I. Bevezetés

1. AZ MI FOGALMA

1956 nyár. Darthmouth College-i konferencia

Kezdeti cél:

Az emberi gondolkodás számítógép segítségével történő reprodukálása.

Első szakasz (60-as évek)

- Eredmények: kétszemélyes játékok (dáma, sakk), beszélgető program (ELIZA,1966)
- Módszerek, eszközök: GPS, rezolúció (1966), LISP(1958), mesterséges neuronhálózatok (1969), evolúciós algoritmusok (1959), Turing teszt
- Kudarcok: DOCTOR-PARRY, nyelvi fordítók, kombinatorikus robbanás

ELIZA

- Illesztési szabályok
  - **<a>** *ön* **<b>** *engem* **<c>.**
  - *Miért gondolja, hogy ön* **<a>** *én* **<b>** **<c>**?
    - Úgy érzem, hogy *ön* mostanában *engem* un.
    - *Miért gondolja, hogy ön* úgy érzi, hogy *én* mostanában unom?
- Emlékezési szabályok
- Folytatási szabályok

Második szakasz (70-es évek)

- Eredmények: SHRDLU (1972), BACON, AM, EURISKO
- Módszerek, eszközök: Prolog, heurisztikus keresési technikák, tudásábrázolási módszerek (kognitív modellek)
- Kudarcok: MI fejlődési trendje, meseíró program

Harmadik szakasz (80-as évek)

- Eredmények: DENDRAL (1969-78), MYCIN(1976), PROSPECTOR(1979), XCON (1982)
- Módszerek, eszközök: tudásalapú szakértő rendszerek, shell-ek, módszertanok, nem klasszikus logikák, bizonytalanság kezelése
- Kudarcok: rendszerek elkészítése lassabb, mint a gyorsan változó programozási környezet

Negyedik szakasz (90-as évektől)

- Eredmények: logisztika, űrkutatás, Deep Blue, döntés támogató rendszerek, nyelvi fordítók, robotika (beszélgetés, gépi látás, tervgenerálás, gépi tanulás)
- Módszerek, eszközök: elosztott tudás reprezentálása (mesterséges neuron háló, evolúciós algoritmus, ágens szemlélet), döntéselmélet (valószínűségi hálók), beszédfelismerés (rejtett Markov modellek)

MI helye

- Az MI az emberi gondolkodás számítógépes reprodukálása szempontjából hasznos elveket, módszereket, technikákat kutatja, rendszerezi, fejleszti
- Nem az emberi gondolkodás számítógépes modelljét, hanem egy feladat minél jobb minőségű számítógépes megoldását keresi

Az MI az informatikának a gondolkodási-tudományos előőrse.
Vámos Tibor

MI tárgya

- Azon feladatok számítógépes megoldása, amelyek
  - megoldása nehéz, komplex ismereteket igényel
  - az embertől is kellő szakértelmet, kreativitást és intuíciót kíván (szemléletmód váltások)
  - megoldásukban ma többnyire az ember a jobb

- a probléma tere (lehetséges válaszok száma) nagy, az összes lehetőség kipróbálása szisztematikus úton nem lehetséges,
- a válasz sokszor elemi tevékenységek sorozatával írható le,
- amely előre nem rögzíthető, hanem több lehetséges sorozat közül kell kiválasztani.
- irányított keresésre van szükség.

2. PROBLÉMA MODELLEZÉS

- Problémareprezentációs modellek:
  - Állapottér reprezentáció
  - Probléma redukció, dekompozíció
  - Logikai reprezentáció
  - Strukturált objektum alapú reprezentáció
  - Elosztott reprezentáció

Útkeresési probléma

- Általában a problématérnek megfeleltethető egy irányított gráf, ahol a csúcsok a lehetséges válaszok, az élek az ezek közötti szomszédsági kapcsolatok. Ebben a gráfban a kiinduló válasznak megfelelő csúcstól indulva egy helyes választ reprezentáló csúcsot keresünk.
- Speciálisan, amikor a lehetséges válaszokat elemi lépések sorozataként adjuk meg, akkor ezek ábrázolhatók egy úttal, amelyek egy közös kezdő csúcsból indulnak. Ebben a gráfban kell olyan utat keresünk, amelyik egy helyes választ reprezentál.

Gráfreprezentáció fogalma

- Egy útkeresési probléma gráfreprezentációja egy (R,s,T) hármas, amelyben
  - $R=(N,A,c)$ δ-gráf az un. reprezentációs gráf,
  - az $s \in N$ startcsúcs a kiinduló pont,
  - a $T \subseteq N$ halmazbeli célcsúcsok.
- A feladat megoldása:
  - egy $t \in T$ cél megtalálása
  - egy $s \rightarrow T$, esetleg egy $s^* \rightarrow T$ optimális út megtalálása

Gráf fogalmak 1.

csúcsok, ir. élek	N, $A \subseteq N \times N$ (számosság)
él n-ből m-be	$(n,m) \in A$ $(n,m \in N)$
n utódai, szülei	$\Gamma(n)$, $\Pi(n)$, $\pi(n)$
irányított gráf	$R=(N,A)$
σ-tulajdonság	$\lvert\{(n,m) \in A \mid m \in N\}\rvert < \sigma \ \forall n \in N$
élköltség	$c: A \rightarrow \mathbf{R}$, $c(n,m)$ $\forall (n,m) \in A$
δ-tulajdonság	$c(n,m) \geq \delta > 0 \quad \forall (n,m) \in A$
δ-gráf	δ, σ -tulajdonságú élsúlyozott irányított gráf

Gráf fogalmak 2.

irányított út	$\alpha=(n,n_1),(n_1,n_2),\ldots,(n_{k-1},m)$ $(n,n_1,n_2,\ldots,n_{k-1},m)$, $n^{\alpha} \rightarrow m$, $n \rightarrow m$
n-ből kiinduló utak	$\{n \rightarrow m\}$, $\{n \rightarrow M\}$
ir. út hossza	$\lvert\alpha\rvert$
ir. út költsége	$c^{\alpha}(n,m) := \Sigma_i\, c(n_{i-1},n_i)$
opt. költség	$c^*(n,m) := \min_{\alpha \in \{n \rightarrow m\}} c^{\alpha}(n,m)$
opt. költségű út	$n^* \rightarrow m$, $n^* \rightarrow M$

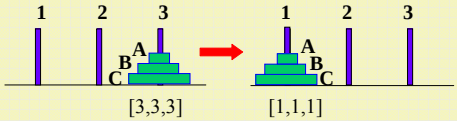
Állapottér-reprezentáció

- Ez egy széles körben (nemcsak a MI-ban) használt modell, amely segítségével egy problémát specifikálhatunk.
- Jellegzetessége, hogy a problémák megoldását műveletek sorozataként fogalmazza meg, ennél fogva az állapottér-reprezentáció alkalmazása egy útkeresési problémát (többnyire speciális útkeresési problémát) ad meg.

Állapottér-reprezentáció modellje

- Állapottér (domináns típusérték-halmaz)
  - invariáns
- Műveletek (elemi lépés az állapottérben)
  - előfeltétel, hatás
- Kezdő állapot(ok) vagy előfeltétel
- Célállapot(ok) vagy utófeltétel

Hanoi tornyai probléma



[3,3,3] [1,1,1]

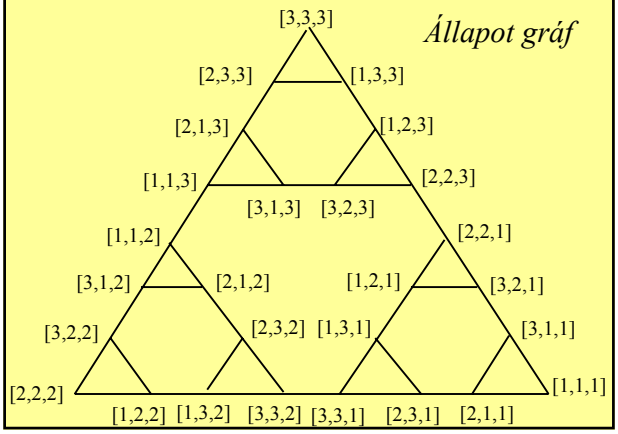
Állapottér: ***Állás***=*vektor*([*A,B,C*];{*1,2,3*})

Művelet: ***Rak***(*honnan, hová*):***Állás*** → ***Állás*** (*a*:***Állás***)

HA *a*-ban a *honnan* nem üres, és
a *hová* üres vagy a *hová* felső korongja nagyobb, mint *honnan* felső korongja

AKKOR *a[honnan felső korongja]* ← *hová*

Állapot gráf



3. KERESÉS

- A keresés egy olyan algoritmus, amely
  - tárolja az addig feltárt információ egy részét
  - felismeri a tárolt információból azt, ha elérte a célját
  - látja az adott pillanatban elvégezhető alternatív lépéseket
  - dönt arról, hogy melyik lépést hajtsa végre az adott pillanatban
  - egy lépést végrehajtásával módosítja a tárolt információt

Kereső rendszer

Procedure *KR*
1. ADAT ← *kezdeti érték*
2. while ¬ *terminálási feltétel*(ADAT) loop
3. select SZ from *alkalmazható* szabályok
4. ADAT ← SZ(ADAT)
5. endloop

end

A reprezentációs gráf feletti keresés, amely a gráf egy részét (*ADAT*) látja, azt változtatja meg (*SZ*) az általa meghatározott (*select*) módon.

A KR részei

globális munkaterület — ADAT
a keresés során megszerzett és megőrzött (deklaratív) ismeret
kezdeti érték, terminálási feltétel

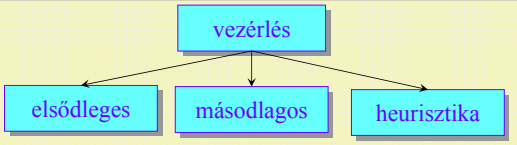
kereső rendszer szabályai — SZ
globális munkaterületet megváltoztató operátorok (procedurális ismeret)
hatás, értelmezési tartomány

vezérlési stratégia — select
végrehajtható szabályt kiválasztó *általános elv* + a konkrét feladattól származó *vezérlési ismeret*

Keresés és a problématér

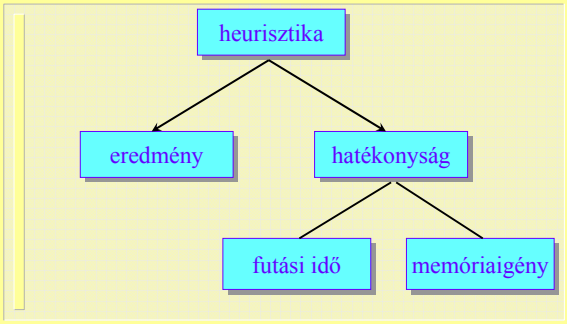
- Egy útkeresés hatékonysága a reprezentációs gráf (itt állapot gráf) bonyolultságát (csúcsok, élek számát, körök gyakoriságát, körök hosszát) meghatározó reprezentáción múlik.
- A reprezentációs gráfnak csak a startcsúcsból elérhető része érdekel bennünket.
- Ha egy keresés nem végez körfigyelést, csak a megelőző csúcsba történő visszalépést zárja ki, akkor tulajdonképpen nem az eredeti gráfon, hanem annak úgynevezett fává egyenesített változatában dolgozik.

Vezérlési vagy keresési stratégia

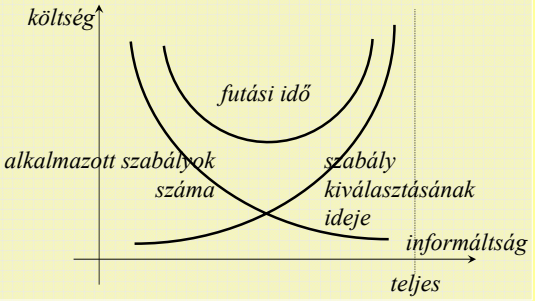


elsődleges	másodlagos	heurisztika
független a feladattól	független a konkrét feladattól, de annak reprezentációs modelljére támaszkodik	Konkrét feladatból származó a reprezentációban nem rögzített ismeretek

A heurisztika hatása a KR működésére

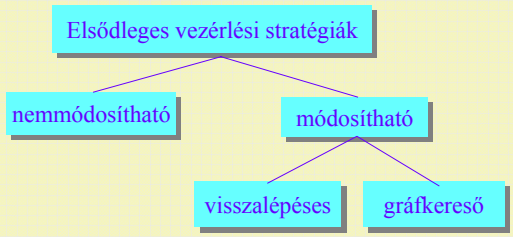


A KR futási ideje



II. Keresések

Elsődleges stratégiák osztályozása



1. Nemmódosítható stratégia

- A keresés során hozott döntések visszavonhatatlanok
- Nincs lehetőség egy korábbi döntési ponthoz visszatérni, és másik döntést hozni

Lokális keresések

- A globális munkaterületen tárolt egyetlen csúcsot (lehetséges választ) annak környezetéből vett lehetőleg jobb csúccsal cserél le.
- A jobbság eldöntéséhez célfüggvényt használ
- Alkalmazás:
  - Adott tulajdonságú elem keresése
  - Függvény optimumának keresése

Hegymászó algoritmus

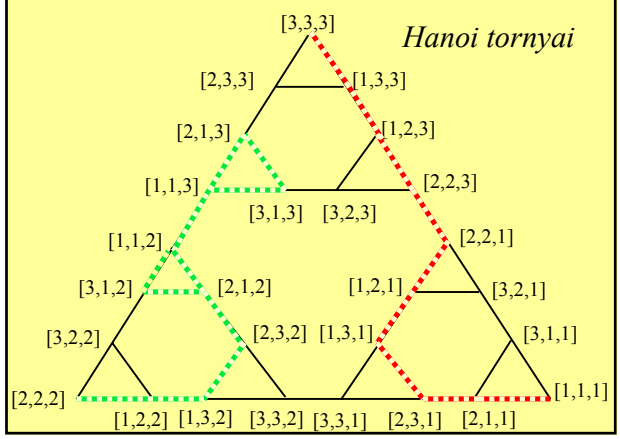
- Lokális optimumban megengedi az aktuális csúcsnál rosszabb értékű legjobb szomszédra lépést, és kizárja a szülő csúcsra való visszalépést.

Procedure ***Hegymászó módszer***

1. $n \leftarrow$ *startcsúcs*
2. while *n nem célcsúcs* loop
3. $n \leftarrow \text{opt}_f(\Gamma(n) \backslash \pi(n))$ // üres halmazra kilép
4. endloop

end

Hanoi tornyai



Megjegyzés

- Hátrányok: Nem végez körfigyelést, ezért
  - lokális optimum hely körül eltévedhet
  - ekvidisztans felületen eltévedhet
  - zsákutcába beragad
  - csak erős heurisztika esetén alkalmazható
- A baj okai:
  - Túl kicsi az algoritmus memóriája
  - Túl erős az alkalmazott mohó stratégia

Tabu-keresés

- Az n aktuális csúcson kívül nyilvántartjuk még
  - az eddig legjobbnak bizonyult n^* csúcsot és
  - az utóbbi néhány aktuális csúcsot; ez a tabu halmaz
- Minden lépésben
  - kiválasztjuk a legjobb csúcsot az aktuális csúcs környezetéből (kivéve ebből a tabu csúcsokat)
  - ha ez jobb, mint az n^*, akkor azt lecseréljük
  - frissítjük a tabu halmazt
- Terminálási feltételek:
  - ha a célfüggvény az n^*-ban optimális
  - ha az n nem vagy az n^* sokáig nem változik.

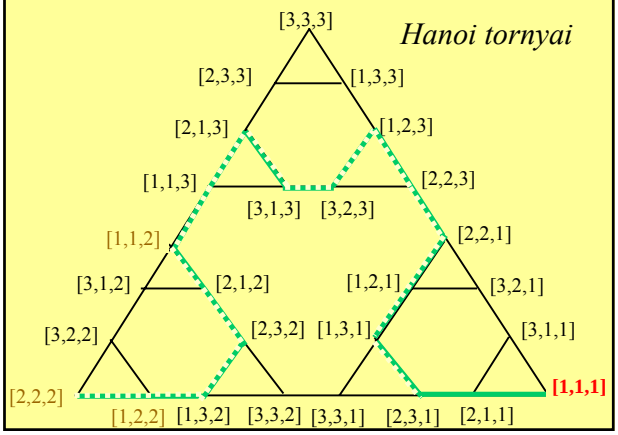
Tabu kerersés algoritmusa

Procedure ***Tabu keresés***

1. $n, n^*, Tabu \leftarrow$ *startcsúcs, startcsúcs,* $\varnothing$
2. while *not terminálási feltétel* (n^* *nem célcsúcs*) loop
3. $n \leftarrow \text{opt}_f(\ \Gamma(n) \backslash Tabu\ )$ // üres halmazra kilép
4. $Tabu \leftarrow$ Módosít($n, Tabu$)
5. if $f(n) > f(n^*)$ then $n^* \leftarrow n$
6. endloop

end

Hanoi tornyai



Megjegyzés

- Hátrányok:
  - A tabu méretét kísérletezéssel kell belőni
  - beszorulhat a keresés

Szimulált hűtés

- A következő csúcs választása véletlenszerű.
- Ha a kiválasztott r csúcs célfüggvény-értéke rosszabb (itt nagyobb), mint az aktuális n csúcsé, akkor annak újcsúcsként való elfogadásának valószínűsége fordítottan arányos $f(r)$ és $f(n)$ különbséggel.

ha $f(r) \le f(n)$ **vagy**

$f(r) > f(n)$ **és** $e^{f(n) - f(r)} > random\ [0,1]$

akkor $n \leftarrow r$

Szimulált hűtés algoritmusa

Procedure ***Szimulált hűtés***

1. $n \leftarrow$ *startcsúcs;* $k \leftarrow 1$
2. while *not terminálási feltétel* (*n nem célcsúcs*) loop
3. for $i = 1 .. L_k$ loop
4. $r \leftarrow$ select($\Gamma(n) \backslash \pi(n)$)
5. if $f(r) \le f(n)$ *or* $f(r) > f(n)$ *and* $e^{\frac{f(n) - f(r)}{T_k}} >$ *random*
6. then $n \leftarrow r$
7. endloop
8. endloop

end

Hűtési ütemterv

- A T csökkentésével csökken egy rosszabb új csúcs elfogadásának valószínűsége.
- Adjunk ütemtervet a T változására
- Az ütemterv elemei:
  - Kezdeti hőmérséklet: T_0
  - Hőmérséklet csökkentésének menete és egy hőmérséklet melletti szakasz hossza:
    (T_k , L_k) $k= 1,2, \ldots$
    ahol minden T_k érték L_k lépésen keresztül van érvényben.

Szimulált hűtés ereje

- A szimulált hűtés algoritmusa (aszimptotikusan) egy optimális megoldáshoz konvergál, ha
  - az algoritmussal bármely csúcsból bármely csúcs véges lépésen belül elérhető (erősen összefüggés, csúcs környezet)
- Ahhoz azonban, hogy véges lépésen belül is egy elég jó megoldást találjunk, megfelelő hűtési ütemtervet kell találni.

