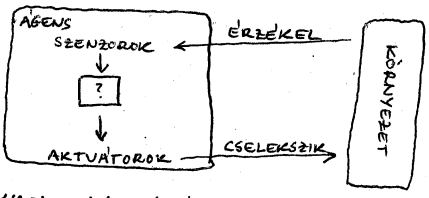
L2 INTELLIGENS RENDSZEREK – INTELLIGENS ÁGENSEK

ÁGENS RENDSZER FOGALMA

ÁGENS - nek tekinthető minden olyan rendszer, amely érzékeli a környezetét és ennek hatására cselekszik. Egy adott környezetben való cselekvésre helyezi a hangsúlyt, akkor a cselekvésbe beletartozik a környezet érzékelése éppúgy, mint a következtetés és a környezetbe való beavatkozás.



Egy ágens cselekvésének kiválasztása egy adott pillanatban függ az összes addig érzékelt bemenettől
=> az ágens viselkedését az az ÁGENS FÜGGVÉNY írja le, ami egy érzékelés sorozatból egy cselekvésre képez → külső leírás

ÁGENS PROGRAM: az ágens függvényének megfelelő viselkedés implementációja, az ágens architektúrán megvalósítva → belső leírás

ÁGENSEK TÍPUSAI: ember, robot, szoftver

RACIONÁLIS ÁGENS: minden lehetséges érzékelés sorozatra olyan cselekvést választ, ami várhatóan maximalizálja a hatékonyságát a bemenő érzékelés sorozat és a rendelkezésére álló belső tudás alapján → mindig a jót választja

HATÉKONYSÁG: objektív mérték. Aszerint választjuk meg, hogy milyen eredményt várunk a környezetben és nem aszerint, hogy az ágensnek miként kell viselkednie.

Egy ágens racionalitása függ:
- a siker definiáló hatékonyság kritériumtól
- az ágens környezetére vonatkozó előzetes ismeretektől
- az ágens által kivitelezhető cselekvésektől
- az ágens által az adott pillanatig érzékelt bemenő sorozattól

A racionalitás csupán a várható legjobb cselekvés kiválasztása, nem az abszolút legjobb, tehát egy racionális ágens nem feltétlenül mindent tud, nem tökéletes. Fontos azonban, hogy helyzetét felmérve, kellő információt begyűjtve cselekedjen.
Az érzékelés során az ágens kezdeti tudása a környezetről módosulhat → tanul a tapasztalatokból
Egy racionális ágens autonóm: meg kell tanulnia tapasztalati úton a kezdetben hiányosan megadott tudást kompenzálni

ÁGENS KÖRNYEZETEK

környezet ~ feladat – ágens ~ megoldás

KÖRNYEZET SPECIFIKÁLÁSA: PEAS – Performance, Environment, Actuators, Sensors

KÖRNYEZET TULAJDONSÁGAI

- Teljesen megfigyelhető: az ágens minden információt el tud érni a környezetéből, amire szüksége van a cselekvés kiválasztásához
  Részben megfigyelhető: ha zajos, nem megfelelően működő szenzorok vagy hiányos adatok miatt a szükséges információknak csak egy részét éri el az ágens. A nem megfigyelhető információkat modell alapján kitalálja.
- Determinisztikus: ha a környezet következő állapotba történő megváltozása csak az aktuális állapotától és az ágens által kivitelezett cselekvéstől függ
  Nem determinisztikus vagy sztochasztikus: korábbi állapotok is befolyásolják. Általában a nem megfigyelhető környezet nem det.
- Epizodikus: ha az ágens aktuális cselekvésének kiválasztása nem függ a korábbi cselekvésektől → atomi epizódok: érzékelés cselekvés
  Szekvenciális: az aktuális cselekvés hatással van a későbbiekre, az ágensnek előre kell tervezni
- Statikus: a környezet nem változik az idő függvényében. Az ágensnek nem kell foglalkozni a környezettel és az idő múlásával amíg eldönti mit csináljon.
  Dinamikus: a környezet változhat az ágens működése során, figyelembe kell venni a cselekvés kiválasztása közben is.
  Szemi-dinamikus: a környezet maga nem változik, de az ágens teljesítménye igen
- Diszkrét: (a környezet állapotára, az idő múlására, illetve az érzékelés - cselekvésre nézve) ha tisztán elkülöníthető halmazoknak tekinthetők
  Folytonos: (-ik-) ha nem felbontható
- Egy-ágens: egy ágens cselekvései befolyásolják a környezet megváltozását
  Több-ágens: - versengő: az egyik ágens hatékonyságának maximalizálása minimalizálja a másik ágens hatékonyságát
  - együttműködő: az egyik ágens hatékonyságának növelése hozzájárul a másik ágens hatékonyságának növeléséhez

ÁGENSEK SZERKEZETE

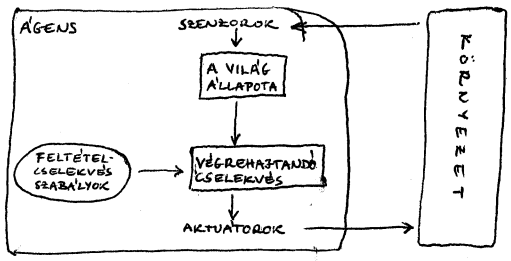
ágens = architektúra + program

ÁGENS PROGRAM: a szenzoroktól kapott aktuálisan érzékelt inputra egy cselekvés - választ küld az aktuátorokra. Csak az aktuális inputot veszi figyelembe, ha a teljes érzékelés sorozatra igény van, akkor memóriára van szüksége.

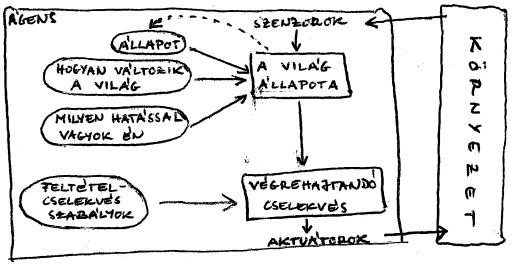
~~ÁGENS PROGRAMOK TÍPUSAI~~

ÁGENS PROGRAMOK TÍPUSAI

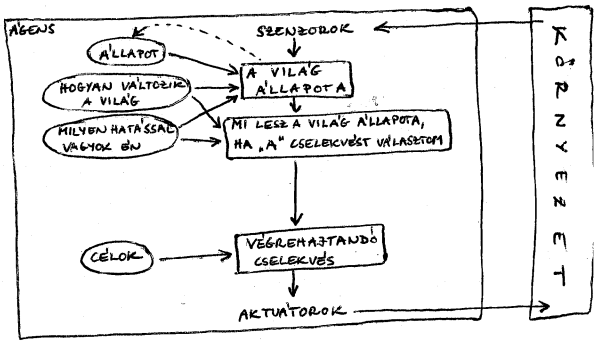
- **Reflex ágens**: az ágens csak az aktuális szenzoros input alapján választ cselekszik, nem veszi figyelembe a korábban érzékelt inputokat, háttér tudást sem használ.
  Feltétel-cselekvés szabályok érvényesülnek (if-then rules)
  Nagyon egyszerű, de bizonyos intelligenciával rendelkező ágensek.
  Csak akkor működik, ha a környezet teljesen megfigyelhető, egyébként véletlenszerű cselekvés kiválasztás.



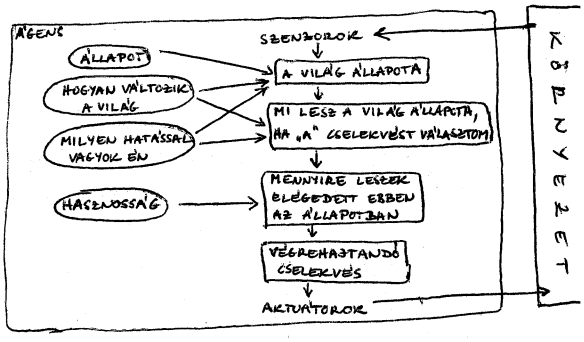
- **Modell-alapú reflex ágens**: a részben megfigyelhető környezet kezelésére az ágens nyomon követi a környezet azon részének változásait, amik nem tud megfigyelni.
  Azaz az ágens fenntart egy belső állapotot, amit a korábbi érzékelés sorozat határoz meg, így a nem megfigyelhető állapot tényéről egy részéről információval rendelkezik. Ezt a tudást a „világ működéséről" nevezzük modellnek.



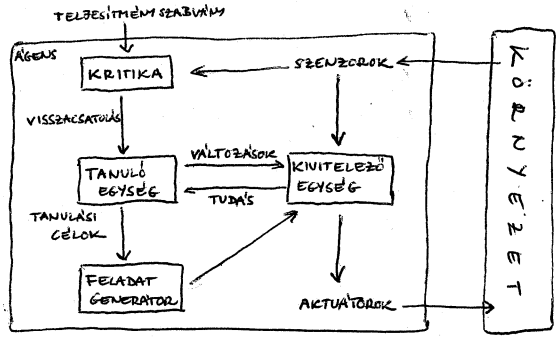
- **Cél alapú ágens**: a környezet állapotának ismerete nem mindig elég a cselekvés kiválasztásához, a helyes döntéshez ezért az ágensnek szüksége van egy cél információra, ami az elérendő állapotot jellemzi.
  A cél bizonyos esetekben közvetlenül elérhető a kiválasztott cselekvés kivitelezése során, míg máskor összetett folyamatra van szükség.
  A célról való tudást használja fel a cselekvések irányítására (keresés és tervezés), a lehetséges cselekvések eredményét összehasonlítja.
  A cselekvés a környezetet az elérendő cél felé módosítja.



- **Hasznosság-függvény alapú ágens**: a hasznosság-függvény egy állapotot vagy állapot sorozatot egy valós számra képez, ami az állapot elérésével való „megelégedettséget" jellemzi.
  Csupán a cél ismerete nem mindig lehetséges, vagy több, egymásnak ellent mondó bitűzött cél közül kell választani.
  Azt a cselekvést választja, amelyik legjobban növeli a hasznosság-függvény értékét.



- **Tanuló ágens**: a tanulás lehetővé teszi, hogy az ágens egy kezdetben ismeretlen környezetben működjön, még jobban mint azt valamilyen kezdeti információ birtokában tehetné.
  Négy komponensre bontható:
  - tanuló egység: fejleszti az ágenst
  - kivitelező egység: kiválasztja a megfelelő cselekvést
  - kritika: információt szolgáltat vissza arról, hogy hogyan működik az ágens, hogyan lehet javítani a teljesítményét
  - feladat generátor: kíváncsi helyzeteket állít elő az ágens tudásának fejlesztéséhez



L3 L4,5

KERESÉS

PROBLÉMAMEGOLDÁS FORMALIZÁLÁSA

Egy ágens a számára ismeretlen értékű közötti állapotok lehetőségeiből úgy választ, hogy megvizsgálja azokat a cselekvéssorozatokat, amelyek ismert állapotba vezetnek és ezek közül a lehető legjobbat választja. → keresés

PROBLÉMAMEGOLDÁS LÉPÉSEI:

- Cél meghatározása: segíti az ágens viselkedésének megtervezését az elérendő állapotok korlátozásával. Az aktuális állapot és az ágens hatékonyságának segítségével határozható meg.
- Feladat meghatározása: az a folyamat, amely meghatározza, hogy az adott cél elérése érdekében milyen cselekvésekre és állapotokat vegyen figyelembe az ágens.
- Keresés: a lehető legjobb cselekvéssorozat kiválasztása. Bemenete a feladat, kimenete a megoldás.
- Megoldás: a keresés eredményeként kapott cselekvéssorozat.
- Kivitelezés: a megoldásként kapott akció-sorozat egyes lépéseinek végrehajtása

A problémamegoldó ágensek tervezésekor statikus, megfigyelhető, diszkrét és determinisztikus környezetet feltételezünk.

