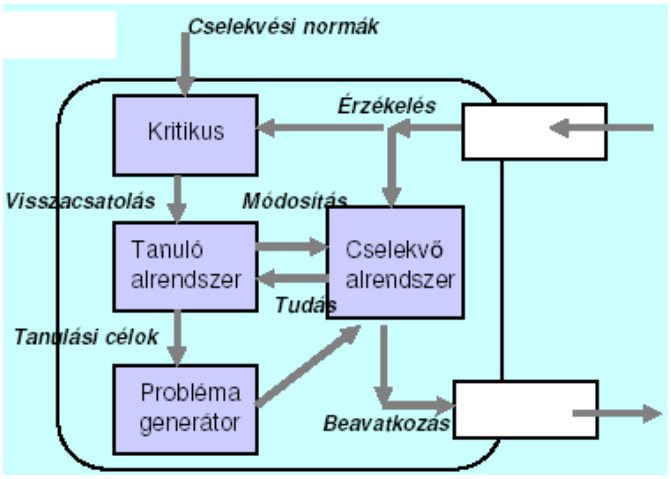
Alkalmazás:
90-es évek, manapság a tervkészítési technikák aranykora
(MI tervkészítés külön tantárgy sok egyetemen)
Nagyon sok hatékonyságra kiélezett módszerek
Sikerek: Sivatagi Háború csapatmozgás tervezése
Élprojektek: Több űrhajóból álló raj mélyűr misszióinak tervezése
Műtétek tervezése
Kritikus helyzetelhárítás döntéstámogatása
Legkomolyabb hazai projektek: SZTAKI rugalmas gyártástervezés, tervadaptáció, terv-újrafelhasználás

(gépi)Tanulás (- machine learning)

Fontos kérdés – Mit biztosít a tanulás egy ágens számára ?
- autonómiát
- rugalmasságot
- robosztusságot
- kinyújtott időtartamot
- fokozott intelligenciát/racionaltást

A tanuló ágens felépítése:

- cselekvő alrendszer
- tanuló alrendszer
- kritikus: közli a tanuló alrendszerrel, hogy az ágens működése mennyire sikeres.
- problémagenerátor: olyan cselekést javasol, amely új, informatív tapasztalatok megszerzéséhez vezet.



A cselekvő alrendszer részei

- a jelen állapotra vonatkozó feltételek közvetlen leképezése a cselekvésekre.
- **egy eszköz**, amely lehetővé teszi, hogy az észlelési sorozatból **a világ fontos tulajdonságaira következtessünk**.
- a világ alakulására vonatkozó információ.
- az ágens lehetséges cselekvéseinek eredményére vonatkozó információ.
- a világ egyes állapotainak kívánatosságát jellemző hasznosság információ.
- egy cselekvés-érték információt - egyes állapotokban mennyire célszerű egyes cselekvések választása.
- az ágens hasznosságát maximalizáló állapotosztályok.

Bármelyik komponens megtanítható, ha adott a megfelelő visszacsatolás.
A komponensek az eddig vett logikák bármelyikében reprezentálhatók.

A rendelkezésre álló visszacsatolás:

- Felügyelt tanulás
  egy komponensnek mind a bemenetét, mind a kimenetét észlelni tudjuk.
- Megerősítéses tanulás
  az ágens az általa végrehajtott tevékenység bizonyos értékelését kapja meg = megerősítés.
- Felügyelet nélküli tanulás
  semmilyen utalás sem áll rendelkezésünkre a helyes kimenetről.

A priori tudás
a kutatások többségében az ágens nem rendelkezik semmilyen előzetes tudással amikor elkezd tanulni.

A különböző feladatok közös magja

A cselekvő alrendszer minden komponense úgy írható le, mint egy-egy függvény (leképezés).
Cselekvés = f(Észlelések, Régebbi cselekvések, TB)
Minden tanulás felfogható úgy, mint egy függvény valamilyen reprezentációjának a megtanulása.

Induktív tanulás
- felügyelt tanulás
- Tiszta induktív következtetés (**indukció**):
  az *f*-re vonatkozó minták egy halmaza alapján, adjon meg egy olyan *h* leképezést (hipotézist), amelyik közelíti *f*-et.