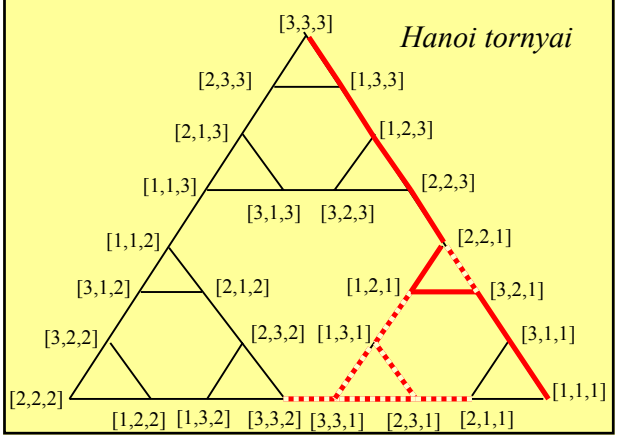
2. Visszalépéses stratégia

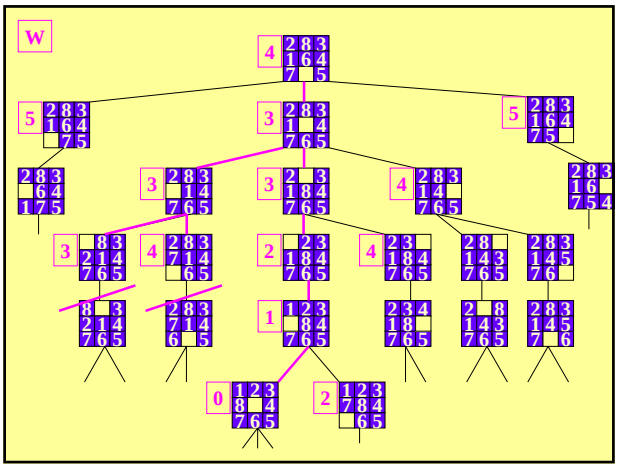
- A visszalépéses kereső rendszer olyan KR, amely
  - globális munkaterülete:
    - út a startcsúcsból az aktuális csúcsba (ezen kívül a még ki nem próbált élek nyilvántartása)
      - Kezdetben a startcsúcsot tartalmazó nulla hosszúságú út
      - terminálás célcsúccsal vagy startcsúcsból való visszalépéssel
  - keresés szabályai:
    - a nyilvántartott út végéhez egy új (ki nem próbált) él hozzáfűzése, vagy az legutolsó él törlése (visszalépés szabálya)
  - vezérlés stratégiája a visszalépés szabályát csak a legvégső esetben alkalmazza

Visszalépés feltételei az aktuális útra

- zsákutca, azaz végpontjából nem vezet tovább út
- zsákutca torkolat, azaz végpontjából kivezető utak nem vezettek célba
- kör, azaz végpontja megegyezik az út egy megelőző csúcsával
- mélységi korlátnál hosszabb

Hanoi tornyai





Heurisztikák

- sorrendi heurisztika:
  - sorrendet ad végpontból kivezető élek (utak) vizsgálatára
- vágó heurisztika:
  - megjelöli azokat a végpontból kivezető éleket (utakat), amelyeket nem érdemes megvizsgálni

Első változat

- A visszalépéses algoritmus első változata az, amikor a visszalépés feltételei közül az első kettőt építjük be a kereső rendszerbe.
- A VL1 véges körmentes irányított gráfokon (nem kell δ-gráf) mindig terminál, és ha létezik megoldás, akkor talál egy megoldást.
- Rekurzív algoritmussal (VL1) szokták megadni
  - Indítás: *megoldás* ← *VL1*(*startcsúcs*)

Recursive procedure *VL1*(*csúcs*) return *megoldás*
1. if *cél*(*csúcs*) then return(*nil*) endif
2. *élek* ← *kivezető-élek*(*csúcs*)
3. while not *üres*(*élek*) loop
4. *él* ← *kivesz-egyet*(*élek*)
5. *megoldás* ← *VL1*(*vég*(*él*))
6. if *megoldás* ≠ *hiba* then
   return(*hozzáfűz*(*él*,*megoldás*)) endif
7. endloop
8. return(*hiba*)

end

Második változat

- A visszalépéses algoritmus második változata az, amikor a visszalépés feltételei közül mindet beépítjük a kereső rendszerbe.
- A VL2 δ-gráfban mindig terminál. Ha létezik a mélységi korlátnál nem hosszabb megoldás, akkor megtalál egy megoldást.
- Rekurzív algoritmussal (VL2) adjuk meg
  - Indítás: *megoldás* ← *VL2*(<*startcsúcs*>)

Recursive procedure *VL2*(*út*) return *megoldás*
1. *csúcs* ← *utolsó-csúcs*(*út*)
2. if *cél*(*csúcs*) then return(*nil*)
3. if *hossza*(*út*) ≥ korlát then return(*hiba*)
4. if *csúcs*∈ *maradék*(*út*) then return(*hiba*)
5. *élek* ← *kivezető-élek*(*csúcs*)
6. while not *üres*(élek) loop
7. *él* ← *kivesz-egyet*(*élek*)
8. *megoldás* ← *VL2*(*fűz*(*út*,*vég*(*él*)))
9. if *megoldás* ≠ *hiba* then return(*fűz*(*él*,*megoldás*))
10. endloop
11. return(*hiba*)

end

Megjegyzés

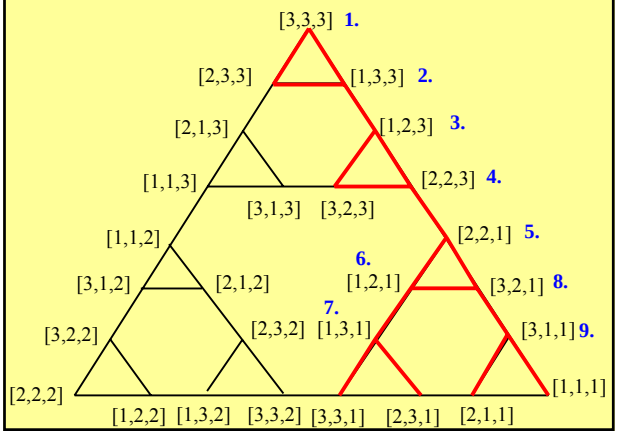
- Ha csak a megadott mélységi korlátnál hosszabb megoldási út van, akkor az algoritmus bár terminál, de nem talál megoldást.
- A mélységi korlát önmagában biztosítja a terminálást körök esetére is.
  - A mélységi korlát ellenőrzése jóval gyorsabb és kevesebb memóriát igényel, mint a körfigyelés.

Értékelés

- ELŐNYÖK
  - könnyen implementálható
  - kicsi memória igényű
- HÁTRÁNYOK
  - nem ad optimális megoldást. (iterációba szervezhető)
  - kezdetben hozott rossz döntést csak sok visszalépés korrigál (visszaugrásos keresés)
  - egy zsákutca részt többször is bejárhat a keresés

3. Gráfkereső stratégia

- A gráfkereső rendszer olyan KR, amelynek
  - globális munkaterülete a startcsúcsból kiinduló már feltárt utakat (részgráfot) tárolja
    - kiinduló értéke: a startcsúcs,
    - terminálási feltétel: megjelenik egy célcsúcs vagy megakad az algoritmus.
  - keresés egy szabálya: egy csúcs rákövetkezőit állítja elő (kiterjeszti),
  - vezérlés stratégiája: a legkedvezőbb csúcs kiterjesztésére törekszik,



3.1. Általános gráfkereső algoritmus

- ***Jelölés:***
  - G - keresőgráf
  - *NYÍLT* - nyílt csúcsok halmaza
  - Γ - kiterjesztés
  - kiterjesztett csúcsok - zárt csúcsok halmaza
- Az absztrakt keresési tér a továbbiakban is egy δ-gráf (nem feltétlenül véges)

Nulladik verzió

Procedure *GK0*

1. $G \leftarrow \{s\}$: *NYÍLT* $\leftarrow \{s\}$
2. while not *üres*(*NYÍLT*) loop
3. $n \leftarrow$ *elem*(*NYÍLT*)
4. if *cél*(n) then return *van megoldás*
5. $G \leftarrow G \cup \Gamma(n)$
6. *NYÍLT* $\leftarrow$ *NYÍLT*$-\{n\}$: *NYÍLT* $\leftarrow$ *NYÍLT* $\cup \Gamma(n)$
7. endloop
8. return *nincs megoldás*

end

Megjegyzés

- Körökre érzékeny
  - Zárt csúcs ne lehessen újra nyílt?
- Nehezen olvasható ki a megoldás
  - Jelölni kellene a megtalált utakat
- Nem garantál optimális megoldást, sőt megoldást sem
  - Jelölni kellene a megtalált utak költségeit

Függvények

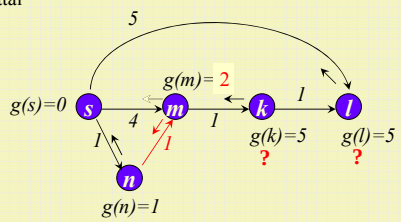
- f:NYÍLT → **R** kiértékelő függvény
  - a 3. lépésben $n \leftarrow min_f(NYÍLT)$
- a G egy s gyökerű irányított feszítőfája pointerekkel
  - π:$G \to G$ $\quad \pi(n) = n$ csúcs egyik szülője
    $\pi(s) = nil$
    - Jó lenne, ha a G-beli optimális költségű utakat mutatná
- Az n csúcshoz vezető, nyilvántartott $\alpha \in \{s \to n\}$ út költsége
  - g:$G \to$ **R** költség függvény $g(n) := c^{\alpha}(s,n)$
    - Jó lenne, ha konzisztens lenne a π-vel
    - Jó lenne, ha a G-beli optimális költséget mutatná

Gyermek csúcs három esete

- Új csúcs
  Ha $m \notin G$ akkor
  $\pi(m) \leftarrow n,\ g(m) \leftarrow g(n)+c(n,m)$
  $NYÍLT \leftarrow NYÍLT \cup \{m\}$
- Régi csúcs, amelyhez olcsóbb utat találtunk
  Ha $m \in G$ és $g(n)+c(n,m)<g(m)$ akkor
  $\pi(m) \leftarrow n,\ g(m) \leftarrow g(n)+c(n,m)$
- Régi csúcs, amelyhez nem találtunk olcsóbb utat
  Ha $m \in G$ és $g(n)+c(n,m) \geq g(m)$ akkor *SKIP*

Optimális költségű konzisztens feszítőfa?

- Ha $m \in G$ és $g(n)+c(n,m)<g(m)$, és m csúcsnak vannak leszármazottai



- Nem törődjünk a zárt m csúcs leszármazottaival, de magát az m csúcsot helyezzük vissza a *NYÍLT* halmazba.

Procedure *GK*
1. $G \leftarrow \{s\}$: $NYÍLT \leftarrow \{s\}$: $g(s) \leftarrow 0$: $\pi(s) \leftarrow nil$
2. **while** not *üres*(*NYÍLT*) **loop**
3. $n \leftarrow min_f(NYÍLT)$
4. **if** *cél*(n) **then return** megoldás
5. $NYÍLT \leftarrow NYÍLT - \{n\}$
6. **for** $\forall m \in \Gamma(n)$ **loop**
7. **if** $m \notin G$ or $g(n)+c(n,m)<g(m)$ **then**
8. $\pi(m) \leftarrow n,\ g(m) \leftarrow g(n)+c(n,m),\ NYÍLT \leftarrow NYÍLT \cup \{m\}$
9. **endloop**
10. $G \leftarrow G \cup \Gamma(n)$
11. **endloop**
12. **return** nincs megoldás

end

Működés és eredmény

A *GK* a működése során egy csúcsot legfeljebb véges sokszor terjeszt ki. (a körökre nem érzékeny)	A *GK* véges δ-gráfban mindig terminál.
A *GK* a működése során bármelyik s-ből elérhető még ki nem terjesztett n csúcsra ismeri az összes $s^* \to n$ optimális útnak egy nyílt csúcsig tartó kezdő szakaszát.	Ha egy véges δ-gráfban létezik megoldás akkor a *GK* egy célcsúcs megtalálásával terminál.

3.2. Nevezetes gráfkereső algoritmusok

- Most az *f* kiértékelő függvény megválasztása következik.

Neminformált

- mélységi,
- szélességi,
- egyenletes

Heurisztikus

- előre tekintő (mohó),
- A, A*, Aᶜ

Heurisztikus függvény

- Azt $h:N \rightarrow \mathbf{R}$ függvényt, amelyik minden *n* csúcsra az abból a célba vezető út költségére ad becslést heurisztikus függvénynek hívjuk.
- $h(n) \approx h^*(n) = \min_{t \in T} c^*(n,t) = c^*(n,T)$
- Példa
  - *h=0*
  - *8*-as játék: *h=W, h=P*

Nevezetes algoritmusok és tulajdonságaik

mélységi	$f = -g,\ c(n,m) = 1$	csak mélységi korláttal garantál megoldást
szélességi	$f = g,\ c(n,m) = 1$	legrövidebb megoldást adja legfeljebb egyszer terjeszt
egyenletes	$f = g$	legolcsóbb megoldást adja legfeljebb egyszer terjeszt
előre tekintő	$f = h$	megengedhető
A	$f = g+h\ ,\ 0 \le h$	megoldást ad, ha van
A*	A alg + $h \le h^*$	optimális megoldást ad, ha van
Aᶜ	A alg + $h(t) = 0$ $h(n)-h(m) \le c(n,m)$	optimális megoldást ad, ha van legfeljebb egyszer terjeszt

monoton megszorításos

„$h(t)=0$” + „$h(n)-h(m) \le c(n,m)$” $\Rightarrow$ „$h \le h^*$”

3.3. *A* algoritmus hatékonysága*

Hatékonyság

Memória igény

Zárt csúcsok száma a termináláskor

- 8-as kirakó: $W \le P \le h^*$
- A* az egyik legjobb

Futási idő

Kiterjesztések száma a zárt csúcsok számához viszonyítva

- *k* és 2^{k-1} között, ahol *k* a zárt csúcsok száma

Olyan problémákat vizsgálunk, ahol van megoldás: az *A* algoritmus* optimális megoldással terminál.

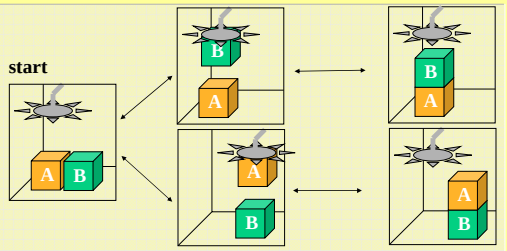
III. REDUKCIÓ, DEKOMPOZÍCIÓ

1. Visszafelé haladó keresés
2. Probléma redukció
3. Probléma dekompozíció
4. ÉS/VAGY gráfok

1. Visszafelé haladó keresés

- Ha a problématér a cél felől nézve egyszerűbb (kevesebb alternatívát mutat), mint a start felől nézve, akkor érdemes visszafelé haladó keresést alkalmazni.
- A talált megoldási utat azonban a starttól a cél felé haladva kell értelmezni. (Van-e az útnak inverze?)

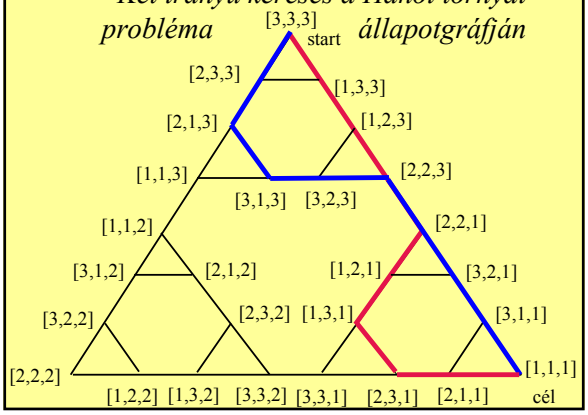
Kockavilág probléma



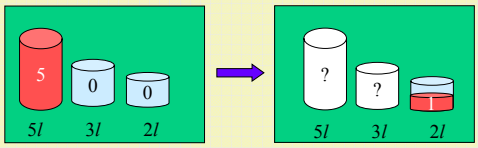
- A megoldási út megtalálása a célból a start felé haladva egyszerűbb.

cél

Két irányú keresés a Hanoi tornyai probléma állapotgráfján



Miért nem oldja meg a kancsó-problémát egy visszafelé haladó keresés?



- Kiindulási célállapotot kiválasztása nem egyszerű: Nem elérhető célállapot például a *(2,2,1)*.
- A visszafelé haladó kereséssel talált *(4,0,1)*→*(5,0,0)* út nem értelmezhető a feladat megoldásaként.

Visszafelé haladó keresés feltételei

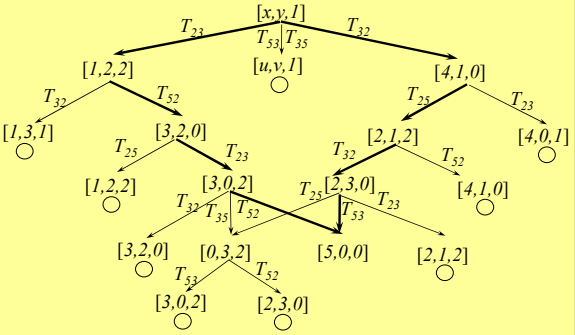
- A műveletek invertálhatóak legyenek (legalábbis a visszafelé haladó keresés által alkalmazottak).
- Konkrét célállapotot kell választani. (Ettől a talált megoldás költsége is függ.)

Mit tegyünk, ha a fenti két feltétel nem áll fenn, de visszafelé haladó keresést akarunk megvalósítani?

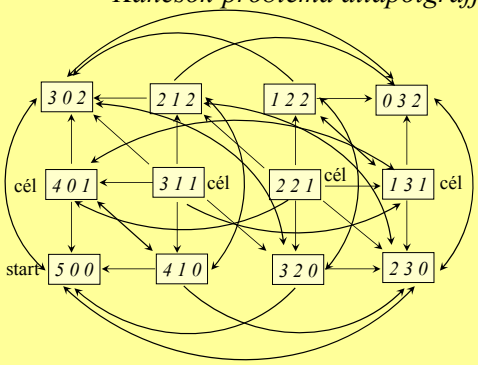
2. Probléma redukció

- Hogyan határozható meg egy csak részben ismert állapothoz az azt megelőző állapot?
- Két kérdésre keressük a választ:
  - Van-e olyan művelet, amellyel elérhető az éppen vizsgált állapot?
  - Melyik az a megelőző állapot, amelyikből egy kiválasztott művelet az éppen vizsgált állapotba vezet?

Kancsók-probléma redukciós gráfja



Kancsók-probléma állapotgráfja



Redukciós reprezentáció fogalma

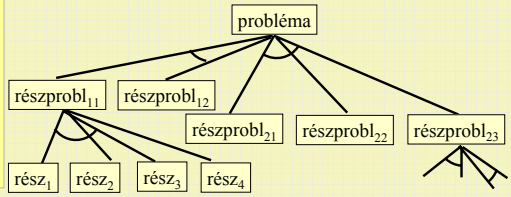
- A reprezentációhoz meg kell adnunk a feladat
  - állapottér-reprezentációját,
  - majd minden művelethez definiálunk egy redukciós műveletet, amely egy állapothoz azokat a megelőző állapotokat rendeli, amelyekből a rögzített művelet az aktuális állapotba vezet.
    - *M* művelethez tartozó redukciós művelet:
      $B_M \subseteq állapot \times állapot$ és
      $b \in B_M(a)$ ha $M(b)=a$

Megjegyzés

- A redukció során eljuthatunk „érdektelen” illetve „hamis” állapot-leírásokhoz.
- Gráfreprezentáció készíthető: ebben kell utat keresni, ehhez kereső rendszer építhető.
- A talált (célból startba vezető) út visszafelé olvasva adja ki a megoldást, amely nem az inverze a talált útnak.

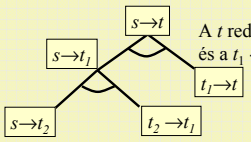
3. Dekompozíció

- A dekompozíció általánosítása a redukciónak: egy feladatot több részfeladatra bontunk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható feladatokat nem kapunk.



A redukció kétfelé bont egy problémát

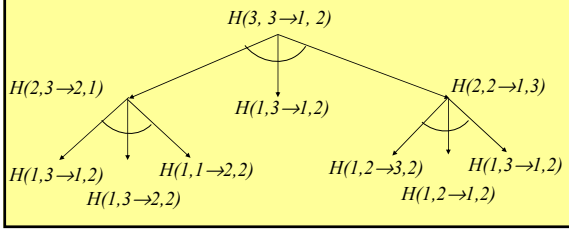
- A redukció során a megoldandó feladatot mindig két részre: egy nyilvánvalóan megoldható és egy további redukálást igénylő részfeladatra bontottuk.



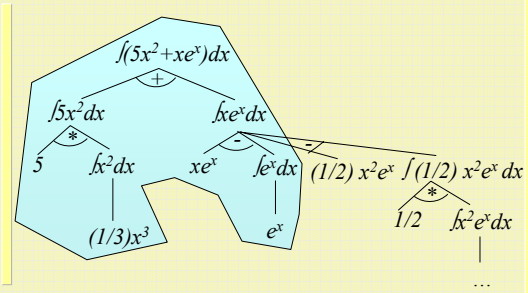
A t redukálásának eredménye a t_1 és a t_1 -ből t-be vezető művelet.

Hanoi tornyai probléma megoldása dekompozícióval

H(n, i→j, k) helyett
H(n-1, i→k, j) H(1, i→j, k) H(n-1, k→j, i)



Integrálszámítás



Dekompozíciós reprezentáció fogalma

- A reprezentációhoz meg kell adnunk:
  - a feladat részproblémáinak általános leírását,
  - az eredeti problémát,
  - az egyszerű problémákat, amelyekről könnyen eldönthető, hogy megoldhatók-e vagy sem, és
  - a dekomponáló műveleteket:
    - D: *probléma* → *probléma*$^+$ és $D(p)=<p_1, \ldots, p_n>$

A dekompozíciós reprezentáció nehéz

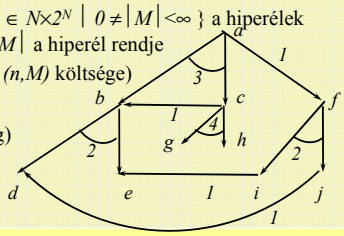
- Dekomponáló műveleteket nagyon nehéz megtalálni.
  - Nem minden feladat dekomponálható.
  - Nem biztos, hogy minden dekomponálást észrevettünk.
  - Hamis dekomponáló műveletek.
- Az egyszerű probléma felismerése sem egyértelmű
  $\int sin(x)e^x dx = \ldots = sin(x)e^x - cos(x)e^x - \int sin(x)e^x dx$
- A megoldás kiolvasása nem nyilvánvaló.

Gráfreprezentáció

- A feladat problématerét nem egy közönséges irányított gráf írja le, hanem egy úgynevezett ÉS/VAGY gráf.
- A megoldást sem egy közönséges irányított út szimbolizálja, hanem egy speciális részgráf: a megoldásgráf
  - A megoldásgráfnak egyértelmű haladási irányt kell kijelölnie a startcsúcsból a célcsúcsokba.
- A probléma megoldása nem a megoldásgráf, de abból olvasásható ki.

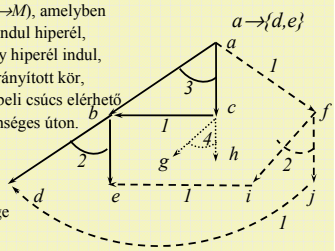
4. ÉS/VAGY gráfok

- Az $R=(N,A)$ élsúlyozott irányított hipergráf, ahol az
  - N a csúcsok halmaza,
  - $A \subseteq \{ (n,M) \in N \times 2^N \mid 0 \neq |M| < \infty \}$ a hiperélek halmaza, $|M|$ a hiperél rendje
  - $(c(n,M)$ az (n,M) költsége)
- σ tulajdonság
- (δ tulajdonság)



Az n csúcsból az M csúcshalmazba vezető irányított hiperút fogalma

- A hiperút egyértelmű haladási irányt kijelölő hiperélek halmaza, azaz
- egy véges részgráf ($n^{\alpha} \rightarrow M$), amelyben
  - M csúcsaiból nem indul hiperél,
  - a többi csúcsból egy hiperél indul,
  - nincs közönséges irányított kör,
  - bármelyik részgráfbeli csúcs elérhető az n csúcsból közönséges úton.
- Hiperút hossza, költsége



Dekompozíciós gráfreprezentáció

- Egy dekompozíciós reprezentációhoz tartozó (R,s,T) gráfreprezentációban
  - az $R=(N,A,c)$ egy olyan ÉS/VAGY gráf (dekompozíciós gráfban), ahol
  - N a részproblémákat,
  - A a dekomponáló műveleteket,
  - c azok költségeit szimbolizálják,
  - s az eredeti problémát,
  - T az egyszerű problémákat jelöli.

Megjegyzés

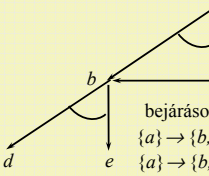
- A probléma megoldását egy $s \rightarrow M \subseteq T$ hiperút, az úgynevezett megoldásgráf megtalálása jelenti. Az eredeti probléma megoldása ebből a megoldásgráfból nyerhető ki.
- A megoldás költsége többnyire nem függ a megoldásgráf költségétől, ezért nem cél, az optimális megoldásgráf előállítása.