KERESÉSI FELADAT FORMÁLIS DEFINÍCIÓJA – 4 komponens

- Kezdeti állapot: az az állapot, ahonnan az ágens indul
- Követő függvény: az ágens számára elérhető lehetséges cselekvések leírása. Egy állapothoz <cselekvés, következő állapot> párokat rendel
- Cél teszt: megmutatja, hogy az aktuális állapot cél-állapot-e vagy sem.
- Út-költség: minden egyes útvonalhoz egy szám értéket rendel; általában tükrözi az ágens hatékonyságát
  Lépés költség: az „a" cselekvés hatására x-ből y állapotba jut ⇒ $c(x,a,y)$

Állapot tér: a kezdeti állapot és a követő függvény által határozható meg: a kezdeti állapotból elérhető összes állapot
Útvonal: állapotoknak olyan sorozata az állapot térben, amelyeket cselekvések sorozata köt össze
Megoldás: a kezdeti állapotból a cél állapotba vezető út
Optimális megoldás: a legalacsonyabb költséggel rendelkező megoldás

KERESÉSI ELJÁRÁSOK

Keresési fa / gráf: a kezdeti állapot és a követő függvény által generált állapot tér
Keresési csomópont: kezdeti állapot, a keresési fa gyökere
Kifejtés: a követő függvény alkalmazása az aktuális állapotra ⇒ új állapot-halmaz generálása
Keresési stratégia: a következőnek kifejtendő állapot meghatározása
Csomópont: 5 komponenssel jellemezhető adatstruktúra: állapot, szülő csomópont, cselekvés (ami a csomópontot generálta), út-költség ($g(n)$: a kezdeti állapotból a csomópontba vezető útvonal költsége), mélység (a kezdeti állapotból a csomópontba vezető lépések száma)

PROBLÉMAMEGOLDÁS HATÉKONYSÁGA

- Teljesség: az algoritmus biztosan megtalálja-e a megoldást ha létezik
- Optimalitás: az algoritmus az optimális megoldást találja-e meg
- Idő komplexitás: mennyi ideig tart a megoldás megtalálása
- Tér komplexitás: mennyi memóriát igényel a megoldás megtalálása

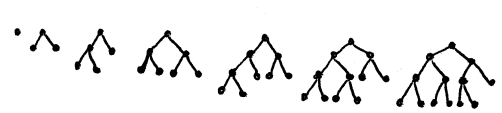
Az idő- és tér komplexitás függ a feladat környezetétől, ezért (állapot gráfok esetén) a következő mennyiségekkel jellemezhető:
- elágazási tényező (b): egy csomópont maximális számú leszármazottja
- a gyökérhez legközelebbi rétegben található cél csomópont mélysége (d)
- az állapottérben található leghosszabb útvonal (m)

„GYENGE" (UNINFORMED) KERESÉS

Gyenge keresési stratégiák esetén nem áll rendelkezésre semmilyen többlet információ az állapotokról, csak ami a feladat definíciójában meg van adva. Ezek a stratégiák csak arra képesek, hogy egy csomópont leszármazottait generálják, illetve meg tudják különböztetni a cél állapotokat a nem cél állapotoktól.

• SZÉLESSÉGI KERESÉS

Először a gyökér csomópont kerül kifejtésre, majd ennek leszármazottai, utána ezek leszármazottai stb. ⇒ Egy adott rétegben minden csomópont kifejtésre kerül mielőtt az algoritmus a következő rétegre lép.



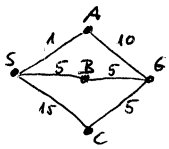
Jellemzői:
- teljes
- optimális, ha az út-költség a csomópont mélységének nemcsökkenő függvénye
- idő komplexitás: $O(b^{d+1})$
- tér komplexitás: $O(b^{d+1})$ (minden kifejtett csomópont a memóriában marad)

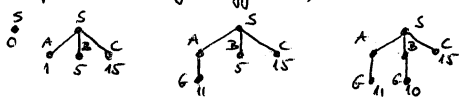
b: minden állapotnak b leszármazottja van

⇒ exponenciális komplexitású keresési problémák közül csak a legkisebbeket lehet „gyenge" keresési stratégiák segítségével megoldani

EGYENLETES-KÖLTSÉGŰ KERESÉS

Mindig a legalacsonyabb út-költségű csomópontot fejti ki. Ha minden lépés költsége egyenlő, akkor a szélességi keresséssel azonos.





Jellemzői:

- teljes, ha minden lépés költsége $\geq \varepsilon$
- optimális, ha minden lépés költsége $\geq \varepsilon$
- idő- és tér komplexitás: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$, ahol C^* az optimális megoldás költsége.

MÉLYSÉGI KERESÉS

Mindig a legmélyebb csomópontot fejti ki.

- nem teljes: megoldást nem tartalmazó végtelen ág esetén nem jut át másik ág kifejtésére
- nem optimális: rossz választás esetén a gyökérhez közel eső megoldás ellenére is b mély ágazat járható be
- idő-komplexitás: $O(b^m)$
- tér-komplexitás: $O(bm)$ miután egy csomópont kifejtésre került és az algoritmus az összes leszármazottját bejárta eltávolítható a memóriából

b: elágazási tényező $\quad$ m: maximális mélység

MÉLYSÉGKORLÁTOZOTT KERESÉS

Mélységi keresés előre meghatározott l mélységkorláttal. Az l-nél mélyebben található csomópontokat úgy kezeli, mintha nem lenne utódunk. Ez a módszer megoldja a végtelen-út problémát. A jó mélységkorlát meghatározása általában nem lehetséges a feladat megoldása előtt.

Jellemzői

- nem teljes, ha $l < d$
- nem optimális, ha $l > d$
- idő-komplexitás: $O(b^l)$
- tér-komplexitás: $O(bl)$

ITERATÍVAN MÉLYÜLŐ MÉLYSÉGI KERESÉS

A cél eléréséig fokozatosan növeli a mélységkorlátot, így az optimális l értéket találja meg. Ez akkor történik, amikor $l=d$, tehát elérte a gyökérhez legközelebb eső megoldás mélységét.

Jellemzői

- teljes, ha az elágazási tényező véges
- optimális, ha az út-költség a csomópont mélységének nemcsökkenő függvénye
- idő-komplexitás: $O(b^d)$ (az utolsó réteg csomópontjait egyszer, az utolsó előtti réteg csomópontjait kétszer generálja és így tovább)
- tér-komplexitás: $O(bd)$

Nagy állapottér és ismeretlen mélységű megoldás esetén a legjobb „gyenge" keresési algoritmus.

KÉTIRÁNYÚ KERESÉS

Egyszerre két keresést futtat: az egyiket a kezdő állapotból előre, a másikat a cél állapotból visszafelé. Akkor áll meg, ha a kettő találkozik. A keresés során egy csomópont kifejtése előtt mindig megvizsgálja, hogy a másik irányú keresés elérte-e már az adott csomópontot.

Jellemzői:

- teljes
- optimális } ha mindkét irányban szélességi keresést alkalmaz
- idő-komplexitás: $O(b^{d/2})$
- tér-komplexitás: $O(b^{d/2})$ (legalább az egyik irányú keresési fát el kell tárolni a memóriában)

	SZÉLESSÉGI KERESÉS	EGYENLETES KÖLTSÉGŰ K.	MÉLYSÉGI KERESÉS	MÉLYSÉG-KORLÁTOZOTT K.	ITERATÍVAN MÉLYÜLŐ K.	KÉTIRÁNYÚ KERESÉS
TELJES	IGEN b véges	IGEN b véges l.k. $\geq \varepsilon$	NEM	NEM	IGEN b véges	IGEN b véges, szélességi k.
OPTIMÁLIS	IGEN egyenlő lépésköltség	IGEN	NEM	NEM	IGEN egyenlő lk.	IGEN egyenlő l.k. szélességi k.
IDŐ-KOMPLEXITÁS	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
TÉR-KOMPLEXITÁS	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$

INFORMÁLT HEURISZTIKUS KERESÉS

Az olyan keresési stratégiák, amelyek a feladat definíción ~~kívül~~ felül további feladat-specifikus információval rendelkeznek, hatékonyabban találnak megoldást.

ELŐRETEKINTŐ KERESÉS (best-first search)

A kifejtendő csomópontot egy kiértékelő függvény segítségével választja ki, ami a céltól való távolságot méri.
=> A legkisebb függvényértékű csomópontot fejti ki. Általában nem a legjobb csomópontot találja meg, hanem csak a legjobbnak tűnőt, a lehető legjobbat.

HEURISZTIKUS FÜGGVÉNY: $h(n)$ = az n csomópontból a cél csomópontba vezető legolcsóbb út becsült költsége
Ha n a cél csomópont, akkor $h(n) = 0$.

MOHÓ ELŐRETEKINTŐ KERESÉS (greedy best-first search)

A célhoz legközelebb eső csomópontot próbálja meg kifejteni => gyorsan megoldáshoz fog vezetni
Csak a heurisztikus függvény alapján értékeli ki a csomópontokat => $f(n) = h(n)$ ($f(n)$: kiértékelő függvény)
Jellemzői:
- nem teljes (elindulhat egy végtelen úton így, hogy nem fordul vissza vagy próbál ki más lehetőséget)
- nem optimális
- tér-, idő komplexitás: $O(b^m)$ → jól megválasztott heurisztikus függvény esetén csökkenthető

A* KERESÉS

A csomópontokat a csomópontba jutás költségének ($g(n)$) és a csomópontból a célba jutás költségének ($h(n)$) kombinációjaként előálló függvénnyel értékeli ki: $f(n) = g(n) + h(n)$
$f(n)$ = az n csomóponton át vezető legolcsóbb megoldás becsült költsége
=> a legolcsóbb megoldás megtalálására érdemében először mindig a legkisebb $f(n)$ értékű csomópontot fejti ki.
Jellemzői:
ELFOGADHATÓ
- optimális, ha $h(n)$ megengedhető heurisztika, azaz sosem becsüli túl a cél elérésének költségét
A kiértékelő függvény értéke monoton nő egy útvonal mentén => kontúrok: egy adott értéknél (kontúr-érték) kisebb vagy egyenlő f értékű csomópontok halmaza
Az algoritmus a keresési ~~pont~~ csomópontból kiindulva f növekvő értékeinek megfelelő ~~sávokban~~ sávokban koncentrikus sávokban elhelyezkedő csomópontokon lépked. → megfelelő heurisztika esetén a sávok a cél csomópont felé nyúlnak
Ha C^* az optimális megoldás költsége, akkor
- A^* minden olyan csomópontot kifejt, ahol $f(n) < C^*$
- A^* kifejthet több olyan csomópontot is, melyre $f(n) = C^*$ mielőtt a cél csomópontot megtalálja

- teljes: mivel f növekvő értékeinek megfelelő sávokba lép, biztosan eljut egy olyan sávba, melyre $f(n) = C^*$
Metszés: bizonyos lehetőségek elvetése azok megvizsgálása nélkül => A^* nem fejti ki azokat a csomópontokat, melyekre $f(n) > C^*$
- idő-komplexitás: optimálisan hatékony - nincs olyan optimális algoritmus, amely kevesebb csomópontot fejt ki
~~időkomplexitása~~ ~~Exponenciális növekedés~~ ~~exponenciálisan rosszabbnak~~ elkerüléséhez feltétele: $|h(n) - h^*(n)| \leq O(\log h^*(n))$ → a heurisztika hibája lassabban nő mint az aktuális út költség logaritmusa
$h^*(n)$: az n csomópontból a célba jutás tényleges költsége
- tér komplexitás: nagy - minden bejárt csomópontot el kell tárolni

MEMÓRIA KORLÁTOZOTT HEURISZTIKUS KERESÉS

ITERATÍVAN MÉLYÜLŐ A* (IDA*)

Az iteratívan mélyülő mélységi keresés mintájára a mélységkorlát helyett itt az $f = g + h$ értéket változtatja.
Minden iteráció során a vágási érték (cutoff) annak a csomópontnak az f értéke lesz, amely az előző iterációban kapott vágási értéket meghaladó f-értékek közül a legkisebb.

REKURZÍV ELŐRETEKINTŐ KERESÉS (RBFS)

Az előretekintő algoritmust ~~utánozza~~ működését valósítja meg lineáris térben. A keresés során ha a korábban már bejárt csomópontok mentén jobb alternatívát talál akkor onnan folytatja.
Jellemzői:
- optimális, ha a heurisztikus függvény megengedhető
- tér-komplexitás: $O(bd)$
- idő-komplexitás: függ a heurisztikus függvény pontosságától és attól, hogy milyen gyakran kell megváltoztatni az a legjobb útvonalat a csomópontok kifejtése során. → akár exponenciális komplexitás

EGYSZERŰSÍTETT MEMÓRIA KORLÁTOZOTT A* (SMA*)

A*-nak megfelelő működés során addig fejti ki a legjobb levelet, amíg rendelkezésére áll elég memória. A legrosszabb levél csomópontot eldobja (legmagasabb f-értékkel rendelkezőt), ennek értékét a szülő csomópontnál megtartja.
Egy „eldobott" részfa őse tudja a részfában található legjobb útvonal értékét → csak akkor állítja vissza a részfát, ha minden más útvonal rosszabbnak adódik.
Jellemzői:
- teljes, ha van elérhető megoldás → $d \leq$ memória mérete (csomópontokban mérve)
- optimális, ha van elérhető optimális megoldás

Memória korlátozásból kifolyólag kiszámíthatatlan idő-komplexitást eredményezhetnek.