Logikai TB, logikai reprezentáció a logikai kifejezések tanulása:

Két jellegzetes módszer:

döntési fa módszerek:
a logikai kifejezések egy szűkített – a tanulás céljára tervezett - halmazát használják,

verzió-tér megközelítés:
általánosabb, de gyakran nem hatékony.

A legfontosabb döntés, amellyel a tanuló ágens tervezője szembesül:
a kívánt függvény ***reprezentációjának*** megválasztása !

Kompromisszum: a ***kifejezőképesség*** és a ***hatékonyság*** között
Kifejezőképesség: a kívánt függvény jól ábrázolható-e
Hatékonyság: a tanulási probléma kezelhető lesz-e
egy adott reprezentációs nyelv választás esetén

DÖNTÉSI FÁK

Döntési fák tanulása

döntési fa bemenete: egy tulajdonsághalmaz segítségével leírt objektum vagy szituáció

kimenete: egy igen/nem „döntés"

döntési fa = egy logikai függvény
a fa mindegyik **belső csomópontja** = valamelyik tulajdonság tesztje
a csomópontból kiinduló mindegyik **él** = a teszt lehetséges értékeivel címkézett
a fa mindegyik **levél csomópontja** = az a logikai érték, amelyet akkor kell kiadni, ha ezt a levelet elértük.

A döntési fák kifejezőképessége

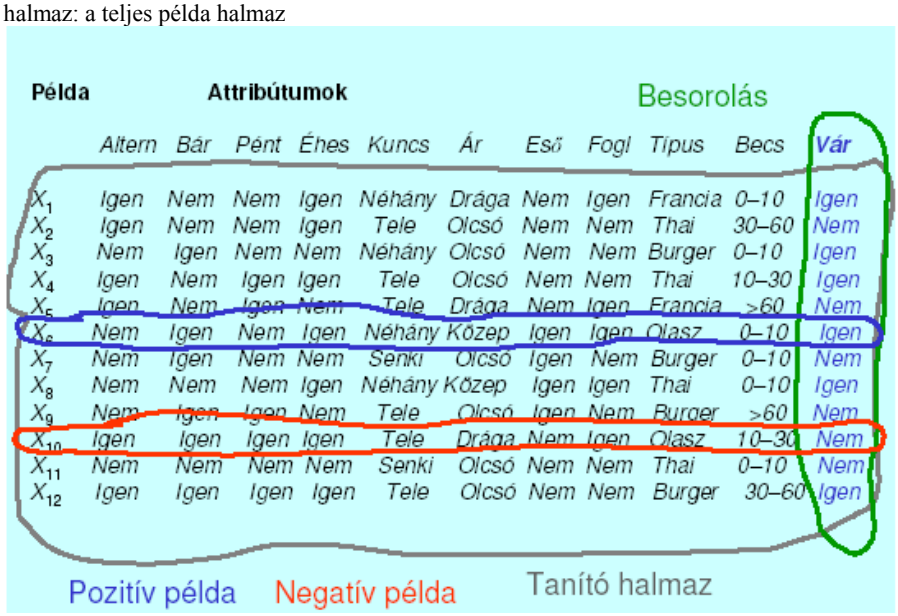
A döntési fák implicit módon egyetlen objektumra korlátozott állításokat fogalmaznak meg = ítélet logika.
A döntési fák kifejezőképessége:
teljesek az ítélet logikai nyelvek osztályán belül,
minden logikai függvény felírható döntési faként.
(az igazságtábla minden sorát felvesszük, mint a fa egy bejárását, az igazságtábla mérete = exp.)

A döntési fa **nem** minden függvény reprezentációjára alkalmas.

Az n attribútumon értelemezett összes logikai függvény száma $2^{(2^n)}$!