Keresés ÉS/VAGY gráfban

- Egy ÉS/VAGY gráfon folyó keresés a startcsúcsból kivezető hiperutak (köztük a megoldásgráfok) között folyik.
- Az útkereső algoritmusainkat közönséges gráfokra fogalmaztuk meg. De mivel minden hiperút fogalma rokon a közönséges útéval, ezért a tanult keresések könnyen adaptálhatók ÉS/VAGY gráfokra.
- Vegyük szemügyre az ÉS/VAGY gráfok és a közönséges δ-gráfok közötti kapcsolatot, hogy ezt az adaptációt elvégezhessük.

Különbség a közönséges út és a hiperút bejárása között

- Egy közönséges út bejárásán az úton fekvő élek felsorolását értjük. Ennek megadása egyértelmű.
- A hiperút is egyértelmű haladási irányt jelöl ki, de ez többféleképpen is bejárható.



bejárások:

$\{a\} \rightarrow \{b,c\} \rightarrow \{b\} \rightarrow \{d,e\}$

$\{a\} \rightarrow \{b,c\} \rightarrow \{d,e,c\} \rightarrow \{d,e,b\} \rightarrow \{d,e\}$

Megjegyzés

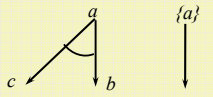
- A bejárás a hiperút összes hiperélét tartalmazó hiperél-sorozat, amelyben ugyanaz a hiperél többször is szerepelhet.
- Az $n \to M$ hiperút (k,K) hiperéle legfeljebb annyiszor szerepel egy bejárás során, amennyi közönséges út vezet a hiperútban az n csúcsból k csúcshoz.
- Egy bejárás véges hosszú.
- Egy hiperútnak véges sok különböző bejárása lehet.

Hiperút bejárása

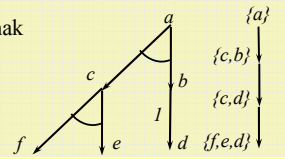
- Az $n \to M$ hiperút egy bejárásán a hiperút csúcsaiból képzett halmazoknak olyan felsorolását értjük, amelyben
  - Az első az $\{n\}$ halmaz, a második az n csúcsból kivezető (egyetlen) hiperél utódhalmaza.
  - Általában egy C halmaz után a $C-\{k\}\cup K$ halmaz következik, ha van olyan (k,K) hiperél az $n \to M$ hiperúton, hogy $k \in C$ és $k \notin M$.
  - A bejárás utolsó csúcshalmaza az M halmaz.

Bejárások ábrázolása közönséges utakkal

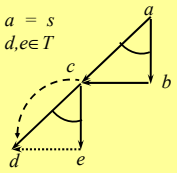
Egyetlen hiperélből álló útnak egyetlen bejárása van



Több hiperélből álló útnak egyik bejárása

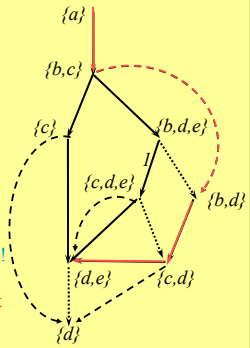


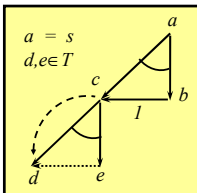
ÉS/VAGY gráf átalakítása



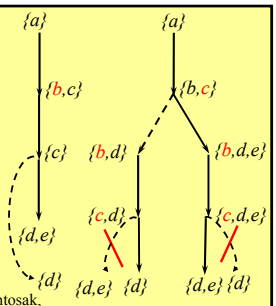
A startból induló hiperutak bejárásait közönséges útként rajzoljuk fel.
Az átalakítás nem költségtartó.

Nem kell egy hiperútnak minden bejárása!
Törölni kell a hamis bejárásokat, azokat, ahol ugyanaz a csúcs többször és másként kerül helyettesítésre!





1. Egy hiperútnak elég lenne csak egy bejárását megadni, ezért az átalakítás egy lépésében elég egy halmaznak egy csúcsát helyettesíteni



2. Számunkra csak az $s \to M \subseteq T$ hiperutak a fontosak, ezért a célcsúcsokból már ne lépjünk tovább.

hamis bejárások!

3. A hamis bejárások kizárása érdekében a közönséges gráfot fává egyenesítve adjuk meg, és ha ennek egy csúcsát címkéző csúcshalmazból olyan k csúcsot választunk a továbblépéshez, amelyhez a halmazhoz vezető úton korábban már a (k,K) hiperélt illesztettük, akkor it isk ezt a hiperélt használhatjuk fel, más él nem vezethet ki.

Átalakító algoritmus

- Betesszük egy SOR-ba az $\{s\}$ halmazt, mint startcsúcsot.
- Amíg a SOR nem üres addig kiveszünk a SOR-ból egy C halmazt, és generáljuk a C-ből kivezető éleket:
  - Legyen k C-beli nem célcsúcs. (Ha C csupa célcsúcsból áll, akkor a C maga is célcsúcs és belőle nem indul ki él.)
  - Ha a C-hez vezető úton nincs olyan él, amelyet $(k,K) \in A$ hiperéllel generáltunk, akkor az összes $(k,K) \in A$ hiperélre generálunk egy-egy $C-\{k\}\cup K$ utódot.
  - Ha a C-hez vezető úton van olyan él, amelyet egy $(k,K) \in A$ hiperéllel generáltunk, akkor csak ezzel a (k,K) hiperéllel generálunk egy $C-\{k\}\cup K$ utódot.
  - Az utódokat betesszük a SOR-ba.

Tétel

1. Az átalakítással nyert közönséges gráfok minden megoldási útja egy $s \rightarrow M \subseteq T$ hiperútnak egy bejárását írja le.
2. Az átalakítás minden $s \rightarrow M \subseteq T$ hiperút valamelyik bejárásához véges lépésben megfeleltet egy közönséges megoldási utat.

- Megjegyzés:
  - Az átalakított gráf egy δ-gráf (költségek!)
  - Az átalakítást beépítik a keresésekbe.

Visszalépéses keresés ES/VAGY gráfokon

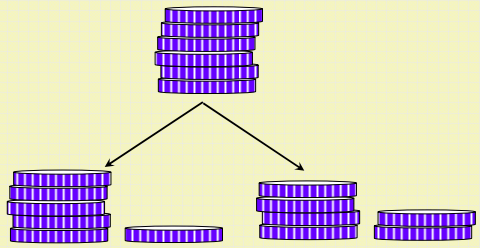
```
Recursive procedure VL2(bejárás) return megoldás
1.    C ← vége(bejárás)
2.    if csupacél(C) then return(nil) endif
3.    if hossza(bejárás) ≥ korlát then return(hiba) endif
4.    if C ∈ maradék(bejárás) then return(hiba) endif
5a.   k ← kivesz-egy-nemcélcsúcsot(C)
5b.   hiperélek ← kivezető-hiperélek(k)
6.    while not üres(élek) loop
7.          (k,K) ← kivesz(hiperélek)
8.          megoldás ← VL2( hozzáfűz(C-{k}∪K, bejárás) )
9.          if megoldás ≠ hiba then
                  return(hozzáfűz((C, C-{k}∪K), megoldás))
            endif
10.   endloop
11.   return(hiba)
end
```

IV. KÉTSZEMÉLYES JÁTÉKOK

Teljes információjú, véges, zéró összegű kétszemélyes játékok

- Két játékos lép felváltva adott szabályok szerint.
- Mindkét játékos ismeri a maga és az ellenfele összes választási lehetőségét, és azok következményeit.
- Mindkét játékos minden lépésében véges számú lehetőség közül választhat;
  minden játszma véges lépésben véget ér.
- Amennyit az egyik játékos nyer, annyit veszít a másik. (Legegyszerűbb változatban: egyik nyer, másik veszít, esetleg lehet döntetlen is)

Grundy mama játéka

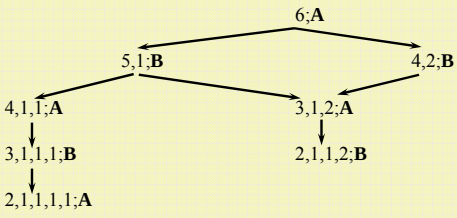


Állapottér-reprezentáció

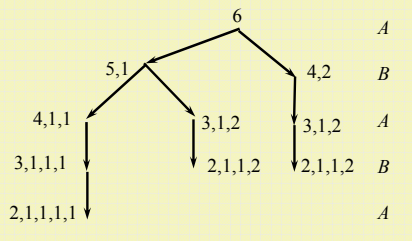
- állapot - állás + soron következő játékos
- művelet - lépés
- kezdő állapot - kezdőállás + kezdő játékos
- célállapot - végállás
  (nyerő, vesztő vagy döntetlen)

Egy játszma egy kezdőállapotból célállapotba vezető (mindig véges) műveletsorozat

Grundy mama játékgráfja



Grundy mama játékfája



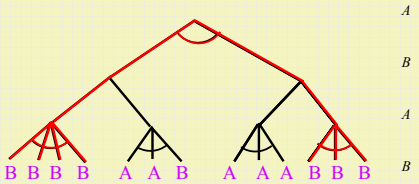
Játékfa

- csúcs - állás (egy állásnak több csúcs is megfelelhet)
- szint - játékos
- él - lépés (szintről szintre)
- gyökér - kezdőállás (kezdő játékos)
- levél - végállások
- ág - játszma

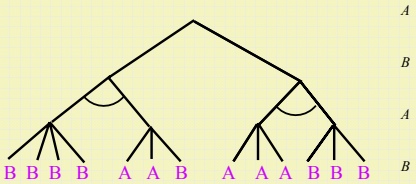
Nyerő stratégia

- Az egyik játékosnak akkor van nyerő stratégiája (vagy nem vesztő stratégiája), ha mindig tud olyat lépni, hogy ellenfele bármilyen játéka esetén számára kedvező végállásba tud jutni.

*Nyerő stratégia keresése az **B** játékos ÉS/VAGY fájában*



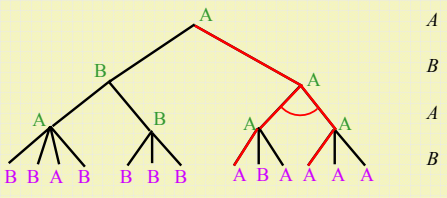
*Nyerő stratégia keresése az **A** játékos ÉS/VAGY fájában*



- Csak az egyik játékosnak lehet nyerő stratégiája.

Tétel

- Egyik játékos számára mindig létezik nyerő stratégia (nem vesztő stratégia).



Részleges játékfa-kiértékelés

- Minimax algoritmus
- Negamax algoritmus
- Átlagoló kiértékelés
- Váltakozó mélységű
- Szelektív kiértékelés
- Alfabéta algoritmus

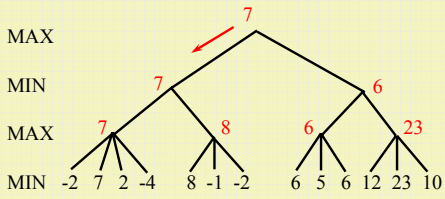
Kiértékelő függvény

- Minden esetben szükségünk van egy olyan heurisztikára, amely megbecsüli, hogy egy állás mennyire ígéretes.
- Ez mindig csak az egyik játékos szempontjait tükrözi.
- Sokszor ez egy *f*:állás → [-100..100] függvény.

Minimax algoritmus

- Adott állásból indulva felépítjük a játékfa néhány szintjét.
- A részfa leveleit kiértékeljük aszerint, hogy azok számunkra kedvező, vagy kedvezőtlen állások.
- Az értékeket felfuttatjuk a fában. (Saját szintjeink csúcsaihoz azok gyermekeinek maximumát, ellenfél csúcsaihoz azok gyermekeinek minimumát rendeljük.)
- Soron következő lépésünk ahhoz az álláshoz vezet, ahonnan a gyökérhez felkerült a legnagyobb érték.

Példa



Megjegyzés

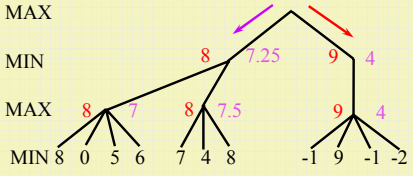
- Az algoritmust minden alkalommal, valahányszor mi következünk, megismételjük, hiszen lehet, hogy az ellenfél nem az általunk várt legerősebb lépésekkel válaszol, mert:
  - eltérő mélységű részfával dolgozik,
  - más kiértékelő függvényt használ,
  - nem minimax eljárást alkalmaz,
  - hibázik.

Negamax algoritmus

- Negamax eljárást könnyebb implementálni.
  - Az ellenfél szintjén levő levelek értékének vesszük a (-1)-szeresét, majd
  - minden szinten szülő=max(-gyerek$_1$,..., -gyerek$_n$).

Átlagoló kiértékelés

- Az (*m*,*n*) átlagolás célja a kiértékelő függvény esetleges tévedéseinek simítása.
- Legyen például *n*=*2* és *m*=*2*.



Váltakozó mélységű kiértékelés

- Célja, hogy a kiértékelő függvény minden ágon reális értéket mutasson.
- A részfa felépítését módosítjuk úgy, hogy egy adott szinttől kezdve, akkor vesszük bele egy csúcs utódait a részfába, ha minden utódra teljesül a nyugalmi teszt:
  - $|f(\text{szülő}) - f(\text{gyerek})| < K$

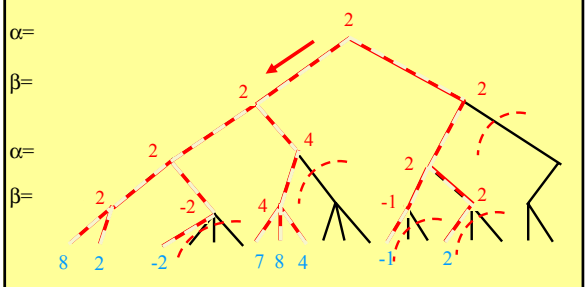
Szelektív kiértékelés

- Célja a memória-igény csökkentése.
- Elkülönítjük a lényeges és lényegtelen lépéseket, és csak a lényeges lépéseknek megfelelő részfát építjük fel.

Alfa-béta algoritmus

- Visszalépéses algoritmus segítségével járjuk be a részfát. (mélységi bejárás) Az aktuális úton fekvő csúcsokat:
  - a mi szintünkön α értékkel (ennél rosszabb értékű állásba innen már nem juthatunk),
  - az ellenfelén β értékkel(ennél jobb értékű állásba onnan már nem juthatunk) látjuk el.
- Lefelé haladva $\alpha=-\infty$, és $\beta=+\infty$.
- Ezek visszalépéskor a felhozott értékre változnak, ha az nagyobb, mint az α, illetve kisebb, mint a β.
- Vágás: ha az úton van olyan α és β, hogy $\alpha \geq \beta$.

Példa



Eredmény

- Több egyformán jó kezdőirány esetén a baloldalit kell választani.
- Ekkor ugyanazt a kezdőlépést kapjuk eredményül, mint a minimax algoritmussal talált baloldali legjobb kezdőlépés.

Hatékonyság

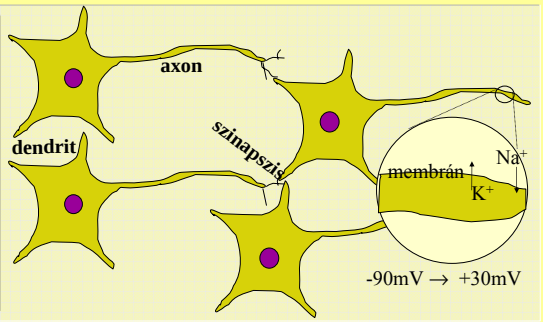
- Tárigény: csak egy utat tárol.
- Futási idő: a vágások miatt sokkal jobb, mint a minimax módszeré.
  - Optimális eset: egy *d* mélységű *b* elágazású fában kiértékelt levélcsúcsok száma: $\sqrt{b^d}$
  - Átlagos eset: egy csúcs alatt, két belőle kiinduló ág megvizsgálása után már vághatunk.
  - Jó eset: A bejárt részfa megfelelő rendezésével érhető el. („cáfoló lépés elve”)

Kétszemélyes játékot játszó program

- Váltakozó mélységű, szelektív, (*m,n*) átlagoló, negamax alfa-béta kiértékelést végez.
- Keretprogram, amely váltakozva fogadja a felhasználó lépéseit, és generálja a számítógép lépéseit.
- Kiegészítő funkciók (beállítások, útmutató, segítség, korábbi lépések tárolása, mentés stb.)
- Felhasználói felület, grafika
- Heurisztika megválasztása (kiértékelő függvény, szelekció, kiértékelés sorrendje)

V. Mesterséges neuronhálók

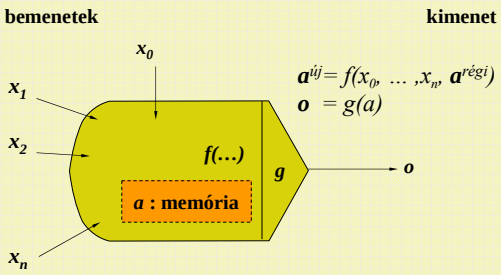
Természetes neuronhálók



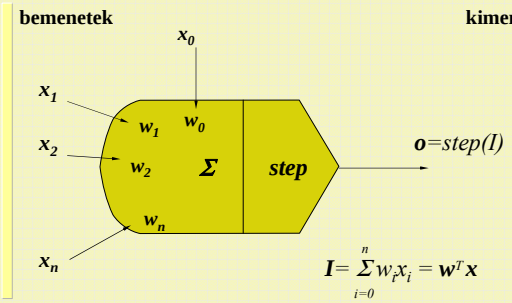
1. Mesterséges neuronháló fogalma

- Mesterséges neuron
  - bemenő értékekből kimenő értéket számoló egység, amelynek számítási képlete változtatható
- Hálózati topológia
  - több mesterséges neuron egymáshoz kapcsolásának módja
- Tanulási szabály
  - egy neuron számítási képletét, esetleg a hálózat topológiáját minták alapján módosító eljárás
- Alkalmazás
  - approximáció
  - asszociatív memória
  - optimalizálás

1.1. Mesterséges neuron



1.1.1. Perceptron



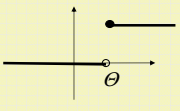
Megjegyzés

- Az x_0 (stimuláló) bemenetnek speciális szerepe van: meghatározza a sejt ingerküszöb értékét.
- Például *step* aktivizációs függvény esetén

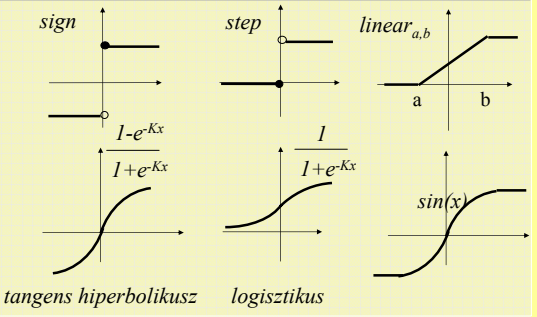
$step(I) = step\left(\sum_{i=1}^{n} w_i x_i + w_0 x_0\right)$

azaz itt egy $\Theta = -w_0 x_0$ küszöbértékű lépcsős függvényről van szó

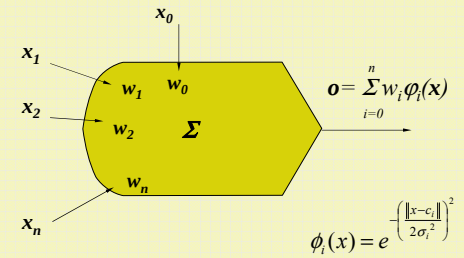
$step_\Theta\left(\sum_{i=1}^{n} w_i x_i\right) = step\left(\sum_{i=1}^{n} w_i x_i - \Theta\right)$



Aktivizációs függvények



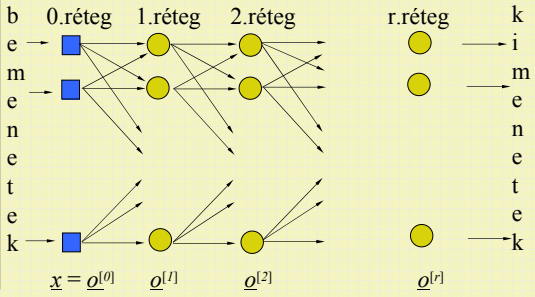
1.1.2. Bázisfüggvényes neuron



1.2. Hálózati topológia

- A mesterséges neuronháló egy olyan irányított gráf, amelynek csúcsai vagy a háló bemeneti értékeit jelenítik meg, vagy egy-egy (általában azonos konfigurációjú) mesterséges neuront szimbolizálnak.

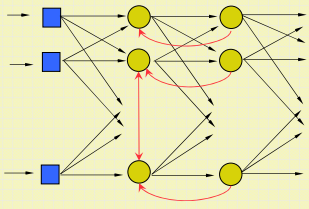
Előrecsatolt, rétegzett topológia



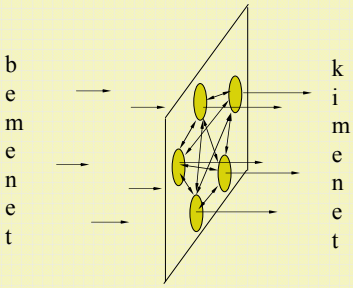
Előrecsatolt, rétegzett hálózat működése

- Számítási modell, amely bemenő értékekre kimenő értékeket számol:
  - Az x_i bemeneti értékek egy *0*-adik réteg neuronjainak kimeneteként foghatók fel ($o_i^{[0]}$),
  - Az *s*-edik rétegnek $n^{[s]}$ darab neuronja van, amely mindegyike kiszámolja a saját kimeneti értékét: $o_j^{[s]}$
  - Az *s*-edik réteg *j*-edik neuronjának *i*-edik bemenete = az *s-1*-edik réteg *i*-edik neuronjának kimenete: $o_i^{[s-1]}$
  - Az *s*-edik réteg *j*-edik neuronjának *i*-edik bemenetéhez tartozó súly: $w_{ij}^{[s]}$

Visszacsatolt rétegzett topológia



Hopfield topológia



Hopfield topológia működése

- Célja egy nyugalmi helyzet előállítása
  - Minden neuron kezdő állapota egy-egy bemeneti érték
  - Egy neuron mindaddig újra számolja a többi neuron kimeneti értéke alapján a belső állapotát, ameddig az eltér a korábbi állapotától.
  - A stabil helyzetben kialakult állapotok lesznek a kimeneti értékek.