HEURISZTIKUS FÜGGVÉNYEK

$h()$: a megoldáshoz vezető útvonal költségének becslése. $h(\text{cél állapot}) = 0$

Heurisztikus függvény meghatározása:
- matematikai módszerrel
- INTROSPECTION
- INSPECTION
- programozó által

Heurisztika: meghatározza a kifejlesztő szempontot a heurisztikus keresés során figyelembe veszi a heurisztikus függvényt és az út költségét.

Egy heurisztika ELFOGADHATÓ / MEGENGEDHETŐ, ha nem becsüli túl a cél elérésének költségét

DOMINÁNS: A h_1 heurisztika dominánsa h_2 felett, ha $h_1(n) \geq h_2(n)$. → az a heurisztika vezet a leggyorsabb kereséshez, ami dominálja a többit (és megengedhető)

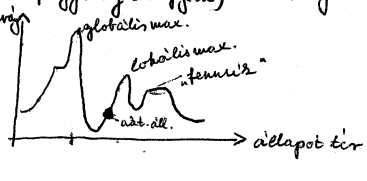
Heurisztika tervezése: jó heurisztika meghatározása nehéz,
- implicit szerepel a feladat meghatározásában (pl. euklideszi távolság útvonal keresés esetén)
- a feladatban szereplő megszorítások elengedésével kapható meg

LOKÁLIS KERESÉSI ÉS OPTIMALIZÁLÁSI FELADATOK

Ha a megoldáshoz vezető útvonal nem számít, csak a cél elérése a lényeg akkor lokális keresési algoritmusokat alkalmazhatunk. Egyetlen állapot használatával működnek és általában csak ennek az aktuális állapotnak valamelyik szomszédjára léphetnek.

Előnyei: - nagyon kis memória használat - konstans
- gyakran értelmes megoldást találnak akár nagy vagy végtelen (folytonos) állapot térben.

ÁLLAPOTTÉRFELSZÍN (state-space landscape)
Az állapot pozícióját és értékelését (heurisztikus függvény vagy cél függvény alapján) ábrázolja.
Cél: globális minimum / maximum megtalálása



- teljes lokális keresési algoritmus: mindig megtalálja a célt, ha létezik
- optimális lokális keresési algoritmus: mindig a globális minimumot / maximumot találja meg

HEGYMÁSZÓ KERESÉS (hill-climbing)

Mindig a növekvő irányba lép, akkor terminál ha csúcspontba ért, ha nincs nála nagyobb értékű szomszéd
Nem néz előre, csak a közvetlen szomszédait vizsgálja. Ha több lehetőséggel talál a továbblépésre akkor véletlenszerűen választ. Elakadhat lokális maximum, „gerinc" vagy „fennsík" esetén.

MÓDSZEREK
- legmeredekebb lejtő (a legmeredekebben emelkedő érték felé lép)
- oldalt mozgó (csak megengedi az aktuális értékkel megegyező értékű állapotba lépést)
- véletlenszerűen újraindító (több hegymászó keresést indít véletlenszerűen kiválasztott kezdeti pontokból, ha valamelyik célba ér akkor megáll) → teljes - nem marad lokális maximumban
- sztochasztikus (a növekvő érték felé mutató lépések közül véletlenszerűen választ)
- első-választás (véletlenszerűen generálja a következő lépés jelölteket, amíg olyat nem kap ami jobb mint az aktuális)

SIMULATED ANNEALING

A hegymászó keresés és a teljesen véletlenszerű lépkedés kombinációja.
Költség minimalizálásra törekszik => „először rázd erősen, hogy lendüljön a labda, majd egyre finomabban"
→ ha a véletlen lépés során jobb állapotba jut, akkor tegye meg, ha nem, akkor $e^{-\Delta E/T(t)}$ valószínűséggel tegye.

LOKÁLIS NYALÁBOLT KERESÉS (local beam search)

k darab véletlenszerűen generált állapotból indul. Minden lépés során a k állapot összes utódját meghatározza. Ha célállapotot talál, akkor megáll, egyébként kiválasztja a k darab legjobb állapotot és folytatja
=> hasznos információ adódik át a k párhuzamos szál között
Hátrány: a k állapot gyorsan koncentrálódhat az állapottér egy kis részén =>

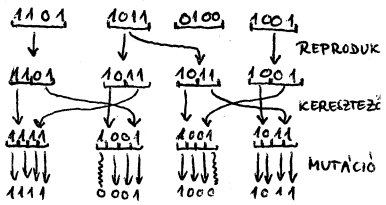
SZTOCHASZTIKUS NYALÁBOLT KERESÉS: nem a legjobb k leszármazottal tartja meg, hanem egy növekvő értékű valószínűség szerint véletlenül választja ki a k darab utódot

GENETIKUS ALGORITMUSOK

A leszármazott állapotok két szülő állapot kombinációjából generálódnak, nem csupán egy állapotból

POPULÁCIÓ: k darab véletlenül választott kezdeti állapot
EGYED: minden egyedet / állapotot egy string reprezentál (általában bináris abc felett)
Cél: a bit-sorozaton optimalizálni egy adott függvényt
FITNESS FÜGGVÉNY: a értékelő függvény. Jobb állapot esetén nagyobb értéket ad vissza

MŰVELETEK
- REPRODUKCIÓ: véletlenszerűen párok kerülnek kiválasztásra adott valószínűség alapján
- KERESZTEZŐDÉS: minden pár esetén kereszteződési pontokat választ. A folyamat elején a populáció nagy mértékű diverzitása miatt nagy lépésekkel teszi meg az algoritmus, majd az állapotok egyre közelebb kerülnek egymáshoz, kisebb változások lesznek.
- MUTÁCIÓ: kis valószínűséggel véletlen változások lépnek fel



CIKLUS A GENETIKUS ALGORITMUSOKBAN

G_x: N bitsztring aktuális generációja $(b_0, b_1, \dots b_{N-1})$

- minden b_i-re legyen $p_i = fitness(G_i) / \Sigma_j fitness(b_j)$
- $G_{x+1} = \emptyset$
- for $i=0$; $i<N/2$; $i=i+1$
  - válassz ki két szülőt $P(szülő = b_i) = p_i$ valószínűséggel
  - kereszteződés: véletlenszerűen cserélj fel néhány bitet a két szülő között, hogy új bitsorozatokat kapj
  - mutáció: az új bitsorozatban valamilyen kis valószínűség alapján inverzálj bitet
  - a két új bitsorozatot add hozzá G_{x+1}-hez

ONLINE KERESÉS

Offline keresési algoritmus: a teljes megoldást kiszámítja mielőtt a valóvilágban alkalmazná.
Online kereső ágenset: először végrehajtanak egy cselekvést, akciót, majd a környezet megfigyelése után kiszámítják a következő lépést.
Nem kell számontartani az összes lehetséges állapotváltozást, csak azt ami valóban meg is történik.
Jól alkalmazható dinamikus, semi-dinamikus és sztochasztikus tartományokban illetve szükséges alkalmazni felderítési feladatoknál, amikor az ágens nem ismeri az állapotokat és cselekvéseket.

Az online ágens ismeri
- az s állapotban lehetséges cselekvések listáját : ACTIONS(s)
- a lépés-költség függvényt : c(s,a,s') (csak akkor használható, ha már tudja, hogy s'-be fog jutni
- a cél-teszt függvényt : GOAL-TEST(s)

Az ágens nem tudja elérni egy állapot kimenetelét, csak ha valóban végrehajtja az állapotban végrehajtható összes cselekvést. Feltételezzük, hogy az ágens felismeri azokat az állapotokat amikor már járt és hogy a cselekvései determinisztikusak. Az ágensnek rendelkezésére állhat egy heurisztikus függvény is.
Az ágens célja egy cél állapot elérése minimális költség áron, ahol a költség a teljes útvonalra vonatkozik, amit az ágens bejár.
KOMPETITÍV RÁTA = aktuális költség / legrövidebb útvonal költsége

ONLINE MÉLYSÉGI KERESÉS

A csomópontokat fizikai elhelyezkedésüknek megfelelő sorrendben fejti ki (az online algoritmus csak azt a csomópontot fejtheti ki ahol éppen van)
- a visszalépésnek is fizikai sorrendben kell történnie
  => a cselekvésnek visszafordíthatóaknak kell lenniük
- legrosszabb eset: minden csomópont kétszer kerül kifejtésre
- megoldás: online iterakív mélységi keresés

ONLINE LOKÁLIS KERESÉS

- HEGYMÁSZÓ ALGORITMUS: Mivel csak egy állapotot tárol a memóriában, ezért már önmagában online algoritmus.
  Lokális maximumba beragadhat -> véletlen újraindítás
  => „véletlen séta": az elérhető állapotok közül választ véletlenszerűen ha az algoritmus beragad
  -> előbb vagy utóbb biztosan célba ér
  Hegymászó algoritmus memóriával kiegészítve: eltárolja a már bejárt állapotok alapján a lehető legjobb H(s) becslést a cél eléréséhez szükséges költségre. =>
- TANULÓ VALÓS-IDEJŰ A* ALGORITMUS (LRTA*)
  - H(s): eltárolja minden csomópont alapján a legjobb ml-költséget a cél elérésére
  - a még ismeretlen csomópontokhoz a lehető legalacsonyabb költséget rendeli
  - tapasztalat alapján frissíti H(s) értékét

KERESÉSI ALGORITMUSOK ÖSSZEHASONLÍTÁSA

EFFEKTÍV ELÁGAZÁSI TÉNYEZŐ (effective branching factor)
- egy keresési fa elágazási rátája, ahol minden csomópontból ugyanannyi él indul ki (BFS)
- d: a megoldás mélysége
  N: kifejtett csomópontok száma => b^* : $N = ((b^*)^{d+1} - 1)/(b^* - 1) = 1 + b^* + (b^*)^2 + (b^*)^3 + \dots + (b^*)^d$
- jól jellemzi a heurisztika minőségét

L6 JÁTÉK STRATÉGIÁK

A több, egymással versengő ágenst tartalmazó környezetekben felmerülhetnek „ellenséges" keresési problémák
→ játékok: determinisztikus, teljesen megfigyelhető környezet, ahol két ágens van, akik felváltva „lépnek" és a hasznosság értékük (utility value) egymás ellentétes.

JÁTÉKOK CSOPORTOSÍTÁSA:
- játékosok száma szerint (2 vagy több)
- versengő vagy együttműködő
- nulla összegű (összes nyereség − összes veszteség)
- diszkrét vagy folytonos
- véges vagy végtelen
- determinisztikus vagy sztochasztikus
- teljes vagy részleges információ tartalmú?

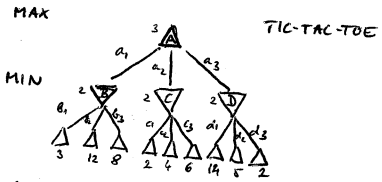
OPTIMÁLIS DÖNTÉS

Két játékos: MAX és MIN, MAX lép először, majd a játék végéig felváltva lépnek.
A játék végén a nyertes játékos jutalmat pontokat kap, a vesztes játékos büntetést pontokat.

JÁTÉK FORMÁLIS DEFINÍCIÓJA ~ keresési feladat $(S, s_0, succ(), S \to P(S), T, V(), T \to \mathbb{R})$
- kezdő állapot: a tábla állását tartalmazza és a soron következő játékost - s_0
- követő függvény: (lépés, állapot) párok listája - succ()
- vég-teszt: meghatározza, hogy a játék mikor ér véget → terminális állapotok - T
- hasznosság függvény (cél függvény): a terminális állapotokhoz számértékkel rendel. pl.: nyertes: +1, vesztes: −1, döntetlen: 0

Játék-fa: a kezdeti állapot és a rá fel legális lépései határozzák meg

OPTIMÁLIS STRATÉGIÁK



lehetőségi stratégia: meghatározza MAX lépését a kezdeti állapotban, majd MAX lépéseit a MIN válaszaként eredményül kapott állapotokban ...