Döntési fa kialakítása példák alapján

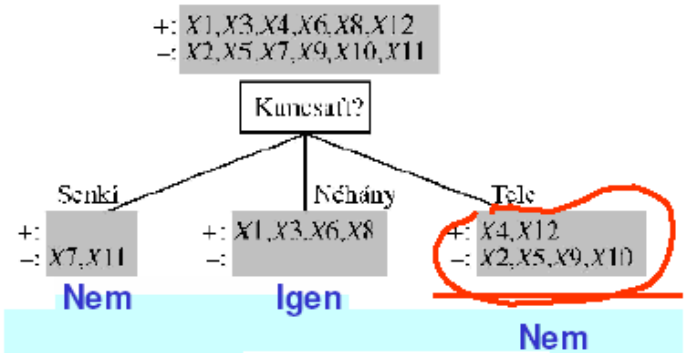
- példa: adottak az attribútumok értékei és a cél predikátum értéke
- példa besorolása a célpredikátum alapján
  pozitív példa – a célpredikátum értéke igaz
  negatív példa – a célpredikátum értéke hamis
- tanító halmaz: a teljes példa halmaz



Ábra: az előbbi fogalmak szemléltetése

Az alapötlet: először a legfontosabb attribútumot teszteljük.
Itt „**legfontosabb**" = amelyik a **legnagyobb eltérést** okozza egy példa besorolásában.
Ilyen módon azt reméljük, hogy kisszámú teszt alapján korrekt besoroláshoz jutunk,
Ami egyben azt is jelenti, hogy a **bejárási utak rövidek** lesznek, és az **egész fa kicsi** lesz.

1. Ha van néhány pozitív és néhány negatív példánk, akkor válasszuk azt az attribútumot, amelyik a legjobban szétválasztja őket
2. Ha az összes megmaradt esetünk pozitív (vagy az összes negatív), akkor készen vagyunk: a válaszunk *Igen* vagy *Nem*.
3. Ha nem maradt egyetlen példánk sem, ez azt jelenti, hogy ilyen példát nem figyeltünk meg eddig. Ilyenkor valamilyen alapértéket adunk vissza, amelyet a csomópont szülőjének többségi besorolási válaszából származtatunk.
4. Ha nem maradt teszteletlen attribútumunk, de maradtak pozitív és negatív példáink, akkor baj van. Ez azt jelenti, hogy **ezeknek a példáknak pontosan megegyezik a leírása, de különböző a besorolásuk.**
   Néhány adat nem korrekt: **zaj** torzítja az adatokat.
   Megoldás? a többségi szavazás használata.



A tanulási algoritmus teljesítményének becslése

A tanulási algoritmus akkor jó, ha jó hipotéziseket szolgáltat azon esetekre, amelyeket nem látott előtte.

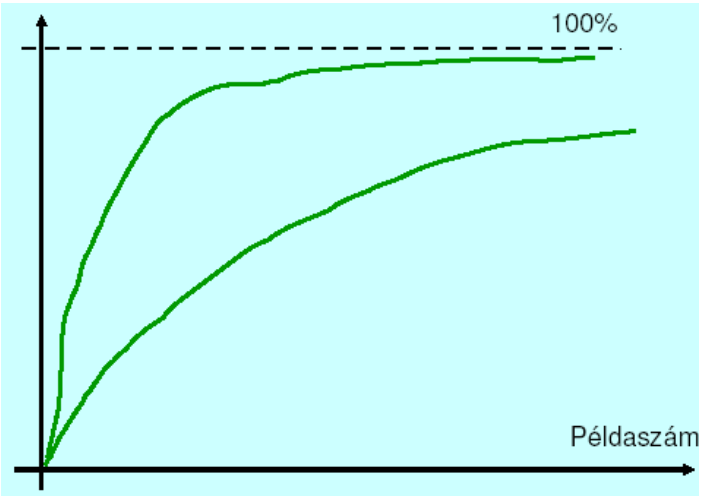
1. Egy nagy példahalmazt bontsuk szét:
egy **tanító halmazra** és egy **teszt halmazra**.

2. A tanuló algoritmust a tanító halmazra alkalmazva
állítsuk elő a *H* hipotézist.

- Vizsgáljuk meg, hogy *H* a teszt halmaz példáinak hány százalékát sorolja be helyesen.

4. Ismételjük a 2-4 lépéseket különböző tanító halmaz méretekre és mindegyik mérethez különböző, véletlenszerűen kiválasztott tanító halmazra.

Tanulási görbe: a tanító halmaz méretének függvényében
(jó, ha a jóslás minősége javul)



A **tökéletes attribútum** a példákat két csoportra bontja,
az egyikbe **csak pozitív**, a másikba **csak negatív** példák tartoznak.