Kapcsolatok osztályozási szempontjai

- kitöltöttség szerint: teljes, egy-egy, vagy véletlen
- rétegelés szerint: Ha a neuronokat rétegekre osztjuk (egy rétegbe tartozó neuronok számításai párhozamosan végezhetők), akkor beszélhetünk rétegen belüli ill. rétegek közötti kapcsolatokról
- irányítás szerint: előre csatolt vagy visszacsatolt

1.3. Tanulás

- A tanulás a hálózat paramétereinek (topológia, aktivizációs függvény) tanító példák alapján történő beállítása.
- Leggyakrabban a súlyokat tanuljuk meg:
  - Mintákat, azaz lehetséges bemeneteket mutatunk a mesterséges neuronhálónak, amely minden mintára kiszámítja a kimenetet, majd ez alapján módosítja a súlyokat: $w_i \leftarrow w_i + \Delta w_i$
  - Ez kihat egy-egy neuron számítási képletére, de közvetve a topológiára is.

Tanulási formák

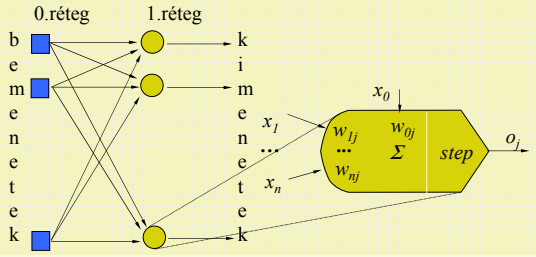
- Felügyelt tanulás tanítóval
  - adottak minta feladatok pontos eredményükkel
- Felügyelt tanulás kritikussal (megerősítéses tanulás)
  - adottak minta feladatok értékelésükkel
- Felügyelet nélküli (önszerveződő) tanulás
  - adottak minta feladatok
- Analitikus tanulás
  - A mintafeladatok felhasználása nem adaptív módon történik

Felügyelt	**Felügyelet nélküli**
Pl: Delta szabály perceptronra ha x_i a neuron *i*-dik bemenete t a várt kimenet o a számított kimenet akkor $\Delta w_i = \eta * x_i * (t-o)$	**Pl:** Hebb szabály perceptronra ha o a neuron számított kimenete x_i a neuron *i*-dik bemenete, ami egyben egy megelőző neuron kimenete is akkor $\Delta w_i = \eta * x_i * o_j$

η a tanulási együttható

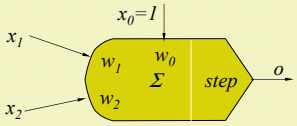
2. Perceptron modell

- Step aktivizációs függvényű, belső állapot nélküli neuronok egy rétegű perceptron hálója, felügyelt tanulással.



Példa

AND művelet



Beállítások:

$x_1, x_2, o \in \{0,1\}$

$x_0 = 1$

$w_0, w_1, w_2 \in R$

$f(x) = step(x)$

Tanulás

- o - a számított kimenet t - a várt kimenet
- ha t-o=0 (t és o azonos) akkor semmit nem kell tenni
- ha t-o=1 (azaz t=1 és o=0) akkor az o-t növelni kell a kiszámításában aktív bemenetek súlyainak növelésével
- ha t-o=-1 (azaz t=0 és o=1) akkor az o-t csökkenteni kell a kiszámításában aktív bemenetek súlyainak csökkentésével
- Ez egy delta szabály: $\Delta w_i = \eta * x_i * (t-o)$

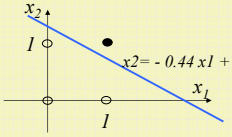
Alkalmazás

$\eta = 0.1$

	x_1	x_2	w_0	w_1	w_2	I	o	t	e
0.			0.08	0.08	0.08				
1.	1	0	0.08	0.08	0.08	0.160	1	0	-1
2.	0	1	-0.02	-0.02	0.08	0.06	1	0	-1
3.	1	1	-0.12	-0.02	-0.02	-0.16	0	1	1
4.	1	0	-0.02	0.08	0.08	0.06	1	0	-1
5.	0	1	-0.12	-0.02	0.08	-0.04	0	0	0
6.	1	1	-0.12	-0.02	0.08	-0.06	0	1	1
7.	1	0	-0.02	-0.08	0.18	0.06	1	0	-1
…									
14.			-0.22	0.08	0.18				
14.	1	0	-0.22	0.08	0.18	-0.14	0	0	0
15.	0	1	-0.22	0.08	0.18	-0.04	0	0	0
16.	1	1	-0.22	0.08	0.18	0.04	1	1	0
17.	0	0	-0.22	0.08	0.18	-0.22	0	0	0

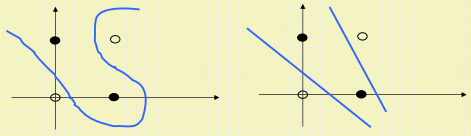
Mit „tud” egy perceptron?

- A perceptron súlyai annak a hipersíknak egyenletét határozzák meg, amelyik a bemeneti érékeket két részre osztja.
  - Az AND műveletre betanított neuron bemenet-párjai a síkon ábrázolhatóak.
  - A neuron összegzett bemenete egy $I(x_1,x_2) = 0$ egyenest jelöl ki.
  - A $w_0 + w_1x_1 + w_2x_2 = 0$ egyenlet grafikonja ketté vágja a síkot: egyik felébe azok a bemenet-párok kerülnek, amelyekre $I(x_1,x_2)<0$ (ilyenkor a kimenet 0), a másikba azok, amelyekre $I(x_1,x_2)<0$ (a kimenet 1).



Lineáris szeparálhatóság

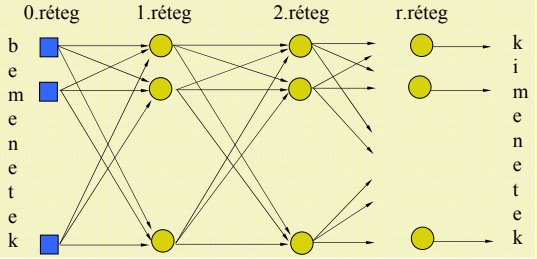
- Egy perceptron csak lineárisan szeparálható osztályozási problémákat képes megoldani.



- Az egyrétegű perceptron modell csak hiperfélsíkokkal képes felosztani a bemenetek terét, a kétrétegű perceptron modell már konvex poliéderekkel, a három rétegű háló pedig tetszőleges poliéderekkel. A többrétegű perceptron modellekhez azonban nem találtak tanuló eljárást, ezért a súlyai csak közvetlen módon (analitikusan) állíthatók be.

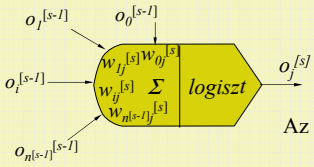
3. Backpropagation modell

- Logisztikus függvényű, belső állapot nélküli neuronok több rétegű hálója, felügyelt tanulással.



Neuron

Az *s*-edik réteg *j*-edik neuronja:



Az első ($s=1$) rétegnél: $o_i^{[s-1]}=x_i$

Energia függvény

A háló energiafüggvénye egyetlen tanító minta esetén:

$$E = \tfrac{1}{2}\sum_{j=1}^{n^{[r]}} (t_j - o_j^{[r]})^2$$

Az E tekinthető úgy is, mint háló $\{w_{ij}^{[s]}\}$ súlyainak $E(w_{11}^{[1]}, \ldots)$ függvénye, ezért értékének minimalizálásához a háló súlyait a gradiens módszer alapján lehet változtatni:

$$w_{ij}^{[s]} \leftarrow w_{ij}^{[s]} - \eta \frac{\partial E}{\partial w_{ij}^{[s]}}$$

Tanulási szabály

Az E többek között függ az *s*-edik réteg *j*-edik neuronjának összegzett bemenetétől ($I_j^{[s]}$) is, ami viszont az *i*-edik súly ($w_{ij}^{[s]}$) értékétől függ:

$$-\eta \frac{\partial E}{\partial w_{ij}^{[s]}} = -\eta \frac{\partial E}{\partial I_j^{[s]}} \frac{\partial I_j^{[s]}}{\partial w_{ij}^{[s]}} = -\eta \frac{\partial E}{\partial I_j^{[s]}} o_i^{[s-1]}$$

$e_j^{[s]}$

hiszen $I_j^{[s]} = \sum_{i=0}^{n^{[s-1]}} w_{ij}^{[s]} o_i^{[s-1]}$

az s-dik réteg j-edik neuronjára hárított hibahányad

s=r eset

$$e_j^{[r]} = -\frac{\partial E}{\partial I_j^{[r]}} = \tfrac{1}{2}\, 2\, (t_j - f(I_j^{[r]}))\, f'(I_j^{[r]}) = (t_j - o_j^{[r]}) o_j^{[r]} (1 - o_j^{[r]})$$

Ui: Egyrészt az $o_j^{[r]} = f(I_j^{[r]})$ miatt az energiafüggvény felírható az

$E = \frac{1}{2}\sum_{j=1}^{n^{[r]}} (t_j - f(I_j^{[r]}))^2$ alakban, ahol $f(I_j^{[r]}) = o_j^{[r]}$

másrészt $f'(x) = f(x)(1-f(x))$, hiszen $f(x) = \dfrac{1}{1+e^{-x}}$

s<r eset

E függ $s+1$-edik réteg $I_k^{[s+1]}$összegzett bemeneteitől is, amelyek azonban mindannyian függnek az *s*-edik réteg *j*-edik neuronjának $I_j^{[s]}$ összegzett bemenetétől.

$$e_j^{[s]} = -\frac{\partial E}{\partial I_j^{[s]}} = -\sum_{k=1}^{n^{[s+1]}} \frac{\partial E}{\partial I_k^{[s+1]}} \frac{\partial I_k^{[s+1]}}{\partial I_j^{[s]}} = \sum_{k=1}^{n^{[s+1]}} -\frac{\partial E}{\partial I_k^{[s+1]}} \frac{\partial I_k^{[s+1]}}{\partial I_j^{[s]}} =$$

Felismerve a $e_k^{[s+1]}$ jelölést a $-\dfrac{\partial E}{\partial I_k^{[s+1]}}$ -ban, egy rekurzív képlethez jutunk:

$$e_j^{[s]} = \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} \frac{\partial I_k^{[s+1]}}{\partial I_j^{[s]}}$$

s<r eset folytatása

Az $I_k^{[s+1]}$ függ az s-edik réteg j-edik neuronjának ($o_j^{[s]}$) kimenetétől, ezért $\frac{\partial I_k^{[s+1]}}{\partial I_j^{[s]}} = \frac{\partial I_k^{[s+1]}}{\partial o_j^{[s]}} \frac{\partial o_j^{[s]}}{\partial I_j^{[s]}}$

Tudva, hogy az $I_k^{[s+1]} = \sum_{j=1}^{n^{[s]}} w_{jk}^{[s+1]} o_j^{[s]}$ ezért $\frac{\partial I_k^{[s+1]}}{\partial o_j^{[s]}} = w_{ik}^{[s+1]}$

Továbbá az $o_j^{[s]} = f(I_j^{[s]})$ miatt $\frac{\partial o_j^{[s]}}{\partial I_j^{[s]}} = f'(I_j^{[s]}) = o_j^{[s]}(1- o_j^{[s]})$

Összeolvasva

$$e_j^{[s]} = \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} \frac{\partial I_k^{[s+1]}}{\partial I_j^{[s]}} = \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{ik}^{[s+1]} f'(I_j^{[s+1]}) =$$

$$= o_j^{[s]}(1- o_j^{[s]}) \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{ik}^{[s+1]}$$

Összesítve

Ha $s=r$ akkor

$$e_j^{[r]} = o_j^{[r]}(1-o_j^{[r]})(t_j-o_j^{[r]})$$

$$\Delta w_{ij}^{[r]} = \eta\, e_j^{[r]}\, o_i^{[r-1]}$$

Ha $s<r$ akkor

$$e_j^{[s]} = o_j^{[s]}(1- o_j^{[s]}) \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{ik}^{[s+1]}$$

$$\Delta w_{ij}^{[s]} = \eta\, e_j^{[s]}\, o_i^{[s-1]}$$

rekurzív szabályt kaptuk a súlyok módosítására.

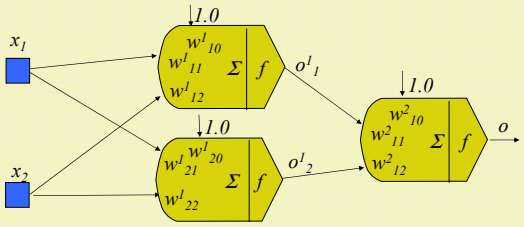
Algoritmus

1. Az $\underline{x}$ bemeneti vektorból kiindulva rétegenként kiszámoljuk a neuronok kimeneteit: $o_j^{[s]}$, így eljutunk a kimeneti réteg kimeneteihez $o_j^{[r]}$ is.
2. A kimeneti réteg minden neuronjára kiszámoljuk a lokális hibát: $e_j^{[r]} = o_j^{[r]}(1- o_j^{[r]})\,(t_j- o_j^{[r]})$, és a súlytényezők megváltozását: $\Delta w_{ij}^{[r]} = \eta\, e_j^{[r]} o_i^{[r-1]}$.
3. Rétegenként hátulról előre haladva számoljuk a neuronok hibáját: $e_j^{[s]} = o_j^{[s]}(1- o_j^{[s]}) \sum_{k=1}^{n^{[s+1]}} e_k^{[s+1]} w_{jk}^{[s+1]}$,

   és a súlytényező-változást: $\Delta w_{ij}^{[s]} = \eta\, e_j^{[s]} o_i^{[s-1]}$.
4. Módosítjuk a hálózat súlyait: $w_{ij}^{[s]} \leftarrow w_{ij}^{[s]} + \Delta w_{ij}^{[s]}$

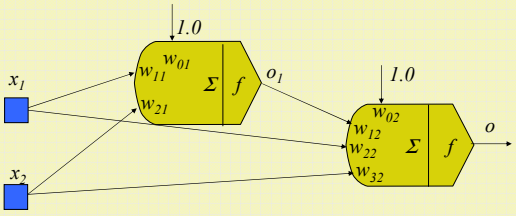
XOR művelet 1. példája

Beállítások: $x_1, x_2 \in \{0,1\}$ $f(x)=logiszt(x)$ $o^s_i \in (0,1)$
$w^s_{ij} = rand(-0.1 , 0.1)$ $o^s_0 = 1.0$ $\eta = 1.0$



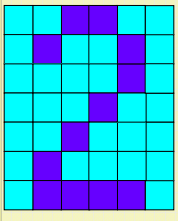
XOR művelet 2. példája

Beállítások: $x_1, x_2 \in \{0,1\}$ $f(x)=logiszt(x)$ $o_i \in (0,1)$
$w_{ij} = rand(-0.1 , 0.1)$ $o^s_0 = 1.0$ $\eta = 1.0$



Számjegy felismerés

Bemeneti értékek száma: 42 Kimenetek száma: 10

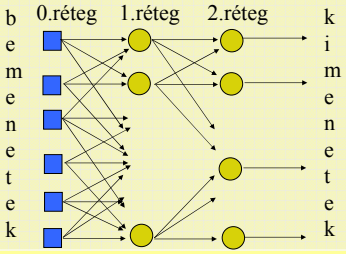


Minden számjegyhez tartozik egy kimenet

Közbülső réteg sejtjeinek száma: 11

Számjegy felismerés

Beállítások: $x_i \in \{0,1\}$ $f(x)=logiszt(x)$ $o^s_i \in (0,1)$
$w^s_{ij} = rand(-0.1 , 0.1)$ $o^s_0=1.0$ $\eta=0.35$



VI. Evolúciós algoritmusok

Evolúciós (genetikus) algoritmusok

- Egy adott pillanatban nem egyetlen lehetséges választ, hanem lehetséges válaszok (egyedek) halmazát, populációját tartjuk nyilván.Egy populáció annál jobb, minél inkább olyan egyedekkel rendelkezik, amelyek a kitűzött probléma helyes válaszai vagy azokhoz közeli lehetséges válaszok.
- A populációt lépésről lépésre próbáljuk meg egyre jobbra változtatni. A populáció megváltozása visszavonhatatlan. Ez tehát egy nem-módosítható stratégia.

Evolúciós algoritmus működése

- Először egy alkalmas kezdőpopulációt választunk.
- Minden lépésben
  - *Szelekció*: Kijelöljük szülőknek a rátermettebb egyedeket.
  - *Rekombináció (keresztezés)*: A szülőkből öröklődéssel utódokat állítunk elő.
  - *Mutáció*: az utódokat kismértékben módosítjuk.
  - *Visszahelyezés*: új populáció kialakítása
- A cél lehet egy keresett célegyed előállítása, vagy a populáció globális értékének változatlansága.

Példa

- Hol veszi fel az $f:[0 .. 1024] \rightarrow [-1,1]$ függvény a egész intervallumon a maximumát? (A f-et nem ismerjük, de az $f(x)$-t tetszőleges x-re ki tudjuk számolni)

x	*kód*	*f(x)*	*(f(x)+1)/(Σ+10)*	*rulett*
13	0000001101	0.22	0.10	2
53	0000110101	0.79	0.15	0
119	0001110111	0.87	0.15	1
339	0101010011	-0.35	0.05	0
358	0101100110	-0.03	0.08	1
482	0111100010	0.84	0.15	2
602	1001011010	-0.88	0.01	0
778	1100001010	0.84	0.15	2
841	1101001001	0.85	0.15	2
956	1110111100	-0.82	0.01	0

Össz:	2.33
Átl:	0.23
Max:	0.87

rulett	szelekció		rekombináció	mutáció
rulett	*x*	*kód*	*kód*	*kód*
2	13	0000001101	0000001001	0000001001
0	53	1101001001	1101001101	1101001101
1	119	0000001101	0000001010	0000001010
0	339	1100001010	1100001101	1100001101
1	358	0001110111	0000011100	0001011100
2	482	1100011100	1101110111	1101110111
0	602	0101100110	0101100010	0101100010
2	778	0111100010	0111100110	0111100110
2	841	0111100010	0111001001	0111001001
0	956	1101001001	1101100010	1101100000

u = v = populáció mérete

x	*kód*	*f(x)*
9	0000001001	0.15
845	1101001101	0.81
10	0000001010	0.17
781	1100001101	0.87
92	0001011100	0.99
887	1101110111	0.22
354	0101100010	-0.10
486	0111100110	0.80
457	0111001001	0.99
832	1101000000	0.92

Össz:	2.33
Átl:	0.23
Max:	0.87

→

Össz:	5.82
Átl:	0.58
Max:	0.99

Alapalgoritmus

Procedure EA
p ← *kezdeti populáció*
while *terminálási feltétel nem igaz* loop
  p' ← *szelekció(p)*
  p'' ← *rekombináció(p')*
  p''' ← *mutáció(p'')*
  p ← *visszahelyezés(p, p''')*
endloop

Algoritmus elemei

- Kódolás (egyed reprezentáció)
- Rátermettségi (fitnesz) függvény
  - Kapcsolat a célfüggvénnyel
- Evolúciós operátorok
  - szelekció, rekombináció (keresztezés), mutáció, visszahelyezés
- Kezdő populáció, Megállási feltétel
- Stratégiai paraméterek
  - populáció mérete, mutáció valószínűsége, utódképzési ráta, visszahelyezési ráta, stb.

Kódolás

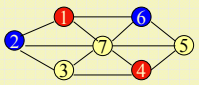
- Egy egyedet jelsorozattal (kromoszóma) kódolunk.
- A jelek vagy azok csoportjai (gén) írják le az egyed tulajdonságait: attribútum és érték.
- Sokszor egy jelnek a kódsorozatban elfoglalt pozíciója (lókusz) adja meg az attribútumot, a jel pedig az értéket (allél). Ilyenkor a kód szerkezete tulajdonságonkénti feldarabolhatóságot mutat.
- Gyakori megoldás:
  - valós (egész) számok tömbje, bináris tömb,
  - permutáció, fa-ábrázolás
- Kódolás és a rátermettségi függvény kapcsolata

Gráfszínezési példa kódolásai és rátermettségi függvényei

- Adott egy véges egyszerű gráf, amelynek a csúcsait a lehető legkevesebb szín felhasználásával úgy kell kiszínezni, hogy a szomszédos csúcsok eltérő színűek legyenek.

1. 2. 3. 4. 5. 6. 7.

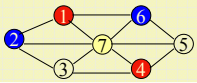
direkt



f = 10

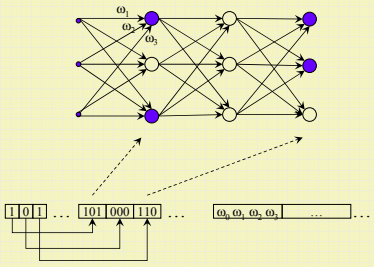
7	1	4	6	5	2	3

indirekt

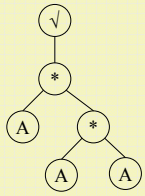


f = 5

Mesterséges neuronháló hierarchikus kódolása



Kifejezésfa kódolása



√ * A * A A

Szelekció

- **Célja**: a rátermett egyedek kiválasztása úgy, hogy a rosszabbak kiválasztása is kapjon esélyt.
- Néhány módszer:
  - **Rátermettség arányos**: a rátermettségi függvényre vagy annak skálázására épülő rulett kerék algoritmus
  - **Rangsorolásos**: rátermettség alapján sorba rendezett egyedek közül a kisebb sorszámúakat nagyobb valószínűséggel választja ki
  - **Versengő**: véletlenül kiválasztott egyedcsoportok (pl. párok) legjobb egyedét választja ki.
  - **Csonkolásos**: a rátermettség szerint legjobb valahány egyedből véletlenszerűen választ néhányat.