MINIMAX ÉRTÉK v. JÁTÉK ELMÉLETI ÉRTÉK (minimax value / game theoretic value)
Egy csomópont/állapot minimax/GTV értéke annak a terminális állapotnak az értéke (hasznosság / utility), ahova a játék jutni fog, ha mindkét játékos optimálisan játszik.

$$\text{minimax-érték}(n) = \begin{cases} \text{hasznosság}(n) & \text{ha } n \text{ terminális állapot} \\ \max_{s \in succ(n)} \text{minimax-érték}(s) & \text{ha } s \text{ MAX csomópont} \\ \min_{s \in succ(n)} \text{minimax-érték}(s) & \text{ha } s \text{ MIN csomópont} \end{cases}$$

MINIMAX ALGORITMUS

Rekurzív keresési algoritmus:
- saját lépés esetén olyat választ, hogy a kapott állapotban a minimax-érték maximális legyen
- ellenfél lépése esetén olyat választ, hogy a kapott állapotban a minimax-érték minimális legyen

=> kezdetben a terminális állapotokhoz rendeli az értékeket, majd ezeket visszaterjeszti a gyökérig minimax döntések alapján

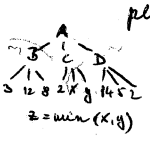
```
MinimaxValue(s) =
  if (s is terminal) return V(s)
  else let {s_1, s_2 ... s_k} = succ(s)
       let V_i = GTV(s_i) for each i
       if (player-to-move(s) = 1) return max(V_i)
       else return min(V_i)
```

Jellemzői:
- idő-komplexitás: $O(b^m)$
- tér-komplexitás: $O(bm)$ - ha az összes utódot egyszerre generálja
  $O(m)$ - ha egyenként generálja az utódokat
- effektív elágazási tényező: b

ALFA-BÉTA METSZÉS (alpha-beta pruning)

A helyes minimax-döntés kiszámításához nem szükséges mindig az összes csomópontot megvizsgálni → a keresési fa egyes ágait figyelmen kívül hagyhatjuk.
=> alfa-béta metszés: a minimax algoritmusnak felel meg, de elhagyja azokat az ágakat amik nem befolyásolják a végső döntést

Tekintsünk egy olyan n csomópontot, ahova a játékos el tud jutni. Ha a játékosnak van egy jobb választási lehetősége (m) úgy, hogy m n-nél szintén vagy még feljebb áll, akkor az n állapotba sosem fog eljutni a játék során => elhagyható részfa



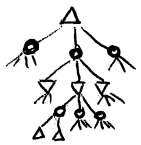
pl.: $GTV(gyökér) = \max(\min(3,12,8), \min(2,x,y), \min(14,5,2))$
$= \max(3, \min(2,x,y), 2)$
$= \max(3, z, 2)$ ha $z \leq 2$
$= 3$
=> a döntés független az elhagyott x és y levelektől

2 paraméter: - alfa: MAX számára az eddig bejárt út során található legjobb (legmagasabb) értéke
- béta: MIN számára az eddig bejárt út során található legjobb (legalacsonyabb) értéke

Jellemzői: • idő-komplexitás: $O(b^{m/2})$
• effektív elágazási tényező: $\sqrt{b}$

VÉLETLEN TÉNYEZŐT TARTALMAZÓ JÁTÉKOK

A játék során előre nem meghatározható kimenetelű tényező is szerepet játszanak (pl. dobókocka)
A keresési fát a MIN és MAX csomópontok mellett VÉLETLEN csomópontokkal kell kiegészíteni
A lehető legjobb értékű lépést keressük → pontos érték nem számítható ⇒ várható érték



EXPECTIMINIMAX ALGORITMUS

A terminaló, MIN és MAX csomópontok viselkedése ugyanaz, mint eddig, a véletlen csomópontok a lehetséges értékek súlyozott átlagát kapják:

$$\text{expectiminimax}(n) = \begin{cases} \text{hasznosság}(n) & \text{ha } n \text{ terminaló állapot} \\ \max_{s \in succ(n)} \text{expectiminimax}(s) & \text{ha } n \text{ MAX csomópont} \\ \min_{s \in succ(n)} \text{expectiminimax}(s) & \text{ha } n \text{ MIN csomópont} \\ \sum_{s \in succ(n)} P(s) \cdot \text{expectiminimax}(s) & \text{ha } s \text{ véletlen csomópont} \end{cases}$$

Jellemzői: • idő-komplexitás: $O(b^m n^m)$, ahol n a véletlen lehetőségek száma

KIÉRTÉKELÉS

A kiértékelő függvény egy becslést ad a játék várható hasznosságára egy adott csomópontban. ~ heurisztika.
Egy játékprogram teljesítménye nagy mértékben függ a kiértékelő függvény megválasztásától.
- a terminaló állapotokhoz ugyanazt az értéket kell rendelnie, mint a valódi hasznosság függvény
- gyors számítás

⇒ a korlátozott számítási lehetőség miatt a végső kimenetet gyakran „találgatással" állapítja meg.
- az állapotot jellemző tulajdonságai alapján azokat kategóriákba, ekvivalencia osztályokba sorolja → az egyes állapotokról nem tudja megmondani, hogy milyen eredményre vezetnek, de a csoporton belül a különböző eredményt adó csomópontok arányát meg tudja határozni. ⇒ várható érték
- az állapotok jellemzőiből számított értékek kombinációját határozza meg

KERESÉS MEGÁLLÍTÁSA

Amikor a keresés leállítása szükséges, a kiértékelő függvény segítségével kiértékeli a játék állását.
- fix mélységkorlát esetén: úgy választjuk meg, hogy a mélységkorlát elérése ne halasztja meg az időkorlátot
- iteratívan mélyülő algoritmus esetén: az időkorlát elérésekor az addig elért legmélyebb keresési pontot adja vissza

Tökéletesebb módszerek:
- csak nyugalmi csomópontban állhat meg: a közeljövőben nem fog nagyon „elszállni" az értéke
  A nem nyugalmi csomópontokat tovább fejti.
- horizont-effektus: az ellenfél részéről egy elkerülhetetlenül komoly károsodást okozó lépés kövekezik.
  ⇒ a „látóhatárnál" mélyebben található jó és rossz lehetőségeket az állapotértékben nem lehet számításba venni
  Megoldás: kezdetben keresebb lépés vizsgálata → a látóhatár/horizont kitolása

17 LOGIKAI ÁGENSEK - ÍTÉLET KALKULUS (PROPOSITIONAL LOGIC)

MI központi kérdése: tudásábrázolás és gondolkodás (reasoning)
- részben megfigyelhető környezetében érzékelt információk kiegészítése
- természetes nyelvek megértése
- rugalmasság (új feladatok, tanulás, alkalmazkodás)

LOGIKA: tudás ábrázolás elsődleges eszköze.
Logikai ágensek tudása mindig jól meghatározott: minden állítás igaz vagy hamis az adott világban

TUDÁS ALAPÚ ÁGENSEK

Tudás-bázis (KB): a tudást reprezentáló nyelven megadott információk a világról
Két művelet: új információ hozzáadása és információ lekérdezése
Következtetés: a meglévő információból új levezetése

Az ágens új információként hozzáadja a tudásbázishoz, amit a környezetből érzékel; a tudásbázisból lekérdezi, hogy milyen cselekvést hajtson végre.

Tudást reprezentáló nyelv meghatározása:
- deklaratív: a tervező a környezetről való kezdeti tudást mondatonként hozzáadja a rendszerhez. → egyszerű
- proceduális: a tervező az elvárt viselkedést programkód formájában adja meg a rendszernek → hatékony

LOGIKA

szintaktika: megadja a jól-formált mondatokat
szemantika: minden lehetséges világ (modell) esetén megadja a mondatok igazságértékét
pl.: m α modellje, ha α igaz az m modell esetén

következmény (entailment): ha egy mondat szükségszerűen levezethető egy másikból, akkor ez következménye a másodiknak
$\alpha \models \beta$ akkor és csak akkor, ha minden modellre, melyre α igaz β is igaz

Egy következtetési algoritmus
- IGAZSÁGTARTÓ, ha csak következmény mondatokat vezet le (sound / truth-preserving)
- TELJES, ha minden következmény mondatot levezet (complete)

Ha a tudásbázis igaz a valós világban, akkor bármely olyan α mondat, melyet egy igazságtartó következtetéssel lehet a tudásbázisból levezetni, igaz lesz.

ÍTÉLETKALKULUS (PROPOSITIONAL LOGIC) - KIJELENTÉSLOGIKA

Szintaktika: megengedett mondatok definiálása
- Atomi mondatok: propozíciók - igaz vagy hamis értékük lehet, jelölés: P, Q, R...
  IGAZ értéke mindig igaz, HAMIS értéke mindig hamis
- Összetett mondatok: egyszerű mondatokból logikai kötőszavak segítségével állnak elő
  Kötőszavak: negáció ($\neg$), konjunkció ($\wedge$), diszjunkció ($\vee$), implikáció ($\rightarrow$), ekvivalencia ($\leftrightarrow$)

Szemantika: minden propozíciós szimbólumhoz megadja az igazságértékét → interpretáció
$\models_i \varphi$ → φ igaz az i interpretációra nézve
$\not\models_i \varphi$ → φ hamis az i interpretációra nézve

szemantikai szabályok:
$\models_i T$ minden i-re
$\not\models_i F$ minden i-re
$\models_i \neg\varphi$ iff $\not\models_i \varphi$
$\models_i \varphi \wedge \psi$ iff $\models_i \varphi$ és $\models_i \psi$
$\models_i \varphi \vee \psi$ iff $\models_i \varphi$ vagy $\models_i \psi$
$\models_i P$ iff $i(P) = T$

Összetett mondatok igazságértéke: az egyszerűbb komponensek igazságértékének kombinációja a kötőszavak szer
→ igazságtábla

P	Q	¬P	P∧Q	P∨Q	P→Q	P↔Q
H	H	I	H	H	I	H
H	I	I	I	H	I	I
I	H	H	I	H	H	I
I	I	H	I	I	H	H

Következtetés: annak meghatározása, hogy van-e olyan α mondat, melyre $KB \models \alpha$ => az összes modellre ellenőrzi, hogy α igaz-e ott, ahol KB igaz
Jellemzői:
- igazságtartó, teljes
- teljes
- idő komplexitás: $O(2^n)$
- tér komplexitás: $O(n)$

MONDATOK TULAJDONSÁGAI

- EKVIVALENCIA · α és β mondatok ekvivalensek, ha ugyanazokra a modellekre igazak
  $\alpha \equiv \beta$ akkor és csak akkor, ha $\alpha \models \beta$ és $\beta \models \alpha$
- ÉRVÉNYESSÉG : egy mondat érvényes, ha minden modellre igaz → tautológia $\models \alpha$
- KIELÉGÍTHETŐSÉG : egy mondat kielégíthető, ha létezik olyan modell, melyre igaz

KÖVETKEZTETÉSI TÉTEL (ENTAILMENT / DEDUCTION THEOREM):
α és β mondatokra $\alpha \models \beta$ akkor és csak akkor, ha $\alpha \rightarrow \beta$ érvényes
$\alpha \vdash \beta$ iff $\vdash (\alpha \rightarrow \beta)$

REDUCTIO AD ABSURDUM
$\alpha \vdash \beta$ akkor és csak akkor, ha $(\alpha \wedge \neg\beta)$ nem kielégíthető

ÁTÍRÁSI SZABÁLYOK / EKVIVALENCIA SZABÁLYOK

- KOMMUTATIVITÁS
  $P \wedge Q \equiv Q \wedge P$
  $P \vee Q \equiv Q \vee P$
  $P \leftrightarrow Q \equiv Q \leftrightarrow P$
- ASSZOCIATIVITÁS
  $((P \wedge Q) \wedge R) \equiv (P \wedge (Q \wedge R))$
  $((P \vee Q) \vee R) \equiv (P \vee (Q \vee R))$
- DISZTRIBUTIVITÁS
  $(P \wedge (Q \vee R)) \equiv ((P \wedge Q) \vee (P \wedge R))$
  $(P \vee (Q \wedge R)) \equiv ((P \vee Q) \wedge (P \vee R))$
  $(P \rightarrow (Q \vee R)) \equiv ((P \rightarrow Q) \vee (P \rightarrow R))$
  $(P \rightarrow (Q \wedge R)) \equiv ((P \rightarrow Q) \wedge (P \rightarrow R))$
- KÉTSZERES NEGÁCIÓ
  $\neg\neg P \equiv P$
- DE MORGAN SZABÁLYOK
  $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$
  $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$
- KONTRAPOZÍCIÓ
  $(P \rightarrow Q) \equiv (\neg Q \rightarrow \neg P)$
- EGYÉB EKVIVALENCIÁK
  $(P \rightarrow Q) \equiv (\neg P \vee Q)$
  $(P \leftrightarrow Q) \equiv ((P \rightarrow Q) \wedge (Q \rightarrow P))$
  $(P \leftrightarrow Q) \equiv ((P \wedge Q) \vee (\neg P \wedge \neg Q))$
  $(P \wedge \neg P) \equiv$ HAMIS
  $(P \vee \neg P) \equiv$ IGAZ

KÖVETKEZTETÉS

LEVEZETÉSI SZABÁLYOK (INFERENCE RULES)

Olyan következtetési láncok levezetésére alkalmazható, amelyek végül a kívánt cél eléréséhez vezetnek.