Egy nagymértékben haszontalan attribútum, mint pl. a *Típus* olyan példa halmazokat hoz létre, amelyek nagyjából ugyanolyan arányban tartalmaznak pozitív és negatív példákat, mint az eredeti halmazok.

Az egész döntési fa információ tartalma = a válasz a tanító halmazban található pozitív és negatív példák arányaként
Tegyük fel, hogy a tanító halmaz *p* pozitív és *n* negatív példát tartalmaz:
két válasz: *v1* , *v2*
válaszok valószínűsége: $P(v1) = p/(p+n)$, $P(v2) = n/(p+n)$,
Ekkor **az egész fa információ tartalmának** becslése:

$$I(\frac{p}{p+n},\frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Az attribútum tesztjének **információ nyeresége** az **eredeti** információ igény és a **teszt utáni** új információ igény **különbsége**:

$$Nyereség(A) = I(\frac{p}{p+n},\frac{n}{p+n}) - Maradék(A)$$

ahol Maradék(A):

$$Maradék(A) = \sum_{i=1}^{v}\frac{p_i+n_i}{p+n} I(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i})$$

Példa:

Sz.	Érdekes Lap	Sok Reklám	Sok Script	Sok Link	Sok Text	Témába Vág
X1	I	N	N	I	N	N
X2	N	I	N	N	N	I
X3	I	N	I	I	I	I
X4	I	N	N	N	I	I
X5	I	N	I	N	N	N
X6	N	I	I	N	I	I
X7	I	I	N	I	N	I
X8	N	N	N	N	N	N

Az egész fa információtartalma Ifa = I(3/8, 5/8).

Ifa = I (hamisat eredményező példák száma/példák száma, igazat eredményező példák száma/példák száma)

Az egyes **attributúmok nyeressége**:

ÉrdekesLap nyeressége:

Ifa - (5/8 I(3/5,2/5) + 3/8 I(2/3, 1/3))

Ifa - (igazak száma/példák száma* I(az igazakból az igazat eredményezők száma/igazak száma, az igazakból az hamisat eredményezők száma/igazak száma)
+ hamisak száma/példák száma* I(a hamisakból az igazat eredményezők száma/hamisak száma, az hamisakból az hamisat eredményezők száma/hamisak száma)
)

SokReklám nyeressége:
Ifa - (3/8 I(1, 0) + 5/8 I(2/5, 3/5))

Zaj és túlzott illeszkedés

Ha két vagy több példának azonos a leírása (az attribútumok alapján), de különböző a besorolása (zaj), akkor az algoritmus nem találhat olyan döntési fát, amely az összes példával konzisztens.

Megoldás: minden ilyen levél csomópont vagy az adott bemeneti leírásnak a saját példa halmazán mért többségi besorolását adja vissza, vagy az egyes besorolásoknak a relatív gyakoriságuk alapján becsült valószínűségét adja vissza.

Amikor alapvető információ hiányzik, a tanuló algoritmus is találhat olyan döntési fát, amely az összes példával konzisztens.
Ok: az algoritmus felhasználhatja az ***irreleváns*** attribútumokat is - ha vannak - arra, hogy hamis megkülönböztetéseket tegyen a példák közt.

Ha a lehetséges hipotézisek halmaza nagy, akkor mindig vigyázni kell, nehogy értelmetlen „szabályosságot" találjunk az adatokban = **túlzott illeszkedés.**
Rendkívül általános jelenség, az összes tanulási eljárást sújtja ez a probléma.

Döntési fa nyesés: megakadályozni a nem releváns attribútumok vizsgálatával történő példahalmaz szétbontást.
Példahalmaz kettévágása irreleváns attribútummal:
a kapott részhalmazok azonos arányban tartalmaznak pozitív és negatív példából, mint az eredeti halmaz.