Rekombináció

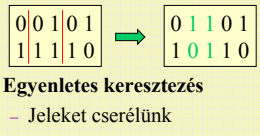
- A feladata az, hogy adott szülő-egyedekből olyan utódokat hozzon létre, amelyek a szüleik tulajdonságait "öröklik".
- Jellemzői
  - Egyszerűbb esetben jelcsoportok (gének) vagy jelek cseréjéről van szó (keresztezés), általános esetben azok transzformációjáról (rekombináció).
  - Ügyelni kell a kód-invariáns megtartására: vizsgálni kell, hogy az új kód értelmes lesz-e (permutáció)

Rekombináció tömbökre

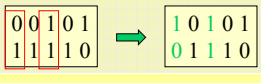
- **Köztes rekombináció**
  - A szülők ($\underline{x}$, $\underline{y}$) által kifeszített hipertégla valamelyik eleme lesz az utód ($\underline{u}$)
  - $\forall i=1\ldots n:\ u_i = a_i x_i + (1-a_i)y_i \quad a_i \in [-h,\ 1+h]$ véletlen
- **Lineáris rekombináció**
  - A szülők ($\underline{x}$, $\underline{y}$) által kifeszített hipertégla átlójának valamelyik eleme lesz az utód ($\underline{u}$)
  - $\forall i=1\ldots n:\ u_i = a x_i + (1-a)y_i \quad a \in [-h,\ 1+h]$ véletlen

Keresztezés

- **Egy- illetve többpontos keresztezés**
  - Kódszakaszokat cserélünk (az allélokat nem szerencsés kettévágni)

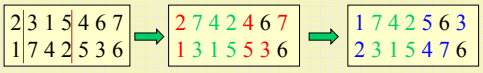


- **Egyenletes keresztezés**
  - Jeleket cserélünk



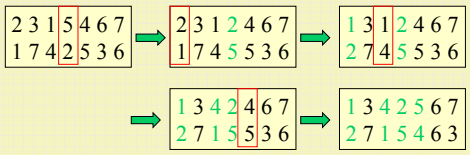
Permutációk keresztezése 1.

- **Parciálisan illesztett keresztezés**
  - Szakasz cseréje után cseréli a permutáció tulajdonságot sértő párokat.



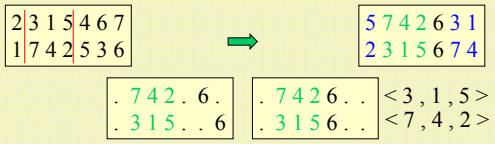
Permutációk keresztezése 2.

- **Ciklikus keresztezés**
  - $a_i \leftrightarrow b_i$; $a_j \leftrightarrow b_j$, ahol $a_j = b_i$ ($j \neq i$); stb.

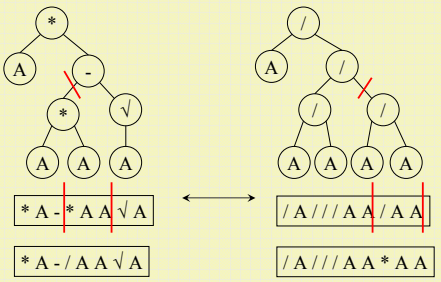


Permutációk keresztezése 3.

- **Rendezett keresztezés**
  - illesztési szakaszon mindent cserélünk, a problémát nem okozó elemeket ciklikusan felzárkóztatjuk az illesztési szakasz jobb széléhez, majd a hiányzó elemeket beírjuk azok szülőbeli sorrendjében.



Kifejezésfa



Mutáció

- A mutáció egy egyed (utód) kis mértékű véletlen változtatását, finom közelítését végzi.
- Valós tömbbel való kódolásnál kis p valószínűséggel:
  - $\forall i=1 \ldots n : z_i = x_i \pm range_i * p$
- Bináris tömbbel való kódolásnál kis p valószínűséggel:
  - $\forall i=1 \ldots n : z_i = 1 - x_i$
- Permutáció esetén kis valószínűséggel választott pozíció-párra
  - csere; a pozíciók közötti szakaszon a jelek léptetése, a jelek sorozatának megfordítása, esetleg átrendezése.

Visszahelyezés

- A visszahelyezés a populációnak az utódokkal történő frissítése. Kiválasztja a populációnak a lecserélendő egyedeit (szelekció), és azok helyére az utódok közül választ (szelekció).

$$\text{utódképzési ráta} = \frac{\text{utódok száma}}{\text{populáció száma}}$$

$$\text{visszahelyezési ráta} = \frac{\text{lecserélendő egyedek száma}}{\text{populáció száma}}$$

  - ha u=v, akkor feltétlen cseréről van szó
  - ha u<v , akkor egy utód több példányban is bekerülhet
  - ha u>v, akkor az utódokat szelektáljuk

VII. Rezolúciós következtetés

Emlékeztető: Elsőrendű logika nyelve

- Elsőrendű logika szintaxisa
  - Jelkészlet és kifejezések
- Elsőrendű logika szemantikája
  - Interpretáció és kiértékelés
- Kielégíthetőségi tulajdonságok
  - érvényes, kielégíthető, kielégíthetetlen
- Logikailag ekvivalens formulák
- Logikai következmény

elemváltozó, konstans szimb. függvény ill. predikátum szimb. műveleti jelek, kvantorok elválasztó jelek, zárójelek

term, atomi formula, formula

Jelkészlet

- Elválasztó jelek
  - vessző: ,
  - zárójelpárok: ( ), [ ], { }
- Logikai műveleti jelek
  - $\neg$ a negáció jele (“nem”)
  - $\wedge$ a konjunkció jele (“és”)
  - $\vee$ a diszjunkció jele (“vagy”)
  - $\rightarrow$ az implikáció jele (“ha ... akkor”)
  - $\leftrightarrow$ az ekvivalencia jele (“akkor és csak akkor”)

Jelkészlet

- Kvantorok
  - $\forall$ univerzális kvantor (“minden”)
  - $\exists$ egzisztenciális kvantor (“van olyan”)
- Elemváltozók vagy változók ($x, y, z, \ldots$)
- Elemkonstansok vagy konstansok ($a, b, c, \ldots$)
- Függvényszimbólumok ($f, g, h, \ldots$ vagy f^n)
- Ítéletváltozók ($p, q, r, \ldots$)
- Predikátumszimbólumok vagy logikai függvények ($P,Q, R, \ldots$ vagy P^n.)

Kifejezés: term

- Minden elemkonstans term.
- Minden elemváltozó term.
- Ha f egy n-argumentumú függvényszimbólum, és a $t_1, \ldots, t_n$ termek, akkor $f(t_1,\ldots,t_n)$ is term.

Kifejezés: atomi formula

- Minden ítéletváltozó atomi formula.
- Ha P egy n-argumentumú predikátumszimbólum, és a $t_1, \ldots, t_n$ termek, akkor $P(t_1,\ldots,t_n)$ atomi formula.

Kifejezés: jólformált formula

- Minden atomi formula egyben formula.
- Ha A és B formulák, akkor a $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$ és $(A \leftrightarrow B)$ kifejezések is formulák.
- Ha A egy formula, és x egy változó, akkor a $\forall xA$ és a $\exists xA$ kifejezések is formulák.

Megjegyzés

- A fentiek egy rekurzív, és teljes képzési szabályt adnak a formulák készítésére.
- A zárójelek elhagyását a precedencia-sorrend megadása teszi lehetővé:
  - $\forall, \exists, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$
- A $\forall xA$ és $\exists xA$ formulákban szereplő x változó az A részformulában univerzálisan, illetve egzisztenciálisan kötött (kvantált)
- A kvantorok hatásköre az A részformula.
- A továbbiakban nem használunk nem kötött (szabad) változókat: $\forall x(P(x,y) \wedge \exists y Q(x,y))$

Interpretáció

- Megválasztjuk az értelmezés nem üres U alaphalmazát, más néven univerzumát.
- Minden konstansszimbólumnak megfeleltetünk egy U -beli elemet.
- Minden n-argumentumú függvényszimbólumhoz hozzárendelünk egy $U^n \rightarrow U$ leképezést.
- Minden n-argumentumú predikátumszimbólumhoz hozzárendelünk egy $U^n \rightarrow \{i,h\}$ leképezést.

Kiértékelés

- Egy formula igazságértékét határozza meg.
- Ha egy formula $\neg A$, vagy $A \wedge B$, vagy $A \vee B$, vagy $A \rightarrow B$, vagy $A \leftrightarrow B$ alakú, és az A és B formulák igazságértéke már ismert, akkor az igazságértékét a műveleti jelek igazságtáblái alapján számoljuk ki.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \rightarrow B$	$A \leftrightarrow B$
i	i	h	i	i	i	i
i	h		h	i	h	h
h	i	i	h	i	i	h
h	h		h	h	i	i

Kiértékelés

- A $\forall xA$ alakú formula pontosan akkor *igaz* egy adott interpretációban, ha bárhogyan is választunk ki az U-ból egy e elemet, amellyel lecserélve az x változó összes előfordulását az A-ban olyan formulát kapunk ($A^{x \leftarrow e}$), amely értéke igaz.
- A $\exists xA$ alakú formula pontosan akkor *igaz* egy adott interpretációban, ha van olyan e eleme az U-nak, amellyel lecserélve az x változó összes előfordulását az A-ban olyan formulát kapunk ($A^{x \leftarrow e}$), amely értéke igaz.

Példák

- $\forall x[P(f(x,x),a) \rightarrow P(x,a)]$
  - $U \sim$ valamelyik(?) számhalmaz

1. $a \sim 1$, $f(x,y) \sim x*y$, $P(x,y) \sim x=y$
   - $\forall x(x^2=1 \rightarrow x=1)$

   Ez a természetes számok körében *igaz*, az egészek körében *hamis* állítás.
2. $a \sim 0$, $f(x,y) \sim x*y$, $P(x,y) \sim x>y$,
   - $\forall x(x^2>0 \rightarrow x>0)$

   Ez a természetes számok körében *igaz*, az egészek körében *hamis* állítás.

Kielégíthetőségi tulajdonságok

- *Kielégíthető* formulának van olyan interpretációja (modellje), amely mellett *igaz* az értéke.
  - $\forall x[P(f(x,x),a) \rightarrow P(x,a)]$
- *Érvényes* formula igazságértéke minden interpretációban *igaz*.
  - $\forall x P(x) \vee \exists y \neg P(y)$
- *Kielégíthetetlen* formula igazságértéke minden interpretációban *hamis*.
  - $\forall x P(x) \wedge \exists y \neg P(y)$

Logikai ekvivalencia

- Két formulát akkor mondunk ekvivalensnek, ha minden interpretációban megegyezik az igazságértékük.
- A nevezetesebb ekvivalenciákat logikai törvényként tartjuk nyilván
- Jele: $\equiv$

Logikai törvények

- *1.* $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$
- *2.* $A \rightarrow B \equiv \neg A \vee B$
- *3.* $A \wedge B \equiv B \wedge A$, $A \vee B \equiv B \vee A$
- *4.* $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$, $(A \vee B) \vee C \equiv A \vee (B \vee C)$
- *5.* $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$, $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
- *6.* $A \wedge T \equiv A$, $A \vee T$ érvényes, ha T érvényes
- *7.* $A \vee F \equiv A$, $A \wedge F$ kielégíthetetlen, ha F kielégíthetetlen
- *8.* $A \wedge \neg A$ kielégíthetetlen, $A \vee \neg A$ érvényes

Logikai törvények

- *9.* $\neg\neg A \equiv A$
- *10.* $\neg(A \wedge B) \equiv \neg A \vee \neg B$, $\neg(A \vee B) \equiv \neg A \wedge \neg B$
- *11.* $A \wedge (A \vee B) \equiv A$, $A \vee (A \wedge B) \equiv A$
- *12.* $A \wedge A \equiv A$, $A \vee A \equiv A$
- *13.* $Qx A(x) \wedge B \equiv Qx(A(x) \wedge B)$, $Qx A(x) \vee B \equiv Qx(A(x) \vee B)$
- *14.* $\neg\forall x A(x) \equiv \exists x \neg A(x)$, $\neg\exists x A(x) \equiv \forall x \neg A(x)$
- *15.* $Q_1 x A(x) \wedge Q_2 x B(x) \equiv Q_1 x\, Q_2 y(A(x) \wedge B(y))$
- *16.* $Q_1 x A(x) \vee Q_2 x B(x) \equiv Q_1 x\, Q_2 y(A(x) \vee B(y))$
  - Q_1 és Q_2 tetszőleges kvantorokat jelöl

Logikai törvények

- *17.* $\forall x A(x) \wedge \forall x B(x) \equiv \forall x(A(x) \wedge B(x))$
- *18.* $\exists x A(x) \vee \exists x B(x) \equiv \exists x(A(x) \vee B(x))$

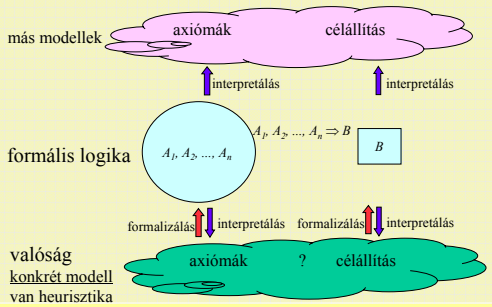
  - Ennek két ekvivalenciának nincsen szimmetrikus párja!

$$\forall x A(x) \vee \forall x B(x) \neq \forall x(A(x) \vee B(x))$$
$$\exists x A(x) \wedge \exists x B(x) \neq \exists x(A(x) \wedge B(x))$$

Logikai következmény

- Egy B formulát az $A_1, A_2, \ldots, A_n$ formulák logikai következményének nevezzük, ha a B *igaz* minden olyan interpretációban, amelyben $A_1, A_2, \ldots, A_n$ mindegyike *igaz*.
- A B formula akkor és csak akkor logikai következménye az $A_1, A_2, \ldots, A_n$ formuláknak, ha az $A_1 \wedge A_2 \wedge \ldots \wedge A_n \rightarrow B$ formula érvényes.
- A B formula akkor és csak akkor logikai következménye az $A_1, A_2, \ldots, A_n$ formuláknak, ha az $A_1 \wedge A_2 \wedge \ldots \wedge A_n \wedge \neg B$ formula kielégíthetetlen.
- $A_1, A_2, \ldots, A_n \Rightarrow B$
  - az $A_1, A_2, \ldots, A_n$ formulák a feltételek vagy axiómák
  - a B a következmény vagy célállítás.

MI nézőpontja



1. Rezolúció

Feladat:

A_1: Ha süt a nap, akkor Péter strandra megy.
A_2: Ha Péter strandra megy, akkor úszik.
A_3: Péternek nincs lehetősége otthon úszni.

Lássuk be, hogy ezekből következik:

B: Ha süt a nap, akkor Péter nem marad otthon.

Formalizálás:

A_1:	$p \to q$	– süt a nap:	p
A_2:	$q \to r$	– Péter strandra megy:	q
A_3:	$\neg(s \wedge r)$	– Péter úszik:	r
		– Péter otthon marad:	s
B:	$p \to \neg s$		

Átalakítás

- Kell: $p \to q,\ \ q \to r,\ \ \neg(s \wedge r) \ \Rightarrow\ p \to \neg s$
- azaz $(p \to q) \wedge (q \to r) \wedge \neg(s \wedge r) \to (p \to \neg s)$ formula érvényes,
- azaz $(p \to q) \wedge (q \to r) \wedge \neg(s \wedge r) \wedge \neg(p \to \neg s)$ formula kielégíthetetlen,
- azaz $(\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg(s \wedge r) \wedge \neg(\neg p \vee \neg s))$ formula kielégíthetetlen,
- azaz $(\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg s \vee \neg r) \wedge p \wedge s$ formula kielégíthetetlen.

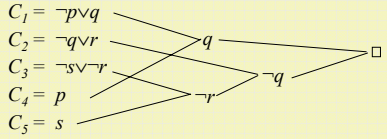
Bizonyítandó

- A
  - $C_1 = \neg p \vee q$
  - $C_2 = \neg q \vee r$
  - $C_3 = \neg s \vee \neg r$
  - $C_4 = p$
  - $C_5 = s$
- úgynevezett klózok között mindig (minden interpretációban) akad hamis értékű

Indirekt bizonyítás

- Tegyük fel, hogy van olyan interpretáció, amikor mindegyik klóz igaz.
- Például $C_4(p)$ is, és $C_1(\neg p \vee q)$ is.
- C_4 *igaz, ezért* a p is, de ekkor a $\neg p$ *hamis*. A C_1 csak úgy lehet *igaz*, ha a q *igaz*.
- Legyen q egy új klóz (C_6), amelynek csakúgy igaznak kell lenni az adott interpretációban, mint a többi klóznak.

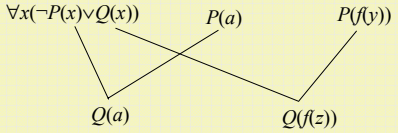
Rezolúciós eljárás



Tehát ha süt a nap, akkor Péter nem marad otthon.

Elsőrendű rezolúció problémái

- Elsőrendben egy kicsit bonyolultabb a rezolúció
  - a klóz-formára hozás (a kvantorok miatt)
  - azonos alakú komplemens literálpár kiválasztása változó helyettesítéssel



Formulák klóz-formára hozása

1. Küszöböljük ki az $\leftrightarrow$ és a $\rightarrow$ műveleti jeleket.
   - (A $\neg$, $\wedge$, $\vee$ műveleti jelekkel helyettesítsük őket.)
2. Redukáljuk a negációk hatáskörét.
   - (DeMorgan törvények)
3. A változók standardizálása
   - (kvantoronkénti átnevezése).
   - Például a $\forall x(P(x) \vee \exists x Q(x))$ formulából $\forall x(P(x) \vee \exists y Q(y))$

4. Küszöböljük ki az egzisztenciális kvantorokat

- Nem ekvivalens, hanem a kielégíthetőséget tartó átalakítás.
- A $\exists x P(x)$ pontosan akkor kielégíthető, ha a $P(a)$ kielégíthető. (Az a egy Skolem-konstans)
- A $\forall x \exists y P(x,y)$ pontosan akkor kielégíthető, ha a $\forall x P(x,g(x))$ kielégíthető. (Skolem-függvény: $g(x)$)
- A $\forall x_1 ... \forall x_k \exists y \, Q_1 z_1 ... Q_r z_r \, A(x_1,.. ,x_k,y,z_1,.., z_r)$ pontosan akkor kielégíthető, ha a $\forall x_1 ... \forall x_k \, Q_1 z_1 ... Q_r z_r \, A(x_1,.., x_k,g(x_1,.., x_k),z_1,.. ,z_r)$ kielégíthető. (Skolem-függvény: $g(x_1,.., x_k)$)
  - Skolem-függvény argumentum száma

5. A kvantorok kiemelése a sorrendjük megtartása mellett, majd a kvantorok elhagyása.
   (**prenex normálforma**: prefixum és mátrix)
6. Hozzuk a formula mátrixát konjunktív normálformára.
   (disztributív törvények)
7. A konjunkció műveleti jeleit elhagyva alakítsuk ki a klózhalmazt.
   Literál, pozitív ill. negatív literál, komplemens literálpár, klóz.
8. Nevezzük át a változókat úgy, hogy egyetlen változó se forduljon elő két klózban. (Ez a lépés késleltethető)

Helyettesítés

- A változó-term rendezett párokat tartalmazó $\alpha=\{v_1|t_1, ..., v_n|t_n\}$ halmazt helyettesítésnek nevezzük, ha $v_1, ..., v_n$ egymástól különböző változókat jelölnek, és $t_i \neq v_i$ $(1 \leq i \leq n)$.
- Az üres halmazt üres helyettesítésként értelmezzük.
- Példa: $A = P(x,f(x),y)$ $\alpha = \{x|y, y|f(y)\}$
  $A\alpha = P(y,f(y),f(y))$

Helyettesítés alkalmazása

- Az $A\alpha$ kifejezést ($\alpha=\{v_1|t_1, ..., v_n|t_n\}$) az A kifejezés egy példányának nevezzük, amelyet úgy képezünk, hogy A-ban a $v_1, ..., v_n$ szabad változók minden előfordulását egyidejűleg rendre $t_1, ..., t_n$ -nel helyettesítjük.
- Pontosabban
  - Ha $\alpha=\{v|t\}$ és $v \notin t$, akkor $A\alpha$-t úgy kapjuk, hogy benne a v minden szabad előfordulását kicseréljük t-re.
  - Ha $\alpha=\{v|t\}$, ahol $v \in t$, és z olyan változó, hogy $z \notin A$, $z \neq v$, és $z \notin t$, akkor $A\alpha=A\{v|s\}\{z|v\}$, ahol $s= t\{v|z\}$.
  - Ha $\alpha=\{v_1|t_1, ..., v_n|t_n\}$, és z olyan változó, hogy $z \notin A$, $z \neq v_i$, és $z \notin t_i$ $(1 \leq i \leq n)$, akkor az $s_i= t_i\{v_n|z\}$ $(1 \leq i \leq n)$ jelölés mellett $A\alpha=A\{v_1|s_1, ..., v_{n-1}|s_{n-1}\}\{v_n|s_n\}\{z|v_n\}$.