- MODUS PONENS
  $\dfrac{\alpha \rightarrow \beta,\ \alpha}{\beta}$ Ha $\alpha \rightarrow \beta$ és α adott, akkor β ezekből következik
- ÉS-ELIMINÁCIÓ
  $\dfrac{\alpha \wedge \beta}{\alpha}$ $\dfrac{A_1, A_2 \dots A_n}{A_i}$ $i \in [1 \dots n]$ Egy több tagú konjunkcióból bármelyik konjugált tag következik
- ÉS-BEVEZETÉS
  $\dfrac{A_1, A_2 \dots A_n}{A_1 \wedge A_2 \wedge \dots \wedge A_n}$
- VAGY-BEVEZETÉS
  $\dfrac{A_i}{A_1 \vee A_2 \vee \dots \vee A_n}$ $i \in [1 \dots n]$
- EGYSÉG REZOLÚCIÓ
  $\dfrac{(A \vee B) \wedge \neg B}{A}$

REZOLÚCIÓ

BIZONYÍTÁS : a következtetési szabályok sorozatának alkalmazása. A bizonyítás megegyezik egy keresési feladatnál a megoldás megkeresésével. Egy bizonyítási hatékony, ha nem foglalkozik a jelentéstelen propozíciókkal, akármennyi is van belőlük.

MONOTONITÁS : a következtetett mondatok halmaza csak új információ hozzáadására esetén növekedhet.
Minden α és β mondatra, ha $KB \models \alpha$ akkor $KB \wedge \beta \models \alpha$

KONJUNKTÍV NORMÁL FORMA : Mivel a rezolúció csak literálok diszjunkciójára alkalmazható, ezért szükséges, hogy a KB csak ilyen diszjunkciókat tartalmazzon. Minden ítélet kalkulusban szereplő logikai kifejezés ekvivalens literálok diszjunkciójának konjunkciójával.

REZOLÚCIÓ LEZÁRTJA (RC(S)) : ~~egyes~~ kifejezések egy S halmazának lezártja azoknak a kifejezéseknek az összessége, amelyek S elemeinek, illetve az S-ből S „leszármazottainak" ismételt rezolúciójaként ~~előállítható~~ előállítható

TÉTEL : Ha kifejezések egy halmaza nem kielégíthető, akkor ezek lezártja tartalmazza az üres kifejezést.

HORN KLÓZOK

L11 KORLÁTOZÁS-KIELÉGÍTÉSI PROBLÉMA (CONSTRAINT SATISFACTION PROBLEM - CSP)

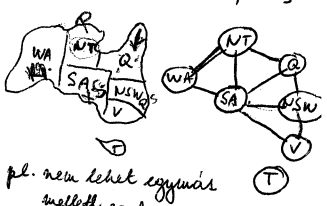
Keresési feladatok: állapotér állapotai ~ „fekete doboz"
CSP: az állapotok és a cél leírt egy szabványos, strukturált és egyszerű reprezentációval jellemezhetőek.

CSP

- Változók egy véges halmaza: $X = \{x_1, x_2 \dots x_n\}$
- Minden változó lehetséges értékeinek véges tartománya: $x_i : D_i = \{a_{i1}, a_{i2}, \dots a_{ini}\}$
- Korlátozások halmaza: a változók által egyszerre felvehető értékekre vonatkozó korlátozás $\{c_1, c_2 \dots c_m\}$
  - változók közötti kapcsolatok (pl. $x_1 \neq x_2$, $x_1 < x_2$ ...)
  - az x_i, x_j, x_k változók között fennálló c_{ijk} korlátozás az összes lehetséges kombináció részhalmaza
    $c_{ijk} \subseteq \{(v_i, v_j, v_k, \dots)$ ahol $v_i \in D_1, v_2 \in D_2 \dots\}$

pl.: x_1, x_2 $D_1 = \{1,2,3\}$ $D_2 = \{2,3\}$
korlátozások: $x_1 = x_2 \Rightarrow \{(2,2),(3,3)\}$
$x_1 < x_2 \Rightarrow \{(1,2),(1,3),(2,3)\}$

KONZISZTENS/LEGÁLIS HOZZÁRENDELÉS: a változókhoz olyan értéket rendel, melyek nem sértik meg egyik korlátozást sem.
TELJES HOZZÁRENDELÉS: minden változónak értéket ad
MEGOLDÁS: olyan teljes hozzárendelés, amely nem sérti meg egyik korlátozást sem
KORLÁTOZÁSI GRÁF: a csomópontok a változóknak felelnek meg, az összeköttetések pedig a korlátozásoknak => egyszerűsíti a feladatot
- általánosítások



pl. nem lehet egymás mellett egyforma színű terület

KERESÉSI FELADAT ~ INKREMENTÁLIS FORMULA
- kezdeti állapot: üres hozzárendelés { }, a változóknak nincs értéke adva
- követő függvény: értéket ad egy üres változónak úgy, hogy az ne alkosson konfliktusban már korábban értékelt változóval.
- cél teszt: azt vizsgálja, hogy az aktuális hozzárendelés teljes-e
- út költség: minden lépésnek konstans értéke van.

CSP CSOPORTOSÍTÁSA
- diszkrét, véges tartományú változók - pl. térkép színezés
- Boolean CSP: a változók értéke igaz vagy hamis lehet
- végtelen tartományú változók - pl. sztringek, számok => nem lehet megoldani a lehetséges hozzárendelések vizsgálatával => „korlátozás-nyelv" (constraint language használata pl. $x_1 + 5 \leq x_3$
- folytonos tartomány - pl. időmérés
- unáris korlátozások: egy változó értékére vonatkozik => a tartomány leszűkítésével kielégíthető
- bináris korlátozások: két változó közötti kapcsolatra vonatkozik
  - grafikus / mátrix reprezentáció
  - minden összetettebb korlátozás-kombináció átalakítható binárissá
- abszolút korlátozások: megszegése kizár egy lehetséges megoldást
- preferencia korlátozások: bizonyos megoldásokat előnyben részesít

KERESÉSI ALGORITMUSOK CSP-K ESETÉN

BACKTRACKING
Olyan mélységi keresés, amely minden lépésben egy változónak ad értéket és visszalép, ha nincs lehetséges konfliktus mentes, legális értékadásra lehetőség az aktuális változó esetén. Nagyobb feladatok esetén nem hatékony.

FORWARD CHECKING
Amikor egy X változónak értéket ad, végigneézi az összes olyan Y változót, ami X-hez kapcsolódik egy korlátozáson keresztül és még nincs hozzá érték rendelve, majd kitörli ezek tartományából azokat az értékeket, melyek miatt konfliktusban állna X-szel. Ha így az algoritmus egy üres tartományt eredményez valamely Y-ra, akkor X-nek nincs legális hozzárendelése.

ARC CONSISTENCY
Korlátozás-terjesztés: egy korlátozás következményeit korábbi lépéseit egyik változóról a másikra =>
Egy összeköttetés két csomópont között (arc) konzisztens, ha az egyik csomópont (kiinduló csomópont) összes lehetséges értéke esetén van olyan hozzárendelési konfiguráció a másik csomópontra, ami konzisztens az előzővel. Az inkonzisztens összeköttetések konzisztenssé tehetők, ha a csomópont tartományából kivesszük az inkonzisztenciát okozó értéket. → Minden alkalommal újabb inkonzisztencia állhat elő => újra kell ellenőrizni amíg minden ilyen konfliktusban álló helyzet megszűnik.

BACKJUMPING
Konfliktus-halmaz: az X változó konfliktus-halmaza azoknak a korábban értéket kapott változóknak a halmaza, amiket egy korlátozás köt X-hez.
Ha a FORWARD-CHECKING algoritmus során egy X változó értékadásakor Y tartományából töröl egy értéket, akkor Y-t hozzá kell venni X konfliktus-halmazához. → A legutóbb a konfliktus halmazba bekerült változóra ugrik vissza amikor szükséges → ugyanaz az eredmény, mint FORWARD CHECKING

CONFLICT-DIRECTED BACKJUMPING:

X_j az aktuális változó, $conf(X_j)$ a konfliktus halmaza

Ha X_j minden értékére inkonzisztens, akkor visszaugrik a legutóbbi X_i-re $conf(X_j)$-vel és módosítja $conf(X_i)$-t

$$conf(X_i) \leftarrow conf(X_i) \cup conf(X_j) - \{X_i\}$$

=> Visszaugrik a keresési fa utolsó helyes pontjára de nem alkalmazkodik, hogy ugyanabba a hibátlan irányba induljon újra, mint korábban.

HEURISZTIKUS MÓDSZEREK

A változók kiválasztásának sorrendjének és az értékadás sorrendjének megválasztására.

- MINIMUM REMAINING VALUES (MRV)

  Azt a változót választja következőnek, aminek a legkevesebb legálisan felvehető értéke van
  => „holt-ágak" magasabbra kerülnek

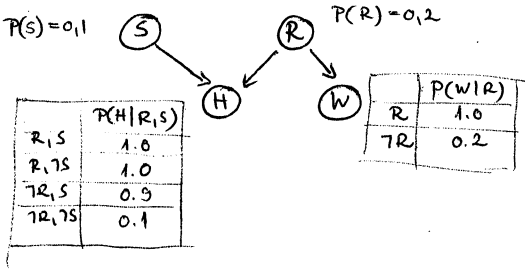
  Nem alkalmazható, ha minden változóra ugyanannyi lehetőség van.
  => „DEGREE HEURISTICS": az elágazási tényező csökkentésére törekszik a jövőbeli választások során azáltal, hogy azt a változót választja, ami a legtöbb korlátozást eredményezi a többi, értékkel még nem rendelkező változóra.

- LEAST CONSTRAINING VALUE

  Azt az értéket választja egy változónak, ami a korlátozási gráfban a változó szomszédjainak értékei közül a legkevesebbet zárja ki.

BAYESIAN NETWORKS

- „WET LAWNS"



	P(H|R,S)
R,S	1.0
R,¬S	1.0
¬R,S	0.9
¬R,¬S	0.1

	P(W|R)
R	1.0
¬R	0.2

$P(R|H) = P(H|R) \cdot P(R) / P(H)$

$P(H|R) = P(H|R,S) \cdot P(S) + P(H|R,\neg S) \cdot P(\neg S)$

$= 1 \cdot 0,1 + 1 \cdot 0,9 = 1$

$P(H) = P(H|R,S) \cdot P(R) \cdot P(S) + = 1 \cdot 0,2 \cdot 0,1 + = 0,02 +$

$+ P(H|R,\neg S) \cdot P(R) \cdot P(\neg S) + \quad 1 \cdot 0,2 \cdot 0,9 + \quad 0,18 +$

$+ P(H|\neg R,S) \cdot P(\neg R) \cdot P(S) + \quad 0,9 \cdot 0,8 \cdot 0,1 + \quad 0,072 +$

$+ P(H|\neg R,\neg S) \cdot P(\neg R) \cdot P(\neg S) = \quad 0,1 \cdot 0,8 \cdot 0,9 = \quad 0,072 = 0,344$

$P(R|H) = 1 \cdot 0,2 / 0,344 = \underline{0,58}$

$P(S|H) = P(H|S) \cdot P(S) / P(H)$

$P(H|S) = P(H|R,S) \cdot P(R) + P(H|\neg R,S) \cdot P(\neg R) = 1 \cdot 0,2 + 0,9 \cdot 0,8 = 0,92$

$P(S|H) = 0,92 \cdot 0,1 / 0,344 = \underline{0,267}$

$P(W|H) = P(W|H,R) P(R|H) + P(W|H,\neg R) \cdot P(\neg R|H) = P(W|R) P(R|H) + P(W|\neg R) P(\neg R|H) =$

$= 1 \cdot 0,58 + 0,2 \cdot 0,42 = \underline{0,664}$

$P(R|H,W) = \dfrac{P(R,W,H)}{P(W,H)} = \dfrac{0,2}{0,2288} = \underline{0,8741}$

$P(R,W,H) = P(R,W,H|S) P(S) + P(R,W,H|\neg S) P(\neg S) = P(R) \cdot P(W|R) P(H|RS) P(S) + P(R) \cdot P(W|R) P(H|R\neg S) P(\neg S)$

$= 0,2 \cdot 1 \cdot 1 \cdot 0,1 + 0,2 \cdot 1 \cdot 1 \cdot 0,9 = 0,2$

$P(W,H) = P(W,H|SR) P(S) \cdot P(R) + \quad P(W|R) P(H|SR) P(S) P(R) + \quad 1 \cdot 1 \cdot 0,1 \cdot 0,2 + \quad 0,02 +$