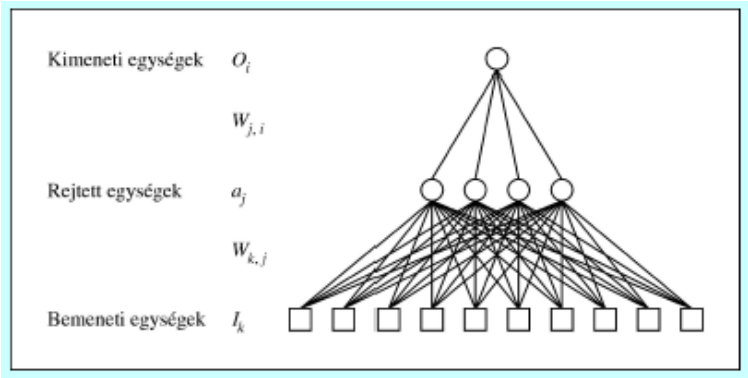
Az információ nyereség közel nulla. Az információ nyereség az irrelevancia jelzésére jól használható.

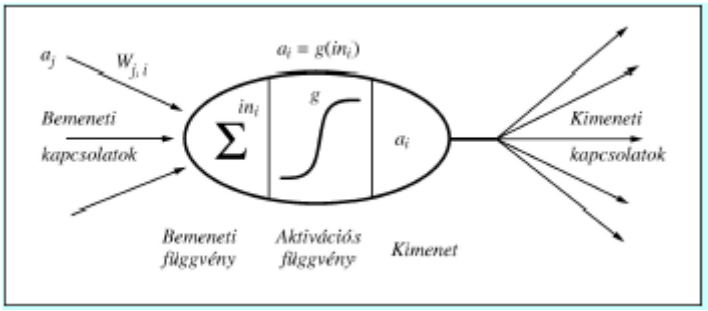
A kérdés: milyen nagy legyen az információ nyereség, hogy egy attribútumot felhasználjunk a példahalmaz megosztására?
Statisztikai teszt: nincs jellegzetes minta = ún. **nulla hipotézis**.
aktuális adathalmaz mennyire tér el a minta teljes hiányától
ha az eltérés már olyan mértékű, hogy nem valószínű, hogy
a minták statisztikai ingadozásából következik, akkor ezt
egy, az adatokban található, lényeges minta jelenlétének
bizonyítékaként értékeljük.

HIÁNY !!! :
eloadas-15d-tanulas-2005 - KIJEGYZETELÉSE

Neurális és valószínűségi hálók tanulása

A neurális háló: nemlineáris approximációt megvalósító, induktív tanulási algoritmussal tanítható matematikai struktúra.





Aktivációs függvény:

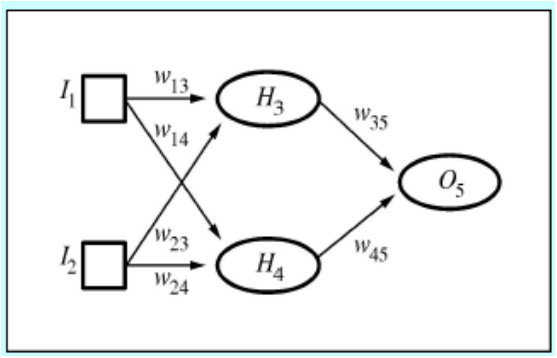
$$a_i \leftarrow g(in_i) = g(\sum_j W_{j,i} a_j)$$

Fajtái:
- ugrás-
- előjel-
- szigmoid-
- arctg-függvény

Hálózati struktúrák:
- előrecsatolt
- visszacsatolt
- egy rétegű – perceptronok
  - nincsenek rejtett rétegek
  - egyszerűbb tanulás
  - nagyon korlátozott a reprezentációs képessége
- több rétegű
  - több rejtett réteg

- egyetlen, megfelelően nagy rejtett réteggel a bemenetek bármely folytonos függvénye reprezentálható

Példa:



$$a_5 = g(W_{3,5}a_3 + W_{4,5}a_4) = g(W_{3,5}g(W_{1,3}a_1 + W_{2,3}a_2) + W_{4,5}g(W_{1,4}a_1 + W_{2,4}a_2))$$

ahol a_i –k az egyes egységek kimeneti értékei, g – az aktivációs függvény, $W_{i,j}$ – az i-ből a j állapotba menő él súlya

Frissítve, kiegészítve: Hadady Zsombor Tamás
/2005-dec.,2006-jan./