Kompozíció

- Az $\alpha = \{x_1|t_1, ..., x_n|t_n\}$ és $\beta = \{y_1|u_1, ..., y_m|u_m\}$ helyettesítések $\alpha\beta$ kompozícióját úgy kapjuk, hogy
  - az $\{x_1|t_1\beta, ..., x_n|t_n\beta, y_1|u_1, ..., y_m|u_m\}$ halmazból töröljük azokat az $x_i|t_i\beta$ elemeket, amelyekre $x_i=t_i\beta$, és
  - azokat az $y_j|u_j$ elemeket, amelyekre az y_j megegyezik $x_1, ..., x_n$ valamelyikével.
- Megj: $W\alpha\beta = (W\alpha)\beta$
- Példa: $\alpha = \{x|f(y), y|z\}$ és $\beta = \{x|a, y|b, z|y\}$
  $\{x|f(b), y|y, x|a, y|b, z|y\}$,
  $\alpha\beta = \{x|f(b), z|y\}$

Legáltalánosabb egyesítő

- Az $A_1, A_2, ..., A_n$ kifejezéseket egyesíthetőnek nevezzük, ha van olyan α helyettesítés, amelyre $A_1\alpha=A_2\alpha=...=A_n\alpha$.
- Ekkor az α az $A_1, A_2, ..., A_n$ kifejezéseknek egy egyesítője.
- Legáltalánosabb egyesítőnek nevezzük az $A_1, A_2, ..., A_n$ kifejezéseknek egy δ egyesítőjét, ha bármely α egyesítő előállítható $\alpha=\delta\alpha'$ formában. (α' egy alkalmas helyettesítés)
- Egyesítő nem mindig létezik.
- Ha van egyesítő, akkor legáltalánosabb is van
- A legáltalánosabb egyesítő nem egyértelmű:
  - $x|y$ vagy $y|x$

Különbségi halmaz

- Az $A_1, A_2, ..., A_n$ kifejezések különbségi halmazát úgy kapjuk meg, hogy
  - először meghatározzuk balról jobbra az első olyan pozíciót, amelyen nem egyezik meg az összes A_i formula ($1 \le i \le n$),
  - majd vesszük az összes, ezen pozíción kezdődő termet.
- Példa: $P(x,g(f(y,z), x)$,
  $P(x,g(a,g(y,z)))$,
  $P(x,g(a,b))$

  $D=\{f(y,z), a\}$

Egyesítő algoritmus

Procedure Egyesítés(H) *return* helyettesítés or *hiba*
1. $\delta \leftarrow \{\}$
2. *loop*
3. $D \leftarrow H$ különbségi halmaza
4. *if* D üres *then return* δ
5. *if* $\neg\exists\ v,t \in D$, ahol v változó, t term, és $v \notin t$
   then return hiba
6. *select* $v,t \in D$, ahol v változó, t term, és $v \notin t$
7. $\delta \leftarrow \delta\{v|t\}$
8. $H \leftarrow H\{v|t\}$
9. *endloop*

end

Példa

- $A_1 = P(x, u, f(g(x)))$ $A_2 = P(a, y, f(y))$
  - $\delta = \{\}$
- 1. iteráció
  - $A_1=P(x, u, f(g(x)))$ $A_2=P(a, y, f(y))$ $D=\{x, a\}$
  - $\delta \leftarrow \delta\{x|a\}$ $A_1 \leftarrow A_1\{x|a\}$ $A_2 \leftarrow A_2\{x|a\}$
  - $\delta = \{x|a\}$
- 2. iteráció
  - $A_1=P(a,u, f(g(a)))$ $A_2=P(a, y, f(y))$ $D=\{u, y\}$
  - $\delta \leftarrow \delta\{u|y\}$ $A_1 \leftarrow A_1\{u|y\}$ $A_2 \leftarrow A_2\{u|y\}$
  - $\delta = \{x|a, u|y\}$

Példa

- 3. iteráció
  - $A_1=P(a,y, f(g(a)))$ $A_2=P(a, y, f(y))$ $D=\{g(a), y\}$
  - $\delta \leftarrow \delta\{y|g(a)\}$ $A_1 \leftarrow A_1\{y|g(a)\}$ $A_2 \leftarrow A_2\{y|g(a)\}$
  - $\delta = \{x|a, u|g(a), y|g(a)\}$
- 4. iteráció
  - $A_1=P(a,g(a),f(g(a)))$ $A_2=P(a,g(a),f(g(a)))$ $D=\{\}$

Rezolválható klózok

- A C_1 és C_2 klózt akkor mondjuk rezolválhatónak, ha egyrészt található bennük olyan komplemens literálpár (pl. C_1-ben P, C_2-ben pedig $\neg P$), pontosabban ezeknek olyan r, illetve s számú előfordulása, hogy
  - $C_1 = P(t_{11},...,t_{1n}) \vee ... \vee P(t_{r1},...,t_{rn}) \vee C_1'$
  - $C_2 = \neg P(u_{11},...,u_{1n}) \vee ... \vee \neg P(u_{s1},...,u_{sn}) \vee C_2'$
- Másrészt (a szükséges változó helyettesítés után) a $P(t_{11},...,t_{1n})$, ..., $P(t_{r1},...,t_{rn})$, $P(u_{11},...,u_{1n})$, ..., $P(u_{s1},...,u_{sn})$ formulák egyesíthetők.

Rezolvens klóz

- Ha a C_1 és C_2 klózok egy δ legáltalánosabb egyesítő mellett rezolválhatók, akkor a rezolvens klózt az alábbi módon képezzük:
- $R(C_1 , C_2) = C_1'\delta \vee C_2'\delta$.
- Az üres klóz ($\square$) a kielégíthetetlen klóz.

Megjegyzés

- Az egyesítésben a literálok mind a negáció jele nélkül szerepelnek.
- A P predikátum előfordulhat a C_1' részben, a $\neg P$ pedig a C_2' részben
- Lehet a C_1' és a C_2' rész üres is.

Példa rezolúciós lépésre

$C_1 = \neg P(x_1,a) \vee \neg P(x_1,y_1) \vee \neg P(y_1,x_1)$
$C_2 = P(x_2,f(x_2)) \vee P(x_2,a)$

1. $P(x_1,y_1)$, $P(y_1,x_1)$, $P(x_2,a)$ formulák egyesíthetők.
$\delta = \{x_1|a, y_1|a, x_2|a\}$
Rezolvens: $\neg P(a,a) \vee P(a, f(a))$
2. $P(y_1,x_1)$, $P(x_2,f(x_2))$ formulák is egyesíthetők.
$\delta = \{y_1|x_2 , x_1|f(x_2)\}$
Rezolvens: $\neg P(f(x_2),a) \vee \neg P(f(x_2),x_2) \vee P(x_2,a)$

Példa:Kuruzslók-e a doktorok?

A_1: Van olyan páciens, aki minden doktorban megbízik.
A_2: A kuruzslókban egyetlen páciens sem bízik meg.

Lássuk be, hogy az A_1 és A_2 állításoknak logikai következménye a
B : Egyetlen doktor sem kuruzsló.

Formalizálás

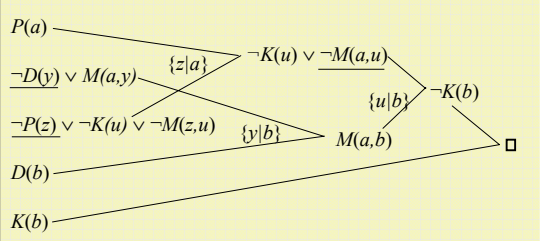
$P(x)$: az x egy páciens
$D(y)$: az y egy doktor
$K(y)$: az y egy kuruzsló
$M(x,y)$: az x megbízik az y-ban

$A_1 = \exists x\{P(x) \wedge \forall y[D(y) \rightarrow M(x,y)]\}$
$A_2 = \forall x\{P(x) \rightarrow \forall y[K(y) \rightarrow \neg M(x,y)]\}$
$B = \forall x[D(x) \rightarrow \neg K(x)]$

Átalakítás

1. $\exists x[P(x)\wedge\forall y(\neg D(y) \vee M(x,y))] \wedge\forall x[\neg P(x) \vee \forall y(\neg K(y) \vee \neg M(x,y))] \wedge\neg\forall x(\neg D(x) \vee \neg K(x))$
2. $\exists x[P(x)\wedge\forall y(\neg D(y) \vee M(x,y))] \wedge\forall x[\neg P(x) \vee \forall y(\neg K(y)\vee \neg M(x,y))] \wedge \exists x(D(x) \wedge K(x))$
3. $\exists x[P(x)\wedge\forall y(\neg D(y) \vee M(x,y))] \wedge \forall z[\neg P(z) \vee \forall u(\neg K(u) \vee \neg M(z,u))] \wedge \exists v(D(v) \wedge K(v))$
4. $[P(a)\wedge\forall y(\neg D(y) \vee M(a,y))] \wedge \forall z[\neg P(z) \vee \forall u(\neg K(u) \vee \neg M(z,u))] \wedge (D(b) \wedge K(b))$
5. $\forall y\ \forall z\ \forall u\ \{[P(a)\wedge (\neg D(y) \vee M(a,y))] \wedge [\neg P(z) \vee (\neg K(u) \vee \neg M(z,u))] \wedge (D(b) \wedge K(b))\}$
6. $P(a)\wedge (\neg D(y) \vee M(a,y)) \wedge [\neg P(z) \vee \neg K(u) \vee \neg M(z,u)] \wedge D(b) \wedge K(b)$

Rezolúciós eljárás



Változó átnevezésre nem volt szükség

Példa: Léteznek-e borbélyok?

A borbélyok az összes olyan embert megborotválják, akik önmaguk nem borotválkoznak.

Nincs olyan borbély, aki egy önmagát borotváló embert borotválna.

Tehát nincsenek borbélyok.

$\forall x\ [B(x)\rightarrow\forall y(\neg S(y,y) \rightarrow S(x,y))]$
$\neg\exists x[B(x) \wedge\exists y(\ S(y,y) \wedge S(x,y))]$
$\neg\exists x\ B(x)$

$S(x,y)$~ x megborotválja y -t
$B(x)$ ~ x egy borbély

$\neg B(x_1) \vee S(y_1,y_1) \vee S(x_1,y_1)$
$\neg B(x_2) \vee \neg S(y_2,y_2) \vee \neg S(x_2,y_2)$
$B(a)$

$\{x_1\setminus y_2,\ x_2\setminus y_2,\ y_1\setminus y_2\}$ → $\neg B(y_2)$ → □ $\{y_2\setminus a\}$

Rezolúció

Procedure Rezolúció($A_1, A_2, ..., A_n \Rightarrow B$) return *logical*

1. $KLÓZOK \leftarrow$ az $A_1, A_2, ..., A_n$ és $\neg B$ formulák klózformája.
2. *loop*
3. *if* $\square \in KLÓZOK$ *then return true*
4. *if* $\neg\exists\ C_1,C_2 \in KLÓZOK$, amelyek rezolválhatók, és még nem ismert minden rezolvensük
   then return false
5. *select* $C_1,C_2 \in KLÓZOK$, amelyeknek $R(C_1,C_2)$ egy még nem ismert rezolvense
6. $KLÓZOK \leftarrow KLÓZOK \cup R(C_1,C_2)$
7. *endloop*

end

Rezolúció tulajdonságai

- A rezolúció helyes eljárás, azaz ha az üres klóz megtalálásával terminál, akkor a kiinduló klózhalmaz kielégíthetetlen.
  - $C_1, C_2 \Rightarrow R(C_1, C_2)$ illetve $C_1 \wedge C_2 \equiv C_1 \wedge C_2 \wedge R(C_1, C_2)$
- A rezolúció teljes, azaz kielégíthetetlen klózhalmazból véges lépésben levezethető az üres klóz
  - S-nek H_S (Herbrand univ.) feletti Herbrand bázisa:$H_S(S)$ (alapklózok)
  - S kielégíthetetlen → $H_S(S)$ egy véges S' részhalmaza kielégíthetetlen → a rezolúció talál S'-ben ellentmondást → a rezolúció S-ben is talál ellentmondást.
- Az elsőrendű predikátumkalkulus parciálisan eldönthető. $\{\neg P(x)\ ,\ P(y)\vee\neg P(f(y)),\ P(a)\ \}$

Nemmódosítható kereső rendszer

- A globális munkaterület: előállított klózok (KLÓZOK)
- a kiindulási érték: kiindulási klózok
- a terminálási feltétel:
  - sikeres: üres klóz;
  - sikertelen: nincs új rezolvens
- kereső szabály: rezolvens képzés (R)
- vezérlési stratégia: nem-módosítható

Nemdeterminisztikusság

- A rezolúció algoritmusa három okból sem determinisztikus:
  - Egy lépésben több klózpárt rezolválhatuk.
  - Két klózban több komplemens literálpár szerepelhet.
  - Rögzített klózpár és komplemens literálpár esetén is többféleképpen rezolválhatunk

2. Rezolúciós stratégiák

- Rezolúciós stratégiának nevezünk bármilyen, a rezolúció alapalgoritmusát kiegészítő olyan előírást, amely
  - korlátozza egy adott pillanatban előállítható rezolvensek körét, illetve
  - szabályozza a rezolvens képzés sorrendjét.
- Másodlagos vezérlési stratégiák
- A korlátozásnál megsérülhet a teljesség:
  - nem vezethető le minden esetben az üres klóz

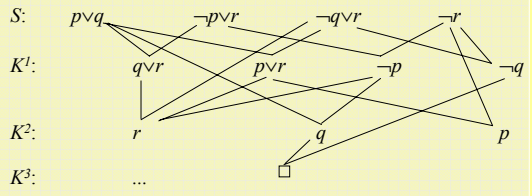
Rezolúciós gráf

- Irányított gráf (csúcs = klóz), ahol egy csúcsnak – a kiinduló klózok csúcsainak kivételével - két szülője van.
- A csúcsokhoz szintszám rendelhető:

  $R(K_1, K_2)$ ~ azok a klózok, amelyek egy K_1 és egy K_2 -beli klóz rezolvenseként állnak elő.

$K^0 = S$	$R^0 = S$
$K^1 = R(S, S)$	$R^1 = K^1 \cup S$
$K^2 = R(K^1, R^1)$	$R^2 = K^2 \cup R^1$
$K^{i+1} = R(K^i, R^i)$	$R^{i+1} = K^{i+1} \cup R^i$

Példa



- Elnevezések:
  - Rezolúciós gráf, bizonyítási gráf, cáfolati gráf

Sorrendi stratégiák

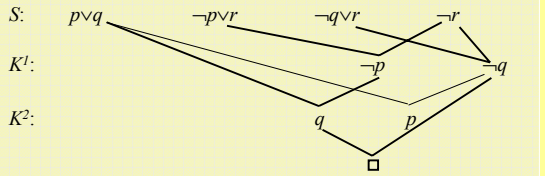
- A *szélességi stratégia* a rezolvenseket a rezolúciós gráf szintjei szerinti sorfolytonos sorrendben állítja elő. Az egyes szinteken véges sok klóz áll, amelyek előállítási sorrendje nincs meghatározva.
- A *klózok hossza (literálok száma) szerinti* sorrendnél a rezolválható klózpárok közül mindig a legrövidebbet rezolváljuk.
  - Két klózpárt először a hosszösszegük alapján hasonlítunk össze, ha ezek nem döntenek, akkor a rövidebbik klózaik hossza a mérvadó, ha ez is azonos, akkor a másik két klóz hossza dönt.

Szűkítő stratégiák

- A rezolúciós gráf méretét szűkítjük:
  - Egységklóz stratégia
  - Lineáris input stratégia
  - Ősre korlátozott stratégia
  - Támogató-halmaz stratégia
- Egyszerűsítő stratégiák
  - Érvényes klózok elhagyása
  - Befoglaló klózok elhagyása
  - Idegen literált tartalmazó klózok elhagyása

Egységklóz stratégia

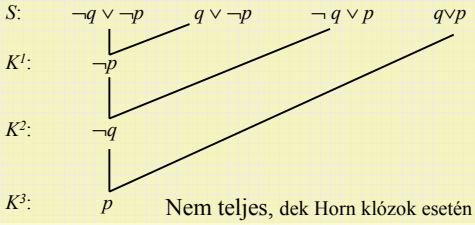
- Az egységklóz stratégiában a rezolválandó klózpár egyike egyetlen literál.



Nem teljes, de Horn klózok esetén az.
Horn klózok: legfeljebb egy pozitív literált tartalmazó klózok.

Lineáris input stratégia

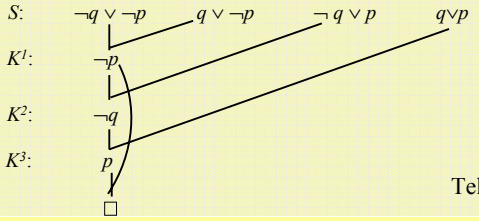
- A lineáris input stratégia esetén a rezolvens klóz egyik szülője az előző lépésben előállított klóz, másik szülőjét a kiinduló klózhalmazból kell venni.



Nem teljes, dek Horn klózok esetén az.

Ősrekorlátozott stratégia

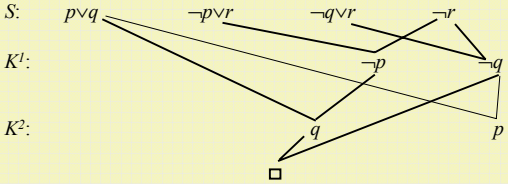
- Az ősrekorlátozott lineáris input stratégia esetén a rezolvens klóz egyik szülője a kiinduló klózhalmazból való, vagy őse a másik szülőnek.



Teljes

Támogató-halmaz stratégia

- A támogató-halmaz stratégia során a rezolválandó klózpár egyike a kiinduló klózok egy megadott T halmazából származik.



Ha S–T kielégíthető, akkor a stratégia **teljes**,
Érdemes T -t a célállítás klózainak választani.

Érvényes klózok elhagyása

- Az érvényes klóz minden interpretációban *igaz* értékű kifejezés, ezért a rezolúciós indirekt bizonyításnál nem alkalmas a cáfolat előállítására, azaz az üres klóz levezetésére.
- Példák:
  - $P(x)\vee B(y)\vee\neg B(y)$
  - $P(f(a))\vee Q(x)\vee\neg P(f(a))$
  - Vigyázat: a $P(x)\vee\neg P(y)$ nem érvényes klóz

Befoglaló klózok elhagyása

- A D klóz magában foglalja a C klózt, vagy másként D a C befoglaló klóza, ha létezik olyan α helyettesítés, amelyre $C\alpha$ része D-nek. Ha egy cáfolathoz kell egy befoglaló klóz, akkor létezik olyan cáfolat is, amelyik a befoglalt klózt használja fel.
- Példák:
  - $P(x)$ befoglaló klóza a $P(y)\vee Q(z)$
  - $P(x)$ befoglaló klóza a $P(a)$

Idegen literált tartalmazó klózok elhagyása

- Egy literált idegennek nevezünk egy klózhalmazban, ha nem lehet rezolválni egyetlen másik klóz megfelelő, komplemens literáljával. Egy ilyen klóz nem vesz részt a cáfolatban.

Procedurális hozzárendelés

- Egy adott modellben (interpretációban) egy függvényszimbólum alappéldánya kiértékelhető, és helyettesíthető az eredményt szimbolizáló konstansszimbólummal.
- Pl: Ha a $P(f(n,1))$ formula konkrét modellje az egész számokról szól, ahol f a kivonás művelete, akkor a $P(f(n,1))\{n|4\}$ vagy másképp $P(4\text{-}1)$ helyébe beírhatjuk a $P(3)$-at.

3. Válaszadás rezolúcióval

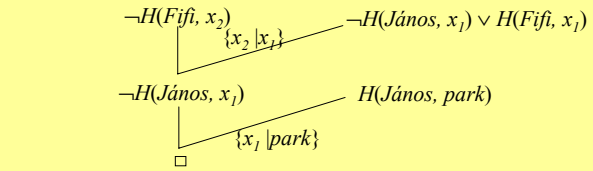
"Ha Fifi mindenhová követi Jánost, és
János most a parkban tartózkodik,
akkor hol van Fifi?"

$H(y,x) \sim y$ dolog az x helyen van.

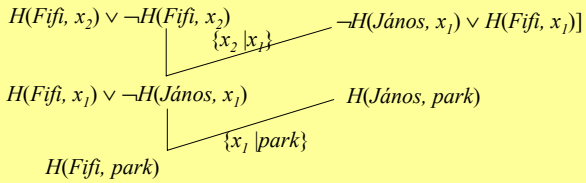
$\forall x[H(János,x) \rightarrow H(Fifi,x)]$
$H(János, park)$

$\Rightarrow \quad \exists x H(Fifi,x)$

Cáfolati gráf



Válaszadási gráf



Megjegyzés

- Rezolúció:
  - $\neg H(János, x_1) \vee H(Fifi, x_1)$
  - $H(János, park)$
  - $\neg H(Fifi,x)$
  - $\Rightarrow \quad \square$
- Válaszadás:
  - $\neg H(János, x_1) \vee H(Fifi, x_1)$
  - $H(János, park)$
  - $H(Fifi, x_2) \vee \neg H(Fifi, x_2)$ (érvényes!)
  - $\Rightarrow H(Fifi, park)$

Válaszadási eljárás

- A kérdést (ki, mit, hol, mikor, mennyiért) egy $\exists x P(x)$ alakú célállítással helyettesítjük.
- Rezolúcióval belátjuk, hogy a célállítás következik az axiómákból.
- A célállítás negáltjából származó klózokat negáltjaik hozzáfűzésével érvényes formulákká egészítjük ki.
- A cáfolati gráf által meghatározott rezolúciót követve létrehozzuk a hasonló szerkezetű válaszadási gráfot, amelynek gyökere tartalmazza az egyik választ.