$P(W,H|S\neg R) P(S) \cdot P(\neg R) + \quad P(W|\neg R) P(H|S\neg R) P(S) P(\neg R) + \quad 0,2 \cdot 0,9 \cdot 0,1 \cdot 0,8 + \quad 0,0144 +$

$P(W,H|\neg SR) P(\neg S) \cdot P(R) + \quad P(W|R) P(H|\neg SR) P(\neg S) P(R) + \quad 1 \cdot 1 \cdot 0,9 \cdot 0,2 + \quad 0,18 +$

$P(W,H|\neg S\neg R) P(\neg S) P(\neg R) = \quad P(W|\neg R) P(H|\neg S\neg R) P(\neg S) P(\neg R) = \quad 0,2 \cdot 0,1 \cdot 0,9 \cdot 0,8 = \quad 0,0144 =$

$= \underline{0,22}$

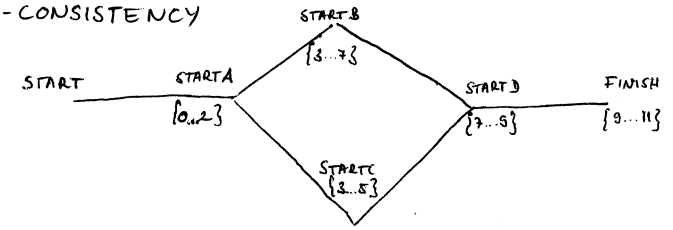
- $P(S|H,W) = \dfrac{P(S,W,H)}{P(W,H)} = \dfrac{0,0344}{0,2288} = \underline{0,15}$

$P(S,W,H) = P(S,W,H|R) P(R) + P(S,W,H|\neg R) P(\neg R) = P(S) \cdot P(W|R) \cdot P(H|RS) \cdot P(R) + P(S) P(W|\neg R) P(H|\neg RS) P($

$= 0,1 \cdot 1 \cdot 1 \cdot 0,2 + 0,1 \cdot 0,2 \cdot 0,9 \cdot 0,8 = 0,0344$

CSP

ARC-CONSISTENCY



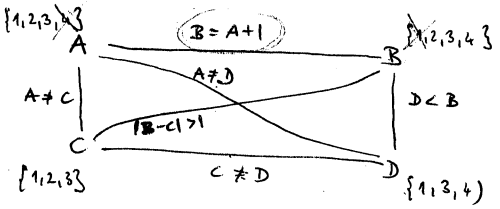
TASK	DURATION	PRECEDES
A	3	B, C
B	2	D
C	4	D
D	2	

C1	$start \leq startA$
C2	$startA+3 \leq B$
C3	$startA+3 \leq C$
C4	$startB+2 \leq D$
C5	$startC+4 \leq D$
C6	$startD+2 \leq FINISH$

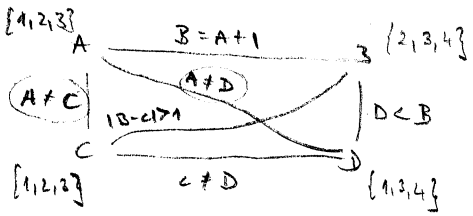
D_i : {0... 11}
D_{start} {0}

startA esetén sosem fordul elő {9,10,11} → töröljük D_{startA}-ból
startB —"— {0,1,2} → töröljük D_{startB}-ből

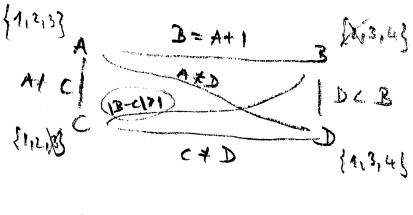
$C1 = \{(0,0),(0,1),(0,2)\dots(0,11)\}$
$C2 = \{(0,3),(0,4),\dots(0,11),(1,4),\dots(8,11)\}$
⋮

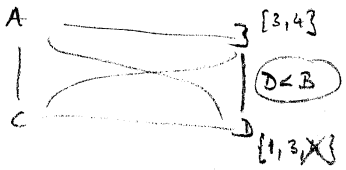
- 

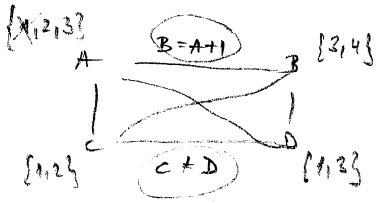
QUEUE = {c(A,B), c(A,C), c(A,D), c(B,C), c(B,D), c(C,D)}
hozzávenni: c(A,C), c(A,D), c(B,C), c(B,D)
⇓
QUEUE = {~~c(A,C)~~, ~~c(A,D)~~, c(B,C), c(B,D), c(C,D)}

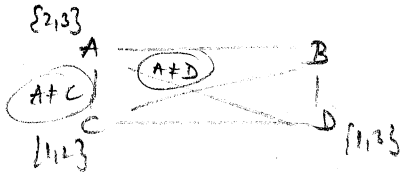


QUEUE = {~~c(B,C)~~, c(B,D), c(C,D)}
hozzávenni: c(A,B), c(A,C), c(B,D), c(C,D)
⇓
QUEUE = {~~c(B,D)~~, c(C,D), c(A,B), c(A,C)}
hozzávenni: c(A,D), c(C,D)
⇓
Q = {~~c(C,D)~~, ~~c(A,B)~~, c(A,C), c(A,D)}
hozzávenni: c(A,C), c(A,D)
⇓
Q = {~~c(A,C)~~, c(A,D)}



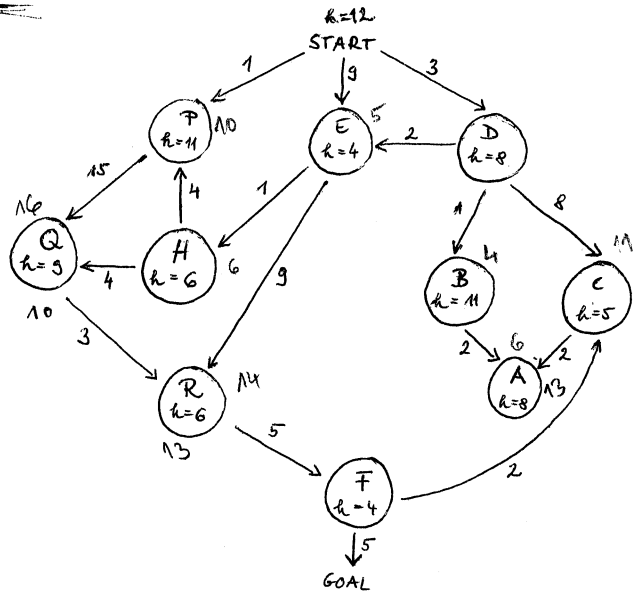






Q = empty

KERESÉSEK



SZÉLESSÉGI K.

START - D - E - P - B - C - H - R - Q - A - F - GOAL

MÉLYSÉGI K.

START - D - B - A - C - A - E - H - P - Q - R - F - C - A - GOAL

UNIFORM COST K.

START - P - D - B - E - ~~A~~ - H - Q - C - (A) - R - F - GOAL

S P D B E A H Q C R F G

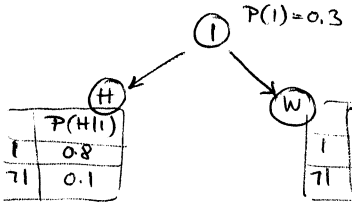
A* K.

S — (P) 11+1 = 12 X
(E) 4+9 = 13
(D) 8+3 = 11
D — (B) 4+11 = 15 X
C 3+8+5 = 16 X
X
(P) — Q 1+15+9 = 25 X
X
(E) — (H) 9+1+6 = 16
R 9+9+6 = 24
H — P 9+1+4+11 = 25 X
(Q) 9+1+4+9 = 23
Q — (R): 9+1+4+13+6 = 2
(F): 17+5+4 = 26
GOAL (F)

S T E H
S T — E - H - Q - R - F - GOAL
S D E H R P B A C Q R F G
S T D P

C — A 3+8+12+8 = 21 X

• ICY ROADS



	P(H|I)
I	0.8
¬I	0.1

	P(W|I)
I	0.8
¬I	0.1

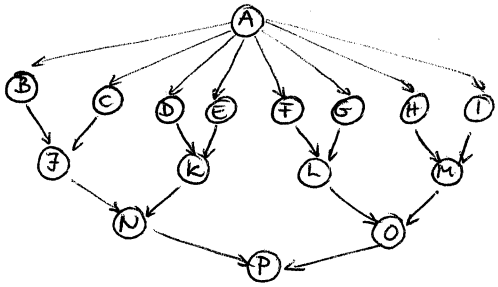
$$P(W) = P(W|I)\cdot P(I) + P(W|\neg I)P(\neg I) = 0{,}8 \cdot 0{,}3 + 0{,}1 \cdot 0{,}7 = 0{,}31$$

$$P(I|W) = P(W|I)\cdot P(I)/P(W) = 0{,}8 \cdot 0{,}3 / 0{,}31 = 0{,}77$$

$$P(H|W) = P(H|W,I)P(I|W) + P(H|W,\neg I)P(\neg I|W) =$$
$$= P(H|I)P(I|W) + P(H|\neg I)P(\neg I|W) =$$
$$= 0{,}8 \cdot 0{,}77 + 0{,}1 \cdot 0{,}23 = 0{,}639$$

$$P(H|W,\neg I) = P(H|\neg I) = 0{,}1$$

• VARIABLE ELIMINATION

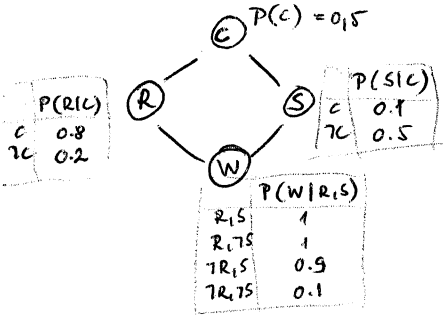


$$P(P) = \sum_{A:O} P(P|N,O)\,P(N|J,K)\,P(O|L,M)\,P(J|B,C)\,P(K|D,E)\,P(L|F,G)\,P(M|H,I)\,P(B|A)\,P(C|A)\,P(D|A)\,P(E|A)\,P(F|A)\,P(G|A)\,P(H|A)\,P($$

$$= \sum_{P} P(P|N,O) \sum_{L,M} P(O|L,M) \sum_{H,I} P(M|H,I) \sum_{A} P(H|A)P(I|A) \sum_{G} P(G|A) \sum_{F} P(L|F,G)P(F|A) \sum_{E} P(C|A) \sum_{K} \sum_{D} P(D|A)P(K|D,E) \sum_{J} P(N|J,K) \sum_{C} P(C|A) \sum_{B} P(J|B,C)P(B|A$$

$$f_{11}(N,O) \quad f_{10}(M,L) \quad f_9(M,I,H,I) \quad f_8(A,N,L) \quad f_7(A,N,L,G) \quad f_6(A,N) \quad f_5(A,N,L) \quad f_4(A,N,K,E) \quad f_3(A,N,K) \quad f_2(J,A) \quad f_1(J,C,A)$$

• MONTE CARLO



	P(R|C)
C	0.8
¬C	0.2

	P(S|C)
C	0.1
¬C	0.5

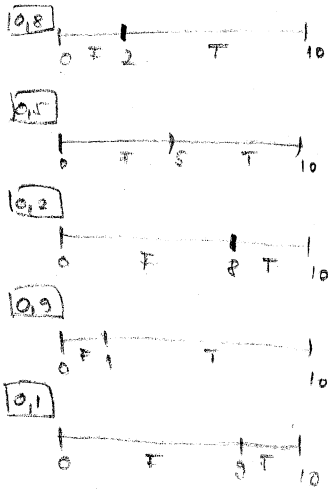
	P(W|R,S)
R,S	1
R,¬S	1
¬R,S	0.9
¬R,¬S	0.1

$$P(W|C) = P(C,W)/P(C)$$
$$P^*(W|C) = \#C,W \,/\, \#C$$

#	C	R	S	W
1.	~~F~~ (3)	~~F~~ (1)	~~F~~ (4)	~~F~~ (1)
2.	T (5)	T (9)	~~F~~ (2)	T (6)
3.	T	T	F	T
4.	T	T	T	T
	~~F~~	~~F~~	T	T
	T	T	~~F~~	T
	~~F~~	~~F~~	~~F~~	T
	~~F~~	~~F~~	T	T

$$P^*(W|C) = \frac{\#(C,W)}{\#C} = \frac{4}{4} = 1$$

3 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3



LOGIKA

$(P \leftrightarrow Q) \wedge (\neg P \wedge Q) \equiv (P \to Q) \wedge (Q \to P) \wedge \neg(P \vee \neg Q) \equiv (P \to Q) \wedge (Q \to P) \wedge \neg\neg(P \vee \neg Q) \equiv (P \to Q) \wedge \underbrace{(Q \to P) \wedge \neg(Q \to P)}_{\text{HAMIS}} \equiv \text{HAMIS}$