Példa

- Hanoi tornyai probléma
- A logikai reprezentációra két lehetőség is kínálkozik:
  - az állapottér-reprezentációra épülő (lásd későbbi tárgyaknál: robotika)
  - a dekompozíciós reprezentációra épülő

Formalizáció

- $t(i,j)$ ~ függvényszimbólum egy korong i-dik rúdról j-dik rúdra való átvitelére
- $t(3,2).t(3,1).nil$ ~ egy mozgatás-sorozat, ahol a „.” egy két-argumentumú függvény infix formában, a *nil* pedig az üres sorozat
- $fűz(x, y, z)$ ~ *az x , y és z* sorozatokat összefűző függvény
- $H(n,i,j,k,x)$ ~ predikátum: az x mozgatás-sorozat az i-ről j-re n korongot visz át

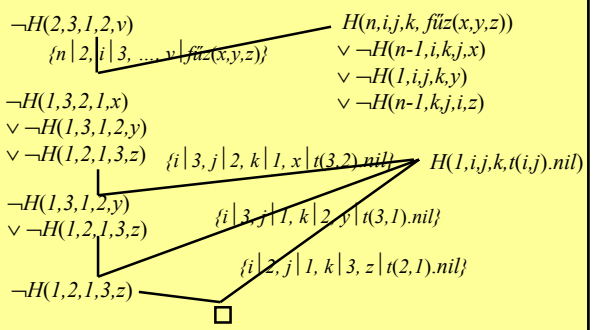
Feladat

- Tényállítás
  - $\forall i \forall j \forall k\ H(1,i,j,k,t(i,j).nil)$
- Szabályállítás
  - $\forall n \forall i \forall j \forall k \forall y \forall z\ [H(n\text{-}1,i,k,j,x) \wedge H(1,i,j,k,y) \wedge H(n\text{-}1,k,j,i,z) \rightarrow H(n,i,j,k,fűz(x,y,z))]$
- Célállítás
  - $\exists v\ H(2,3,1,2,v)$

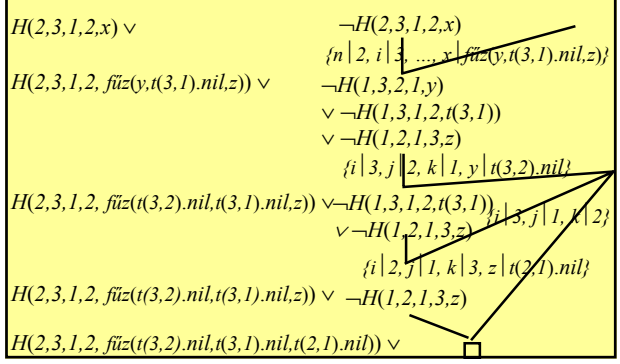
Klózforma

- $H(1,i,j,k,t(i,j).nil)$
- $\neg H(n\text{-}1,i,k,j,x) \vee \neg H(1,i,j,k,y) \vee \neg H(n\text{-}1,k,j,i,z) \vee H(n,i,j,k,fűz(x,y,z))$
- $\neg\ H(2,3,1,2,v)$

Cáfolati gráf



Válaszadási gráf



VIII. Szabályalapú következtetés

1. Reprezentáció
2. Gráfreprezentáció
3. Következtetés
4. A módszer nem teljes

Rezolúció kritikája

- Az $A_1, \ldots, A_n \Rightarrow C$ tételek bizonyítására a rezolúció nem jó MI módszer:
  - A másodlagos vezérlési stratégiák nem eléggé hatékonyak.
  - Nem építhető heurisztika a rezolúció vezérlési stratégiájába, hiszen az állítások a klóz-formára hozás után már nem emlékeztetnek a feladatban betöltött szerepükre.

„Nemcsak maga a gondolat, hanem annak kifejezésmódja is információt hordoz.”

Ha x és y egyaránt zérus, akkor a szorzatuk is az.	Ha x zérus de $x*y$ nem, akkor y sem zérus.
$x=0 \wedge y=0 \rightarrow x*y=0$	$x=0 \wedge x*y\neq 0 \rightarrow y\neq 0$
$Z(x) \wedge Z(y) \rightarrow Z(m(x,y))$	$Z(x) \wedge \neg Z(m(x,y)) \rightarrow \neg Z(y)$
$A \wedge B \rightarrow C$	$A \wedge \neg C \rightarrow \neg B$
$\neg A \vee \neg B \vee C$	$\neg A \vee C \vee \neg B$

A kifejezésmód által hordozott információ heurisztikaként épülhet be a következtetésbe

Ha például be kell látnunk azt, hogy

$$A, A\rightarrow B, C\rightarrow\neg A, \neg B\rightarrow D \Rightarrow B$$

akkor a formulák alakja segítheti a bizonyítást:

$$A, A\rightarrow B \Rightarrow B$$

A rezolúció nem képes felhasználni ezt a segítséget:

$$A, \neg C\vee\neg A, \neg A\vee B, B\vee D, \neg B$$

Olyan következtetési eljárás kell, ahol az állítások megőrzik eredeti alakjukat

axiómák → tények — Konkrét ismeret implikáció nélküli formulákban

szabályok — Általános ismeret implikációs formulákban $A \rightarrow B$

célállítás

A következtetési eljárás iránya

- Előre haladó
  - A tényekből indulva a szabályok segítségével közbenső állításokat vezetünk le, azokból újabb állításokat, amíg a célállítást meg nem kapjuk.
- Visszafelé haladó
  - A szabályokat visszafelé alkalmazva a célállításhoz elégséges részcélokat jelölünk ki, azok előfeltételeül újabb részcélokat, amíg a tények által igazolt részcélokhoz nem jutunk.

A következtetés egy lépése

- Előre láncolás
  - Ha A tény és $A \rightarrow B$ szabály is igaz, akkor B is igaz.
- Visszafelé láncolás
  - Ha $A \rightarrow B$ szabály igaz, akkor B-hez elég belátni A-t.
- Lehet-e a láncolást alkalmazni, ha az $A \rightarrow B$ szabályhoz egy A' tény (vagy B' cél) van megadva?
  - Igen, ha illeszkedik-e az A' az A-hoz?
  - Mire következtethetünk ekkor?

Illesztés logikai háttere

Előre láncolás L bizonyított	Visszafelé láncolás L példányát kell bizonyítani
Szabályillesztés $L, L' \rightarrow W, L\delta = L'\delta \Rightarrow W\delta$	Szabályillesztés $W\delta, W \rightarrow L', L\delta = L'\delta \Rightarrow L\delta$
Célillesztés $L, L\delta = L'\delta \Rightarrow L'\delta$	Tényillesztés $L'\delta, L\delta = L'\delta \Rightarrow L\delta$

Mikor illeszthető X és Y ?

1. ha X és Y egyesíthető (azaz $X\delta = Y\delta$, ahol a δ helyettesítést az egyesítő algoritmus adja)
2. ha $X\delta$ és $Y\delta$ logikailag ekvivalens (ehhez tételbizonyító kell
3. Az első két eset egybeesik, ha az X és Y garantáltan literálok).

Ügyes reprezentációval elérhető, hogy az illesztéseknél csak literálok jelenjenek meg.

1. Reprezentáció

- Az illesztések egyszerű vizsgálata érdekében speciális alakú axiómákat és célállítást használunk, miközben törekszünk az állítások eredeti alakjának megőrzésére.
- ÉS/VAGY formájú (ÉVF) kifejezések a
  - literálok;
  - $A \wedge B$, $A \vee B$ alakú formulák, ahol az A és B maguk is ÉVF kifejezések, vagy ÉVF kifejezések zárójelezett alakjai.

Előre haladó szabályalapú reprezentáció

- Tény:
  - univerzálisan kvantált ÉVF kifejezés.
- Szabályok:
  - $L \rightarrow W$ alakú univerzálisan kvantált kifejezések, ahol L egy literál, a W pedig ÉVF kifejezés.
- Cél:
  - $L_1 \vee \ldots \vee L_n$ alakú egzisztenciálisan kvantált kifejezés, ahol $L_1, \ldots, L_n$ literálok.

Visszafelé haladó szabályalapú reprezentáció

- Tény:
  - $L_1 \wedge \ldots \wedge L_n$ alakú univerzálisan kvantált kifejezés, ahol $L_1, \ldots, L_n$ literálok.
- Szabályok:
  - $W \rightarrow L$ alakú univerzálisan kvantált kifejezések, ahol L egy literál, a W pedig ÉVF kifejezés.
- Cél:
  - egzisztenciálisan kvantált ÉVF kifejezés.

A formulák átalakítása a Skolem normál formára hozáshoz hasonlóan megy

- Szabályokban visszaállítjuk a főimplikációt.
  - (minden más implikációt eliminálunk)
- Célállítást fordítva Skolemizáljuk.
  - Az univerzális kvantoroktól szabadulunk meg.
- Nem alkalmazzuk a disztributív törvényeket.
  - Emiatt nem tudjuk a lehető legáltalánosabb változó-átnevezést alkalmazni, csak főkonjunkciós tényezőnként.

Akadályok

<u>Előre haladó</u>

- Több tényből könnyű egyet csinálni.
- Nem megfelelő alakú cél esetén.

  $(L_1 \vee \ldots \vee L_i) \wedge \ldots \wedge (L_o \vee \ldots \vee L_z)$
- Nem megfelelő alakú szabály:
  - $L_1 \vee L_2 \rightarrow W$ helyett $L_1 \rightarrow W$ és $L_2 \rightarrow W$
  - $L_1 \wedge L_2 \rightarrow W$?

<u>Visszafelé haladó</u>

- Több tényből könnyű egyet csinálni.
- Nem megfelelő alakú tény esetén.

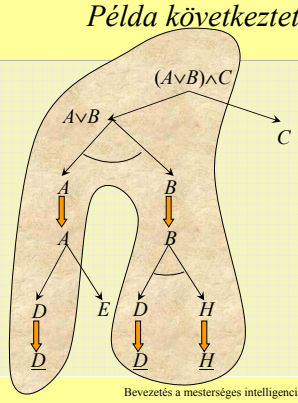
  $(L_1 \wedge \ldots \wedge L_i) \vee \ldots \vee (L_o \wedge \ldots \wedge L_z)$
- Nem megfelelő alakú szabály:
  - $W \rightarrow L_1 \wedge L_2$ helyett $W \rightarrow L_1$ és $W \rightarrow L_2$
  - $W \rightarrow L_1 \vee L_2$?

2. Gráfreprezentáció

- A megcélzott logikai következtetést egy alkalmas gráfbeli út megkeresésének problémájává fogalmazzuk át.
- A logikai reprezentációnak egy ÉS/VAGY gráfot feleltetünk meg, amelynek bizonyos hiperútjai egy-egy bizonyítást szimbolizálnak.

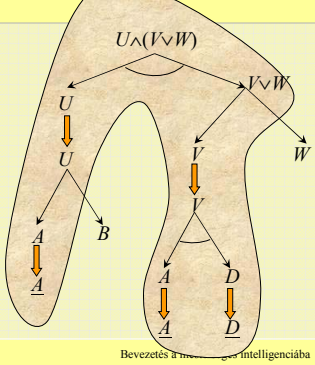
Példa következtetésre

Tény:
$(A \vee B) \wedge C$
Szabályok:
$A \rightarrow D \wedge E$
$B \rightarrow D \vee H$
Cél:
$D \vee G \vee H$



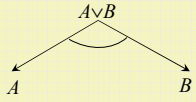
Példa következtetésre

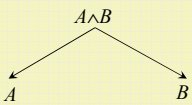
Tény:
$A \wedge C \wedge D$
Szabályok:
$A \vee B \rightarrow U$
$A \wedge D \rightarrow V$
Cél:
$U \wedge (V \vee W)$



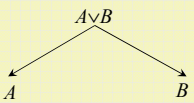
ÉVF kifejezés ÉS/VAGY gráfja

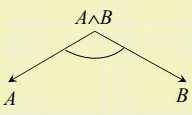
Előre haladó





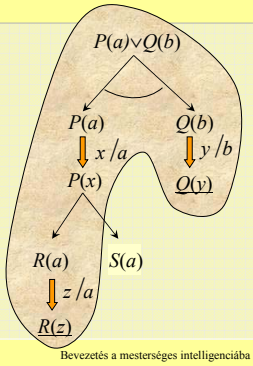
Visszafelé haladó





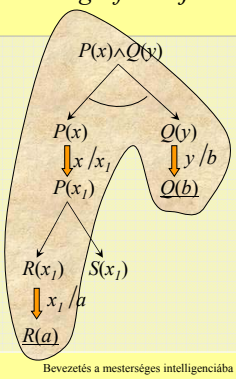
Illesztés ÉS/VAGY gráfba rajzolása

Tény:
$P(a) \vee Q(b)$
Szabály:
$P(x) \rightarrow R(x) \wedge S(x)$
Cél:
$R(z) \vee Q(y)$



Illesztés ÉS/VAGY gráfba rajzolása

Tény:
$Q(b) \wedge R(a)$
Szabály:
$R(x) \vee S(x) \rightarrow P(x)$
Cél:
$P(x) \wedge Q(y)$



Előre haladó szabályalapú gráfreprezentáció

- olyan (R,s,T), ahol
  - $R=(N,A)$ egy ÉS/VAGY gráf, amelyet a tény ÉS/VAGY gráfjából kiindulva építhetünk fel úgy, hogy az összes lehetséges módon illesztjük a szabályokat és a célliterálokat,
  - s - tényállítást szimbolizáló csúcs,
  - T - célliterálokat szimbolizáló csúcsok.

Visszafelé haladó szabályalapú gráfreprezentáció

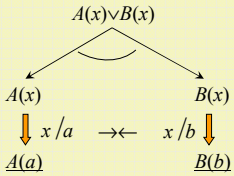
- olyan (R,s,T), ahol
  - $R=(N,A)$ egy ÉS/VAGY gráf, amelyet a cél ÉS/VAGY gráfjából kiindulva építhetünk fel úgy, hogy az összes lehetséges módon illesztjük a szabályokat és a tényliterálokat,
  - s - célállítást szimbolizáló csúcs,
  - T - tényliterálokat szimbolizáló csúcsok.

Megjegyzés

- A reprezentációs gráf mindig egy fa.
- Egy szabály illetve (cél/tény) literál többször is illeszthető.
- A logikai levezetés (bizonyítás) egy megoldásgráfként ($s \rightarrow T$ hiperút) jelenik meg a gráfban.
- Nem minden megoldásgráf jelent levezetést.

Ellentmondásos levezetés

Tény:
$A(x) \vee B(x)$
Cél:
$A(a) \vee B(b)$



Konzisztencia és az egyesítő kompozíció

- Az $\alpha_1, \ldots, \alpha_m$ helyettesítések ($\alpha_i = \{v_{i1} \mid t_{i1}, \ldots, v_{in_i} \mid t_{in_i}\}$) konzisztensek (ellentmondásmentesek),
- ha a
  $V = < v_{11}, \ldots, v_{mn_m} >$ és
  $T = < t_{11}, \ldots, t_{mn_m} >$
  sorozatok egyesíthetők.
- A V és T legáltalánosabb egyesítőjét az $\alpha_1, \ldots, \alpha_m$ helyettesítések egyesítő kompozíciójának (EK) nevezzük.

Megjegyzés

- Ha $\alpha_1, \ldots, \alpha_m$ helyettesítések konzisztensek, akkor nem írnak elő ugyanarra a változóra olyan $\{x \mid t_i\}$ helyettesítéseket, amelyek termjei nem egyesíthetőek:
  - $\alpha = \{x \mid a\}$ és $\beta = \{y \mid a\}$ konzisztens
  - $\alpha = \{x \mid y\}$ és $\beta = \{x \mid a, y \mid b\}$ inkonzisztens
- Az egyesítő helyettesítés független az $\alpha_1, \ldots, \alpha_m$ helyettesítések sorrendjétől.
- A helyes levezetés egy ellentmondásmentes illesztő-helyettesítéseket tartalmazó megoldásgráf.
- Az EK-t válaszadásra is felhasználhatjuk.

Példa

János egy hallgató, vagy Pál nőtlen.

$hallg(János) \vee nőtlen(Pál)$

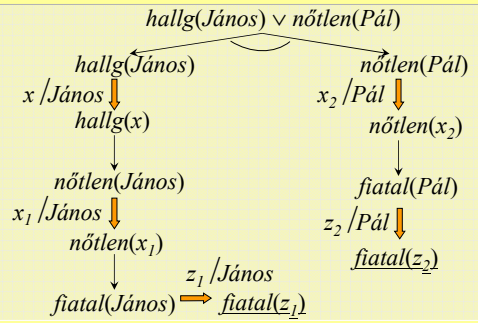
A hallgatók nőtlenek, a nőtlenek pedig fiatalok.

$\forall x\,(hallg(x) \rightarrow nőtlen(x))$
$\forall x\,(nőtlen(x) \rightarrow fiatal(x))$

Nevezzünk meg fiatal embereket!

$\exists z\, fiatal(z)$

Példa folytatása



Példa folytatása

$V = < x, \quad x_1, \quad z_1, \quad x_2, \quad z_2 >$

$T = <$*János, János, János, Pál, Pál*$>$

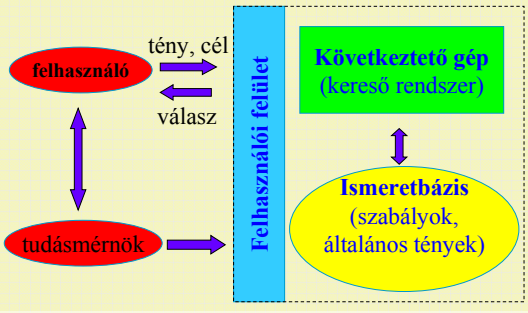
$EK = \{x \,/$*János*$, x_1 \,/$*János*$, z_1 \,/$*János*$, x_2 \,/$*Pál*$, z_2 \,/$*Pál*$\}$

Válasz
$\exists z\, fiatal(z) = \exists z_1\, fiatal(z_1) \vee \exists z_2\, fiatal(z_2)$
$(fiatal(z_1) \vee fiatal(z_2))EK = fiatal(János) \vee fiatal(Pál)$

3. Következtetés

- A reprezentációs gráfban történő irányított út (megoldásgráf) keresése visszalépéses kereséssel.
- A talált megoldásgráf konzisztenciájának ellenőrzése.

Szabályalapú rendszer



Kereső rendszer

- Vezérlési stratégia (elsődleges)
  - visszalépéses stratégia
- Globális munkaterület:
  - reprezentációs gráf startcsúcsból induló hiperútja
- Kereső rendszer szabályai:
  - illesztések, illetve visszalépés

Vezérlési stratégia finomítása

- Másodlagos stratégiák
  - Formulák alakjának megőrzése, az alak felhasználása
  - axiómak felosztásának (szabályok, tények) kihasználása
- Heurisztikák
  - az adott feladat speciális ismeretei

Példa

Tomi és Misi tagjai az alpinisták klubjának.
Egy klubtag síelő vagy hegymászó.
Nincs olyan hegymászó, aki szeretné az esőt.
A havat minden síelő szereti.
Tomi szereti az esőt és a havat.
Misi mindenben ellentétesen vélekedik, mint Tomi.

Ki az a klubtag, aki hegymászó, de nem síelő?

Tény vagy szabály?

- Egyértelmű esetek
  - $A(Tomi) \wedge A(Misi) \wedge Sz(Tomi,eső) \wedge Sz(Tomi,hó)$
  - $\forall x\,(A(x) \rightarrow S(x) \vee H(x))$
  - $\forall x\,(S(x) \rightarrow Sz(x,hó))$
- Nem egyértelmű esetek
  - $\neg\exists x\,(H(x) \wedge Sz(x,eső))$ helyett $\forall x\,(H(x) \rightarrow \neg Sz(x,eső))$
  - $\forall y\,(Sz(Tomi,y) \rightarrow \neg Sz(Misi,y))$
  - $\forall y\,(\neg Sz(Tomi,y) \rightarrow Sz(Misi,y))$

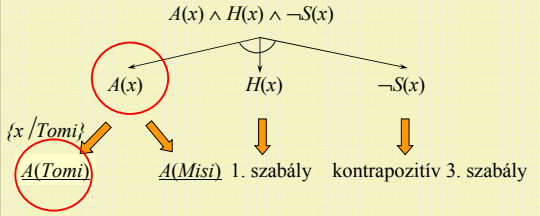
Szabályok formája

- szabály szétbontása
  - $\forall x\,(A(x) \rightarrow S(x) \vee H(x))$ helyett
    $\forall x\,(A(x) \wedge \neg S(x) \rightarrow H(x))$
    $\forall x\,(A(x) \wedge \neg H(x) \rightarrow S(x))$
- szabály átformálása (kontrapozitív alak)
  - $\forall y\,(\neg Sz(Tomi,y) \rightarrow Sz(Misi,y))$ mellett
    $\forall y\,(\neg Sz(Misi,y) \rightarrow Sz(Tomi,y))$

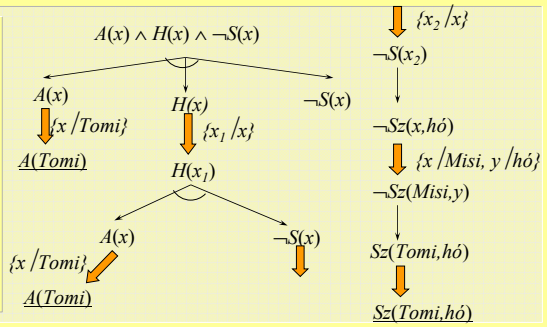
Következtetési irány megválasztása

- Tény:
  - $A(Tomi) \wedge A(Misi) \wedge Sz(Tomi,eső) \wedge Sz(Tomi,hó)$
- Szabály:
  - $A(x) \wedge \neg S(x) \rightarrow H(x)$
  - $A(x) \wedge \neg H(x) \rightarrow S(x)$
  - $S(x) \rightarrow Sz(x,hó)$
  - $H(x) \rightarrow \neg Sz(x,eső)$
  - $Sz(Tomi,y) \rightarrow \neg Sz(Misi,y)$
  - $\neg Sz(Tomi,y) \rightarrow Sz(Misi,y)$
- Cél: $\exists x\,(A(x) \wedge H(x) \wedge \neg S(x))$

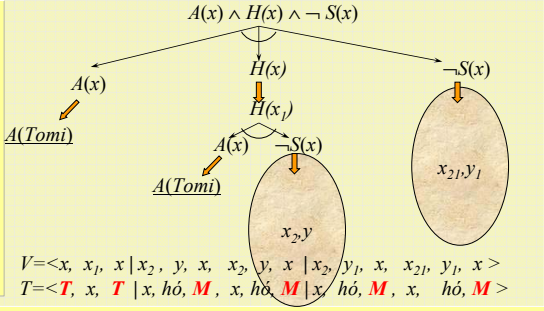
Tény illesztése a szabály illesztése előtt



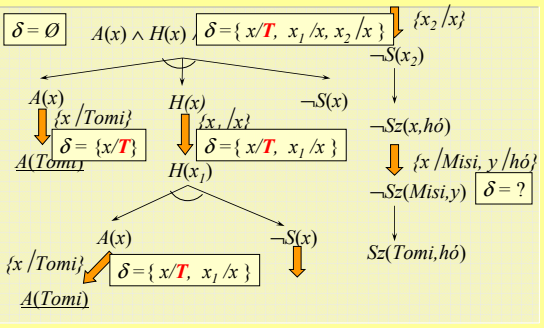
Példa folytatása



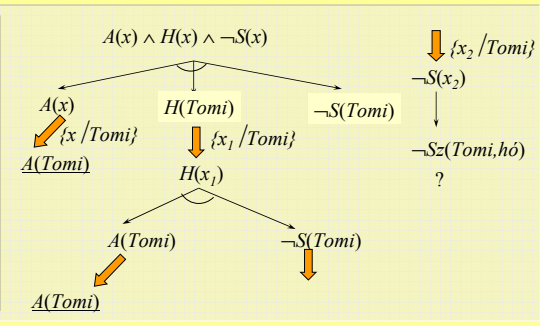
Szabálykapcsolat-gráf



Ellentmondás korai felismerése 1.