CNF-RE

- $(A \to B) \to C \equiv \neg(A \to B) \vee C \equiv \neg(\neg A \vee B) \vee C \equiv (A \wedge \neg B) \vee C = (A \vee C) \wedge (\neg B \vee C)$
- $(A \to B) \vee (B \to A) \equiv (\neg A \vee B) \vee (\neg B \vee A) \equiv$ IGAZ

$\neg(A \vee B) = \neg A \wedge \neg B$

$A \to B = \neg A \vee B$

- $\forall x \left[\left((L(x) \wedge K(x)) \to T(1) \right) \to (P(x) \to H(x)) \right] \equiv$

$\forall x \left[\neg\left(\neg(L(x) \wedge K(x)) \vee T(1) \right) \vee (\neg P(x) \vee H(x)) \right] \equiv \forall x \left[\neg\left((\neg L(x) \vee \neg K(x)) \vee T(1) \right) \vee (\neg P(x) \vee H(x)) \right] \equiv$

$\equiv \forall x \left(\neg(\neg L(x) \vee \neg K(x)) \wedge \neg T(x) \right) \vee \neg P(x) \vee H(x) \equiv \forall x \left((L(x) \wedge K(x)) \wedge \neg T(x) \right) \vee \neg P(x) \vee H(x) \equiv$

$\equiv \forall x \left(L(x) \wedge K(x) \wedge T(x) \right) \vee \neg P(x) \vee H(x) \equiv (L(x) \vee \neg P(x) \vee H(x)) \wedge (K(x) \vee \neg P(x) \vee H(x)) \wedge (T(x) \vee \neg P(x) \vee H(x))$

UNIFIKÁCIÓ

1. nice(Alice) - nice(Mary) → nice(Alice) nice(Mary)
2. sees(x, Alice) - sees(y, Alice) → {x/y}
3. sees(x, Alice) - sees(Mary, y) → {x/Mary, y/Alice}
4. x - child(Alice, x) → -
5. friends(x, y, Alice) ∧ father(sonof(Bob), Bob) - father(z, Bob) ∧ friends(Mary, z, u) → {x/Mary, u/Alice, z/sonof(Bob), y/z
6. R(F(y), x) - R(x, F(A)) → {x/F(y), y/A}
7. R(F(y), y, x) - R(x, F(A), F(v)) → {x/F(y), y/F(A), y/v} ?
8. F(G(w), H(w, J(x,w))) - F(G(v), H(u,v)) → ?
9. F(x, F(u, x)) - F(F(y, A), F(z, F(B, z))) → {x/F(y,A), u/z, x/F(B,z)}

NORMÁL FORMÁRA ALAKÍTÁS + REZOLÚCIÓ

Jack owns a dog.
Every dog owner is an animal lover.
No animal lover kills an animal.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

1. $\exists x\, Dog(x) \wedge owns(Jack, x)$
2. $\forall x (\exists y\, Dog(y) \wedge owns(x,y)) \to animalLover(x)$
3. $\forall x\, animalLover(x) \to \forall y\, animal(y) \to \neg kills(x,y)$
4. $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$
5. $cat(Tuna)$
6. $\forall x\, cat(x) \to animal(x)$

- CNF

(1.) $Dog(D) \wedge owns(Jack, D)$

(2.) $\forall x\, (\neg \exists y\, Dog(y) \wedge owns(x,y)) \vee animalLover(x) \equiv \forall x \forall y\, \neg(Dog(y) \wedge owns(x,y)) \vee animalLover(x) \equiv$
$\equiv \neg Dog(y) \vee \neg owns(x,y) \vee animalLover(x)$

(3.) $(\forall x\, \neg animalLover(x)) \vee (\forall y\, animal(y)) \to \neg kills(x,y)) \equiv \forall x\, \neg animalLover(x) \vee \forall y\, \neg animal(y) \vee \neg kills(x,y) \equiv$
$\equiv \neg animalLover(x) \vee \neg animal(y) \vee \neg kills(x,y)$

(4.) $kills(Jack, Tuna) \vee kills(Curiosity, Tuna)$

(5.) $cat(Tuna)$

(6.) $\neg cat(x) \vee animal(x)$

(+) $\neg kills(Curiosity, Tuna)$ → a bizonyítandó állítás ellentettje

REZOLÚCIÓ

(9): (6) + [x/Tuna] => $\neg cat(Tuna) \vee animal(Tuna)$

(11): (8) + (5) => $animal(Tuna)$

(12): (2) + [x/Jack, y/D] => $\neg Dog(D) \vee \neg owns(Jack, D) \vee animalLover(Jack)$

(10): (12) + (1) => $animalLover(Jack)$

(13): (3) + {x/Jack, y/Tuna} => $\neg animalLover(Jack) \vee \neg animal(Tuna) \vee \neg kills(Jack, Tuna)$

(14): (13) + (11) => $\neg kills(Jack, Tuna)$

(15): (14) + (+) : ellentmondás

TERVEZÉS

- PARTIALLY ORDERED PLAN

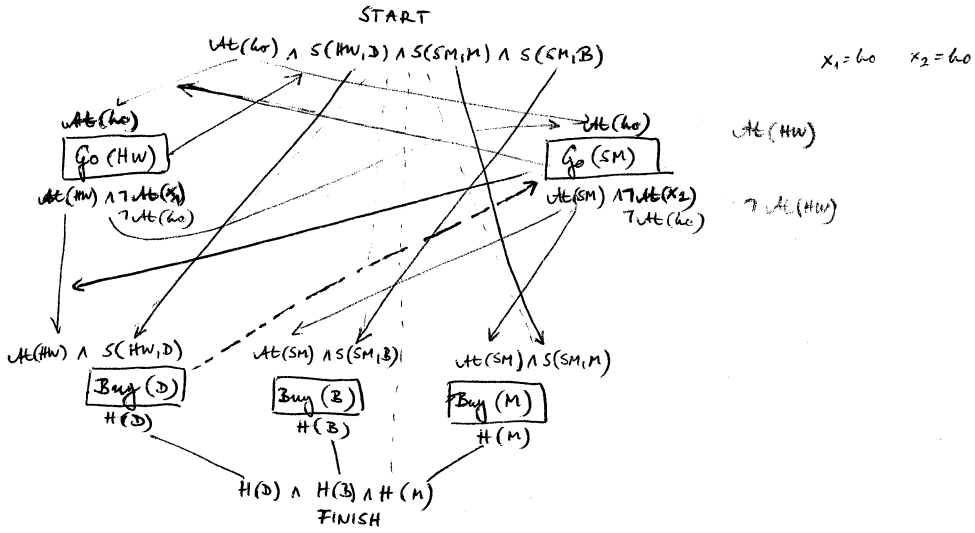
STRIPS REPRESENTATION:

- ACTION

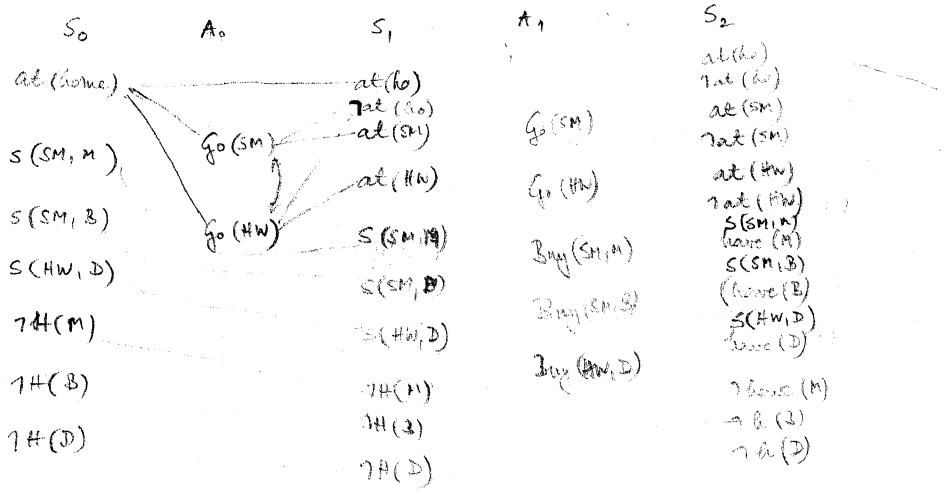
Buy (x, store)
- Pre: At(store), Sells(store, x)
- Eff: Have(x)

Go (x, y)
- Pre: At(x)
- Eff: At(y), ¬At(x)

- GOAL
  Have(milk) ∧ Have(banana) ∧ Have(drill)
- START
  At(home) ∧ Sells(SM, milk) ∧ Sells(SM, banana) ∧ Sells(HW, drill)



- GRAPH PLAN



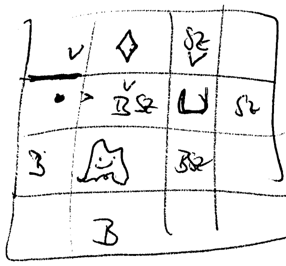
1.) Konv —> CNF

a.) $(A \to B) \to C$

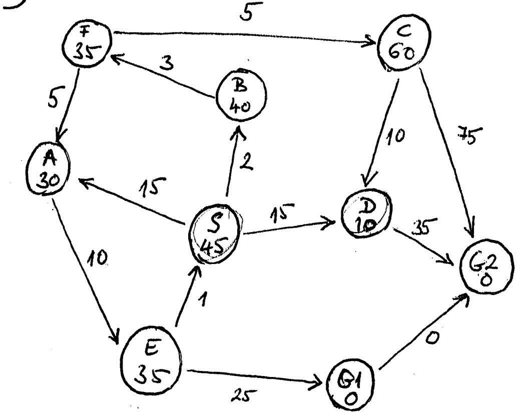
b.) $(A \to B) \vee (B \to A)$

c.) $\forall x \left[\left((L(x) \wedge K(x)) \to T(1) \right) \to \left(P(x) \to H(x) \right) \right]$

$A \to B \equiv \neg A \vee B$



2.) A^* : S —> Gx



S —> A 15 + 30 = 45
B 2 + 40 = 42
D 15 + 10 = 25 —> G2 15 + 35 + 0 = 50 ! 42 < 50 =>

B —> F 2 + 3 + 35 = 40
F —> A 5 + 5 + 30 = 40
C 5 + 5 + 60 = 70
A —> E 10 + 10 + 35 = 55 ! 55>

Nem jó heurisztika
pl. E-ben túlbecsüli a heurisztika

1.) a. $\neg(\neg A \vee B) \vee C \equiv (A \wedge \neg B) \vee C \equiv (A \vee C) \wedge (\neg B \vee C)$

b. $(\neg A \vee B) \vee (\neg B \vee A) \equiv 1$

c. ~~$\forall x \neg[(\neg L(x) \vee K(x) \vee T(x)$~~

$\forall x \{ \neg[\neg(L(x) \wedge K(x)) \vee T(x)] \vee (\neg P(x) \vee H(x)) \} \equiv \forall x \neg[(\neg L(x) \vee \neg K(x)) \vee T(x)] \vee \neg P(x) \vee H(x) \equiv$

$\equiv \forall x [\neg(\neg L(x) \vee \neg K(x)) \wedge \neg T(x)] \vee \neg P(x) \vee H(x) \equiv \forall x [(L(x) \wedge K(x)) \wedge \neg T(x)] \vee \neg P(x) \vee H(x) \equiv$

$\equiv \forall x [L(x) \wedge K(x) \wedge \neg T(x)] \vee \neg P(x) \vee H(x) \equiv (L(x) \vee \neg P(x) \vee H(x)) \wedge (K(x) \vee \neg P(x) \vee H(x)) \wedge (\neg T(x) \vee \neg P(x) \vee H(x))$

3.) Formalizáció

a.) Valaki szereti Jánost

b.) Minden a Vértesben lévő hegycsúcsra igaz, hogy van nála magasabb csúcs a Mátrában.

c.) Csak a halott földönkívüli a jó földönkívüli.

d.) Nincs olyan diák, aki okosabb önmagánál.

e.) Minden diák jó jegyet kap, ha tanul a ZH-ra.

~~a.)~~ loves(x, János)

b.) ~~$\forall y$ in(x, Mátra) magasság(x) > magasság(y) in(y, Vértes)~~

$\forall y$ in(y, Vértes) $\exists x$ in(x, Mátra) magasság(x) > magasság(y)

a.) $\exists x$: szeret(x, János)

b.) $\forall x$ (csúcs(x) ∧ Vértesben(x)) —> $\exists y$ (csúcs(y) ∧ Mátrában(y) ∧ magasabb(y, x))

c.) $\neg \exists x$ FK(x) ∧ ¬H(x) ∧ Jó(x)

$\forall x$ (FK(x) ∧ Jó(x)) —> H(x)

d.) $\forall x$ diák(x) —> ¬okosabb(x, x)

e.) $\forall x$ diák(x) ∧ tanul(x) —> jójegyetkap(x)

4, a) $\forall x.\ h(x) \to g(x)$ I
$\forall x.\ f(x) \to g(x)$ I
$\exists x.\ f(x) \wedge h(x)$ H

interpretáció $\mathcal{U} = \{A, B\}$
$I(f) = \{A\}$
$I(g) = \{A, B\}$
$I(h) = \{B\}$

b) $\forall x\ \exists y\ f(x,y)$ I
$\exists y \forall x.\ f(x,y)$ H

$\mathcal{U} = \{A, B\}$
$I_1(f) = \{\langle A,A\rangle, \langle B,B\rangle\}$
$I_2(f) = \{\langle A,B\rangle, \langle B,A\rangle\}$

5,
Szín(Kék, Sárga)
Szín(Kék, Sárga)
Szín(Kalap(Fiúnév), Kék)
$F(x, F(u, x))$

Szín(x, y)
Szín(x, x)
Szín(Kalap(y), y)
$F(F(y, A), F(z, F(B, z)))$

Egyesítés $x = $ Kék $y = $ Sárga
X
+
$y = B$ $z = A$
$F(F(B,A), F(A, F(B,A)))$

Ágens model - mik változhatnak Környezet 5

Keresések - algoritmusok, összehasonlításuk
2 csoport - tárolja a csomópontokat
- eldobja a kibontott csomópontokat

Informált keresések - eff. elágazási tényező
heurisztika

Minimax, α-β

Rezolúció

Tervezés - részben rendezett - melyik keresési modellt alkalmazzák?
- gráf alapú

Kényszerkielégítési problémák

Bizonytalanság kezelés - Bayesian Networks
láncszabály, feltételes valószínűség, teljes valószínűség, conditioning ...

Mintavételezés (Monte-Carlo)

$F(x, F(u, x))$ $\quad F(\underbrace{F(y, A)}_{x}, \underbrace{F(z, F(B, z))}_{u})$

$F(F(y, z), F(z, F(y, z)))$

$y = B$
$A = z$
$z = u$

$B = y$
$A = z$

Játék példa

$(P \leftrightarrow Q) \wedge (\neg P \wedge Q) = (P \to Q) \wedge (Q \to P) \wedge (\neg P \wedge Q) = (P \to Q) \wedge (Q \to P) \wedge \neg(P \vee \neg Q) = F$

$Q \to P$

F

$$A \wedge B = \neg(\neg A \vee \neg B)$$

$x/F(y) \quad y/F(A) \quad x/F(v)$

$v/f(x,u)$

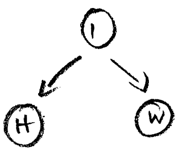
w/v

$w/v \quad v/u$

$x/F(y) \quad y/F(A) \quad v/F(A)$

$A \to B = \neg A \vee B$

$H \to I \quad H$



$P(H|W) = P(H|W,I) \cdot P(I|W) + P(H|W,\neg I) P(\neg I|W)$

$$P(A \wedge B | C) = P(A|C) \cdot P(B|C)$$

egyz.

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

tels.

$$\frac{(A \vee B) \wedge (\neg B \vee C)}{A \vee C}$$

$(P \to Q) \equiv \neg Q \to \neg P$

$P \leftrightarrow Q \equiv ((P \wedge Q) \vee (\neg P \wedge \neg Q))$

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$$

$P(B|A) = P(A,B) / P(A)$

$$P(R|W,H) = \frac{P(R,W,H)}{P(W,H)} = \frac{0,2}{0,12288} = 0,8741$$

$$P(R,W,H) = P(R,W,H|S)P(S) + P(R,W,H|\neg S)P(\neg S) = P(R)\cdot P(W|R)\cdot P(H|RS)\cdot P(S) + P(R)\cdot P(W|R)\cdot P(H|R\neg S)\cdot P(\neg S) = 0,2$$

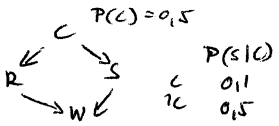
$$P(W,H) = P(WH|SR)\cdot P(S)\cdot P(R) + P(WH|S\neg R)\cdot P(S)\cdot P(\neg R) + P(WH|\neg S R)\cdot P(\neg S)\cdot P(R) + P(WH|\neg S\neg R)\cdot P(\neg S)\cdot P(\neg R) = 0,12288$$

$$P(S|WH) = \frac{P(S,W,H)}{P(W,H)} = \frac{0,0344}{0,12288} = 0,1503$$

$$P(S,W,H) = P(S,W,H|R)P(R) + P(S,W,H|\neg R)P(\neg R) \overset{?}{=} P(S)\cdot P(W|S)\cdot P(H|RS)\cdot P(R) + P(S)\cdot P(W|S)\cdot P(H|S\neg R)\cdot P(\neg R) = 0,1$$

MONTE CARLO

	P(R|C)
C	0,8
¬C	0,2



	P(S|C)
C	0,1
¬C	0,5

R,S	P(W|R,S)
R,S	1,0
R,¬S	1,0
¬R,S	0,9
¬R,¬S	0,0

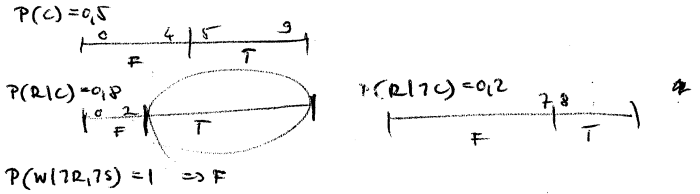
$P(W|C) = P(C,W)/P(C)$

$P^*(W|C) = \#C,W / \#C$

#	C	R	S	W
1	F	F	F	F
2	T	T	F	T
3	T	T	F	T
4	T	T	T	T

	C	R	S	W
1	T	T	F	T
2	T	T	F	T
3	F	F	T	T
4	T	T	F	T
5	F	F	T	T
6	F	F	T	T

?: 0,8 → 0 1 | 2 3 | 9 → "sorsolás" ⇒



$$P(W|C) \approx P^*(W|C) = \frac{\#(C,W)}{\#C} = \frac{4}{4} = 1$$

ICE

776265

REASONING WITH UNCERTAINTY

Axioms of probability: $P(e): e \to [0,1]$

$P(T) = 1 \quad P(F) = 0$

$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$

Conditional probability: $P(A \wedge B) = P(A)\,P(B|A)$

$P(B|A) = P(A \wedge B) / P(A)$

Independence: $P(A \wedge B) = P(A)P(B) \qquad P(B|A) = P(B)$

Conditioning: $P(A) = P(A \wedge B) + P(A \wedge \neg B) =$

$= P(A|B)P(B) + P(A|\neg B)P(\neg B)$

BAYESIAN NETWORKS

Bayes rule: $P(B|A) = \dfrac{P(A|B)P(B)}{P(A)}$

Conditional independence: $P(A \wedge B|C) = P(A|C) \cdot P(B|C)$

$P(A|B,C) = P(A|C) \qquad P(B|A,C) = P(B|C)$

D-separation: if A & B are d-separated given evidence e $\Rightarrow P(A|e) = P(A|B,e)$

Chain rule: variables: $V_1 \dots V_n$ values: $v_1 \dots v_n$

$P(V_1 = v_1, V_2 = v_2 \dots V_n = v_n) = \prod_{i=1}^{n} P(V_i = v_i \mid parents(V_i))$

MACHINE LEARNING

- Information theory

  Entropy: $H(S) = \sum_{i=1}^{n} - P(v_i) \log_2 P(v_i)$

  Gain: $G(S,A) = H(S) - \sum_{v \in V} \frac{|S_v|}{|S|} H(S_v)$

- Inductive logic programming
  - prior satisfiability $\forall e \in E^- \ (B \not\models e)$
  - prior necessity $\exists e \in E^+ \ (B \not\models e)$
  - posterior s. $\forall e \in E^+ \ (B \wedge H \models e)$
  - posterior n. $\forall e \in E^- \ (B \wedge H \not\models e)$
  - Inverting resolution rules
    - absorption $\dfrac{q \leftarrow A \ \& \ p \leftarrow A,B}{q \leftarrow A \ \& \ p \leftarrow q,B}$
    - identification $\dfrac{p \leftarrow A,B \ \& \ p \leftarrow A,q}{q \leftarrow B \ \& \ p \leftarrow A,q}$
    - intra construction $\dfrac{p \leftarrow A,B \ \& \ p \leftarrow A,C}{q \leftarrow B \ \& \ p \leftarrow A,q \ \& \ q \leftarrow C}$
    - inter construction $\dfrac{p \leftarrow A,B \ \& \ q \leftarrow A,C}{p \leftarrow r,B \ \& \ r \leftarrow A \ \& \ q \leftarrow r,C}$

- PAC learning

  $P(h_{bad}$ is consistent with m examples$) \leq (1-\varepsilon)^m$

  $P(H_{bad}$ contains a consistent hyp$) \leq |H_{bad}|(1-\varepsilon)^m \leq |H|(1-\varepsilon)^m$

  δ: upper bound

  $N = m(\varepsilon,\delta) \geq \frac{1}{\varepsilon}\left(\ln\frac{1}{\delta} + \ln|H|\right)$

PROPOSITIONAL LOGIC

- Semantic rules

  $\models_i T$ for all i

  $\not\models_i F$ for all i

  $\models_i \neg\varphi$ iff $\not\models_i \varphi$

  $\models_i \varphi \wedge \psi$ iff $\models_i \varphi$ and $\models_i \psi$

  $\models_i \varphi \vee \psi$ iff $\models_i \varphi$ or $\models_i \psi$

  $\models_i P$ iff $i(P) = T$

- Entailment theorem

  $KB \models \varphi$ iff $\models (KB \to \varphi)$

- Equivalence rules (commutativity, associativity, distributivity, double negation)

  de Morgan's laws: $\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$

  $\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$

  Contraposition: $(P \to Q) \equiv (\neg Q \to \neg P)$

  Other equivalences: $P \to Q \equiv \neg P \vee Q$

  $P \leftrightarrow Q \equiv (P \to Q) \wedge (Q \to P)$

  $P \leftrightarrow Q \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$

  $P \wedge \neg P \equiv F$

  $P \vee \neg P \equiv T$

  Modus Ponens: $\dfrac{A \to B, A}{B}$

  and elimination: $\dfrac{A_1, A_2 \dots A_n}{A_i}$ and introduction: $\dfrac{A_1, A_2 \dots A_n}{A_1 \wedge A_2 \wedge \dots \wedge A_n}$

  or introduction: $\dfrac{A_i}{A_1 \vee A_2 \vee \dots \vee A_n}$ unit resolution: $\dfrac{(A \vee B) \wedge \neg B}{A}$

FIRST ORDER PREDICATE LOGIC

- First order implication rules

  Universal elimination $\dfrac{\forall v \alpha}{Subst(\{v/g\}, \alpha)}$

  Existential elimination $\dfrac{\exists v \alpha}{Subst(\{v/k\}, \alpha)}$

  Existential introduction $\dfrac{\alpha}{\exists v \ Subst(\{g/v\}, \alpha)}$

INFERENCE IN FOPL

- Resolution rules

  unit r.r. $\dfrac{(A \vee B) \wedge \neg B}{A}$ full r.r. $\dfrac{(A \vee B), (\neg B \vee C)}{A \vee C}$ $\dfrac{(\neg A \to B)\ (B \to C)}{\neg A \to C}$

  Generalised r.r.

  2 sentences: $p_1 \vee p_2 \vee \dots p_m$, $q_1 \vee q_2 \dots q_n$ $\quad unify(p_j, \neg q_k) = \theta$

  $\dfrac{p_1 \vee p_2 \vee \dots p_j \dots \vee p_m \quad q_1 \vee q_2 \vee \dots q_k \dots \vee q_n}{Subst(\theta, (p_1 \vee \dots p_{j-1} \vee p_{j+1} \dots \vee p_m \vee q_1 \vee q_2 \dots q_{k-1} \vee q_{k+1} \dots q_n))}$