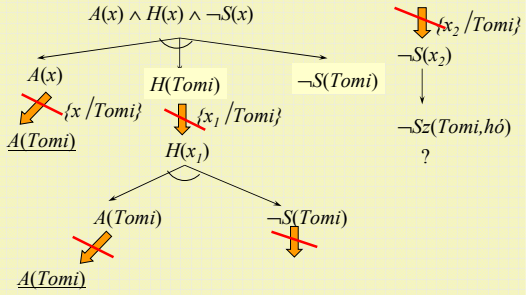
Ellentmondás korai felismerése 2.



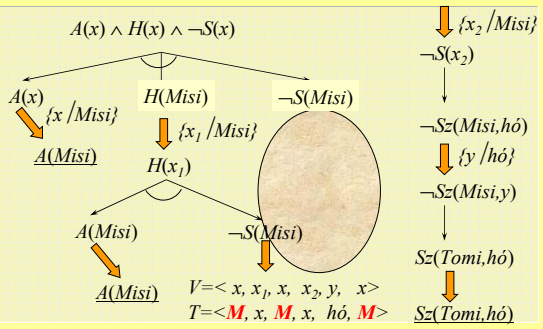
Folyamatos konzisztencia ellenőrzés

- Additív módon számolt aktuális egyesítő kompozíció.
  - Az aktuális hiperút konzisztenciáját vizsgáljuk.
    - Ha ellentmondásos, akkor visszalépünk.
    - Ha nem (van EK), tovább lépünk; ezután elég a soron következő illesztő helyettesítés és a korábban talált EK konzisztenciáját megvizsgálni. (additív számolás)
  - Aktuális EK tárolása minden csúcsnál.
- Változó behelyettesítés a hiperút párhuzamos ágain.
  - Egy változóra vonatkozó helyettesítést, nemcsak az illesztés alatt, hanem az aktuális hiperútban mindenhol érvényesítjük.
  - Változó helyettesítés előtti állapot hisztorizálása.

Példa folytatása: visszalépések



Példa folytatása a sorozatos visszalépések után



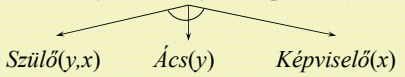
Sorrendi heurisztikák

- Kiértékelő függvény
  - az adott pillanatban illeszthető literálok (tények illetve szabályok) rangsorolására.
- Meta-szabály
  - az adott pillanatban illeszthető literálok (tények illetve szabályok) rangsorolására.

1. Példa

Azt a literált válasszuk, amelyikhez a legkevesebb féle illesztést alkalmazhatjuk!

Tény:
képviselők, ácsok, szülő-gyerek párok
Cél:
Szülő(y,x) ∧ Ács(y) ∧ Képviselő(x)



1. Példa

Szülő(u,v)	Szülő(a,v)	Szülő(u,a)	Szülő(a,b)
1 000 000	12	2	1

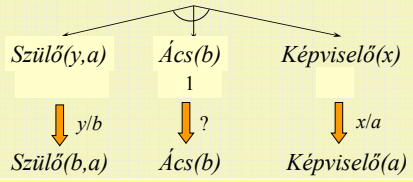
Ács(u)	Ács(a)
10 000	1

Képviselő(u)	Képviselő(a)
350	1

1. Példa

Lehetséges próbálkozások száma a
Szülő, Ács, Képviselő
rögzített illesztési sorrendje mellett:

Szülő, Ács, Képviselő:	1 000 000*1*1
Ács, Képviselő, Szülő:	10 000*350*1
Ács, Szülő, Képviselő:	10 000*12*1
Képviselő, Ács, Szülő:	350*10 000*1
Képviselő, Szülő, Ács:	350*2*1



2. Példa

Mindig a fontosabb szabályt illesszük!

Ha az alábbi szabályok közül mindkettő illeszthető

1. „Ha a betegnek leállt a szíve, akkor szívmasszázst kell alkalmazni.”
2. „Ha a betegnek horzsolt seb van az alkarján, akkor be kell kötözni.”

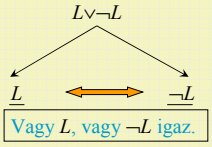
akkor az elsőt kell alkalmazni.

4. A szabályalapú következtetés nem teljes módszer

- A reprezentáció korlátjai miatt
  - Nem minden logikai reprezentáció írható át csak előre vagy csak visszafelé haladó szabály alakúra.
    (Milyen irányú az $L_1 \wedge L_2 \rightarrow L_3 \vee L_4$ szabály?)
- A következtetés korlátjai miatt
  - Nem mindig ad a szabályalapú következtetés megoldásgráfot, amikor a cél következménye az axiómáknak.
- A továbbiakban visszafelé haladó következtetéssel foglalkozunk, de a tanulságok előre haladó következtetésre is alkalmazhatóak.

Korlátozott célrezolúció: 1. Példa

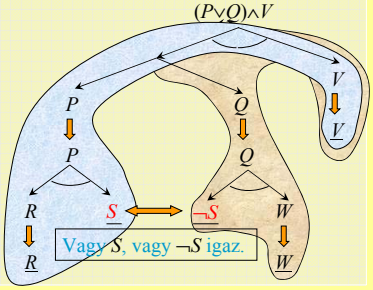
Tény: -
Szabályok: -
Cél: $L \vee \neg L$



KCR kapcsolat

Korlátozott célrezolúció: 2. Példa

Tény:
$R \wedge W \wedge V$
Szabályok:
$R \wedge S \rightarrow P$
$\neg S \wedge W \rightarrow Q$
Cél:
$(P \vee Q) \wedge V$

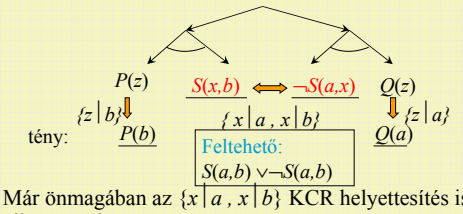


Korlátozott célrezolúció fogalma

- Ha a startcsúcsból kiinduló két különböző hiperúton (klózban) találunk olyan komplemens literálpárt, amelyek a változóik átnevezése után egyesíthetők, akkor azokat célcsúcsoknak tekintjük.
- A konzisztens levezetést az ellentmondásmentes KCR-rel összekapcsolt megoldásgráfok jelzik.

Látszólagos inkonzisztencia

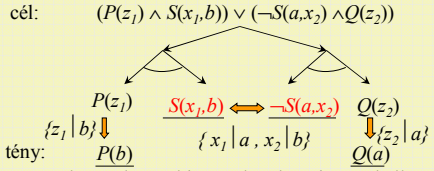
cél: $(P(z) \wedge S(x,b)) \vee (\neg S(a,x) \wedge Q(z))$



- Már önmagában az $\{x \mid a\ ,\ x \mid b\}$ KCR helyettesítés is ellentmondásos.
- A KCR-rel összekötött hiperutakban is lehetnek ellentmondó helyettesítések: $z \mid a\ ,\ z \mid b$

Látszólagos inkonzisztencia feloldása

cél: $(P(z_1) \wedge S(x_1,b)) \vee (\neg S(a,x_2) \wedge Q(z_2))$

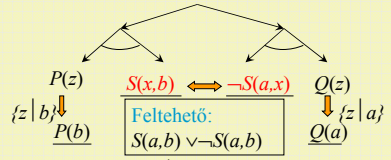


- A KCR-rel összekötött hiperutak változóit meg kell és meg lehet egymástól különböztetni.
  - Ugyanis a célcsúcsból kivezető hiperutak diszjunkcióban álló formulákat szimbolizálnak, amelynek változói egzisztenciálisan kvantáltak. Az egzisztenciális kvantor a diszjunkcióba bevihető (illetve kiemelhető) és a független kvantorokban változóátnevezést lehet alkalmazni.

Konzisztencia ellenőrzés

- Külön-külön kell konzisztencia ellenőrzést végezni az egyes hiperutakra és a KCR hozzájuk tartozó helyettesítéseire.

cél: $(P(z) \wedge S(x,b)) \vee (\neg S(a,x) \wedge Q(z))$



Ha $S(a,b)$ akkor a KCR-ből $\{x \mid a\}$-ra és a baloldali hiperútra.
Ha $\neg S(a,b)$ akkor a KCR-ből $\{x \mid b\}$-re és a jobboldali hiperútra.

IX. Alternatív következtetés

Klasszikus logika problémái

Klasszikus logika	Emberi gondolkodás
Teljes axióma rsz.	Hiányos axióma rsz.
Ellentmondásmentes	Ellentmondások
Monotonitás	Nem monoton
Egyértelmű állítások	Bizonytalan állítások
Függetlenség	Interferencia
Deklaratív	Procedurális is
Predikátum alapú	Objektum alapú is

1. Nemmonoton következtetések

- Hiányos és ellentmondásos axiómarendszerben is működő következtetési technikák.
- Ellentétben a klasszikus logikával, ahol az axiómarendszer kiegészítésekor a korábban levezetett állítások érvényben maradnak, itt egy korábban levezetett állítást nem mindig lehet a bővített axiómákból levezetni, azt törölni kell. A levezetett állítások köre nem bővül monoton módon.

1.1. Körülhatároló technikák

- „Ami nem bizonyítható, az nem igaz”
  - Az adott axiómarendszert kielégítő minimális modellt kell megkeresni úgy, hogy újabb axiómákat veszünk fel.
- Általában nem eldönthető az, hogy egy állítás nem bizonyítható.
  - Speciális állítások esetén viszont igen.

Zártvilág feltételezés

- Ha egy alapatom nem logikai következménye egy Δ axiómarendszernek, akkor hozzávesszük az axiómákhoz annak negáltját.
  („Ami nincs az nem igaz.”)
- $CWA(\Delta) = \{\varphi \mid \Delta \cup \Delta_{asm} \Rightarrow \varphi\}$
- ahol $\Delta_{asm} = \{\neg p \mid p \text{ alapatom és } \Delta \not\Rightarrow p\}$
- Példa: menetrend

1.2. Default következtetés

- A kivétel ellentmondáshoz vezet a klasszikus logikában
  - $Madár(x) \rightarrow Repül(x)$
  - $Strucc(x) \rightarrow Madár(x)$
  - $Strucc(x) \rightarrow \neg Repül(x)$
- Egy lehetséges megoldás
  - $Madár(x) \wedge \neg Strucc(x) \rightarrow Repül(x)$
- Hiányos axiómarendszer esetén később új kivételek jelenhetnek meg.

Default logikai szabály

- A $\alpha(\underline{x}):\beta(\underline{x}) \rightarrow \gamma(\underline{x})$ az a következtetési elv, amely szerint,
  - ha a $\beta(\underline{x})\{\underline{x}/t\}$ egy olyan alapformula amely nem mond ellent az eddig levezetett állításainknak (azaz a $\neg\beta\{\underline{x}/t\}$ nem szerepel azok között),
  - akkor $\alpha\{\underline{x}/t\}$ formula fennállása esetén következtethetünk a $\gamma\{\underline{x}/t\}$ formulára.
- Elnevezések
  - α előfeltétel, β igazolás, γ következmény

Példák

Menetrend

: ¬járat(x,y) → ¬járat(x,y)

Háziasszony

háziasszony(x) : szeret(x,virág) → visz(x,virág)

Madár

madár(x) : repül(x) → repül(x)

2. Valószínűségi következtetés

- Az állítások bizonytalanságának okai:
  - Hiányzó adat: A besárgult bőrű beteg hepatitiszes-e?
    - *p(A) = sárga bőrű hepatitiszesek / sárga bőrű betegek*
  - Pontatlan műszerek pontatlan leolvasása: *80 °C ± 2 °C*
    - *p(A) = 1– 4/80 ≈ 0.95*
  - Elmosódott jelentésű állítások: Jóska magas
- Alapkérdés:
  - Ha tudjuk valamilyen bizonyossággal, hogy A és A→B igaz, akkor milyen bizonyossággal következtethetünk B-re?

2.1. Klasszikus valószínűség számítás

- A központi kérdés annak a meghatározása, hogy mi az *A* valószínűsége, ha *B* bekövetkezik.
- Feltételes valószínűség:

$$P(B \mid A) = \frac{p(B \wedge A)}{p(A)} \quad \text{ha } p(A)>0$$

- Hogyan lehetne egy adott probléma együttes eloszlásfüggvényét minél tömörebben, az ismert ok-okozati összefüggések (függetlenségek) és feltételes valószínűségek segítségével ábrázolni .

Emlékeztető: Valószínűségi változók

- Jelölések:
  - $X_i=x_i$ állításokkal dolgozunk, ahol X_i egy diszkrét valószínűségi változó, x_i pedig az értéke.
  - Az $X_i=x_i$ állítást rövidíthetjük az X_i-vel, amennyiben a szövegkörnyezetből egyértelműen kiderül az értéke.
  - Speciális eset az, amikor az X_i lehetséges értékei: *igaz* vagy *hamis*. Ekkor az X_i=*igaz* helyett használhatjuk az X_i-t, az X_i=*hamis* helyett pedig $\neg X_i$–t.

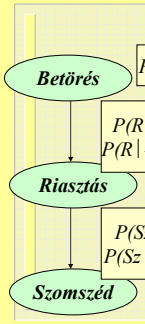
Emlékeztető: Lánc szabály, normalizálás

- Feltételes valószínűség definíciójának többszöri alkalmazása – lánc szabály.
  - $p(X_1,\ldots,X_n) = p(X_1 \mid X_2,\ldots,X_n) * p(X_2 \mid X_3,\ldots,X_n) * \ldots * p(X_{n-1} \mid X_n) * p(X_n)$
- $p(B \mid A) = \frac{S_1}{N}$ értékét αS_1 formában keressük, mert nem ismerjük az *N*-t, de tudjuk, hogy $p(\neg B \mid A) = \frac{S_2}{N}$, és ki tudjuk számolni S_1 és S_2 értékét.
  - ugyanis $1 = p(B \mid A) + p(\neg B \mid A) = \frac{S_1+S_2}{N}$
  - így $N = S_1+S_2$
  - tehát $\alpha = 1/(S_1+S_2)$

Emlékeztető:Feltételes függetlenség

- Közönséges függetlenség
  - $p(A,B) = p(A)\,p(B)$
- Az A és a B feltételesen függetlenek a E fennállása esetén, ha
  - $p(A,B \mid E) = p(A \mid E)\,p(B \mid E)$
- Gyakori alkalmazási formája:
  - $p(A \mid B,E) = p(A \mid E)$
  - Ui: $p(A \mid B,E) = p(A,B,E) / p(B,E) =$
    $= p(A,B \mid E)\,p(E) / p(B \mid E)\,p(E) =$
    $= p(A \mid E)\,p(B \mid E)\,p(E) / p(B \mid E)\,p(E) = p(A \mid E)$

A szomszédunk telefonált, hogy szól a betörés-riasztónk a lakásunkban. Betörtek volna hozzánk? *Russel-Norvig: AI*



$p(B \mid Sz) =$ def
$= p(B,Sz) / p(Sz) = \alpha\, p(B,Sz)$ telj fgl rsz $R,\neg R$
$= \alpha\,[p(Sz,R,B) + p(Sz,\neg R,B)]$
$= \alpha\,[p(Sz \mid R,B)\,p(R \mid B)p(B) +$ lánc
$p(Sz \mid \neg R,B)\,p(\neg R \mid B)]\,p(B)$
$= \alpha\,[p(Sz \mid R)\,p(R \mid B) +$ felt. fgl.
$p(Sz \mid \neg R)\,p(\neg R \mid B)]\,p(B)$
$= \alpha\, 0.0008575$
$p(\neg B \mid Sz) = \alpha\, 0.0507991$
$\alpha = 19.3585$ normalizálás
$p(B \mid Sz) = 0.0166$

Reprezentáció valószínűségi hálóval

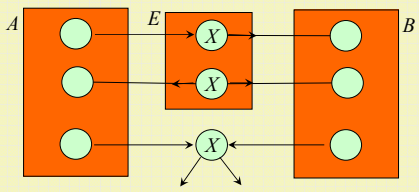
- Tekintsük a tárgyprobléma valószínűségi változóit.
- Feleltessük meg a változókat egy *körmentes irányított gráf* csúcsainak.
- Ábrázoljuk az irányított élekkel a változók közötti közvetlen ok-okozati összefüggéseket (ez által feltárjuk a feltételes függetlenségeket).
- Adjuk meg az csúcsok *feltételes valószínűségi tábláit* (FVT): azokat a $p(X=x \mid szülő(X)=x_1, \ldots, x_k)$, ahol a *szülő(X)* az X csúcsának szülőcsúcsaihoz rendelt $X_1, \ldots, X_k$ változók együttese. Röviden: $p(X \mid szülő(X))$.

Valószínűségi háló kifejező ereje

- A valószínűségi hálóból kiolvasható az adott tárgykör együttes valószínűségi eloszlása (lánc-szabály).
  - $p(X_1, \ldots, X_n) = p(X_n \mid X_1, \ldots, X_{n-1})^*$
    $p(X_{n-1} \mid X_1, \ldots, X_{n-2})^* \ldots {}^* p(X_2 \mid X_1)\, {}^* p(X_1)$
- Sorszámozzuk meg úgy a változókat, hogy ha $i>j$, akkor X_i-ből ne vezessen irányított út X_j-be, azaz minden i-re $szülő(X_i) \subseteq \{X_1, \ldots, X_{i-1}\}$.
- Ekkor a feltételes függetlenség miatt
  - $p(X_i \mid X_1, \ldots, X_{i-1}) = p(X_i \mid szülő(X_i))$,
- így
  - $p(X_1, \ldots, X_n) = \Pi_{i=1\ldots n}\, p(X_i \mid szülő(X_i))$

Feltételes függetlenség felismerése valószínűségi hálókban

Az A és a B feltételesen független a E-re nézve, ha minden A illetve B-hez tartozó csúcs-pár közötti összes útra teljesül az alábbi esetek egyike:



Emlékeztető:Bayes tétel különféle alakjai

a) Klasszikus

$$p(B \mid A) = \frac{p(A \mid B)\,p(B)}{p(A)}$$

b) Háttértudás mellett

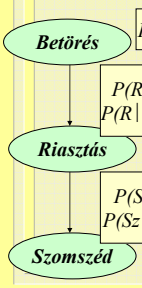
$$p(B \mid A,E) = \frac{p(A \mid B,E)\,p(B \mid E)}{p(A \mid E)}$$

c) Általánosított, ($B_1, \ldots, B_n$ teljes és független)

$$p(B_i \mid A) = \frac{p(A \mid B_i)p(B_i)}{\Sigma_k\, p(A \mid B_k)p(B_k)}$$

A szomszédunk telefonált, hogy szól a betörés-riasztónk a lakásunkban. Betörtek volna hozzánk?

normalizált Bayes tétel



$p(B \mid Sz) = p(Sz \mid B)p(B) / p(Sz)$
$= \alpha\, p(Sz \mid B)p(B)$
$= \alpha\, [p(Sz,R \mid B) + p(Sz,\neg R \mid B)]\, p(B)$
$= \alpha\, [p(Sz \mid R,B)\, p(R \mid B) + p(Sz \mid \neg R,B)\, p(\neg R \mid B)]\, p(B)$
$= \alpha\, [p(Sz \mid R)\, p(R \mid B) + p(Sz \mid \neg R)\, p(\neg R \mid B)]\, p(B)$
$= \alpha\, 0.0008575$
$p(\neg B \mid Sz) = \alpha\, 0.0507991$
$\alpha = 19.3585$
$p(B \mid Sz) = 0.0166$

Valószínűség hálók tervezése

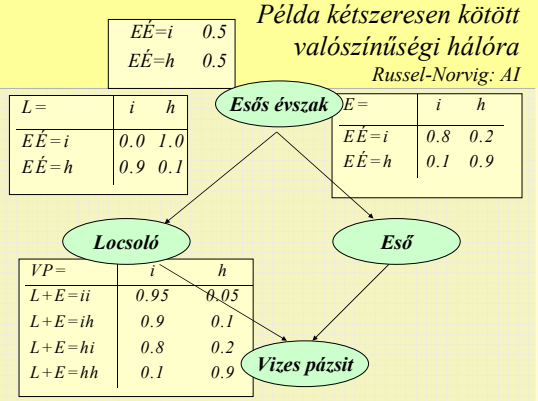
- Határozzuk meg a tárgytartományt leíró változók halmazát, majd meghatározott sorrendben dolgozzuk fel őket:
  - Válasszunk ki olyat, amely kizárólag a hálóhoz csatolt változóktól függ, és új csúcsként vegyük fel a hálóba
  - A hálóbeli változóknak vegyük azt a minimális halmazát, amelyek közvetlenül hatnak az új változóra. Rajzoljuk be ezeket a függőségeket reprezentáló éleket.
  - Töltsük ki az új csúcs FVT-jét.

Következtetés valószínűségi hálókban

- Célja egy $p(X_i \mid X_{j1}, \ldots, X_{jk})$ feltételes valószínűség meghatározása a Bayes módszerre alapuló számítással (Bayes tételek, normalizálás, felbontás teljes fgl. eseményrendszerre, lánc-szabály, feltételes fgl)
- Egy $p(X_i \mid X_{j1}, \ldots, X_{jk})$ egyedileg is számolható, de célszerűbb erre általános algoritmust találni.
- Egy ilyen (rekurzív) algoritmus számításigénye erősen függ a háló bonyolultságától.
- Egyszeresen kötött hálókra (fa-gráfokra), ahol két csúcs között nincsenek alternatív irányítatlan utak, van lineáris futási idejű algoritmus.

Példa kétszeresen kötött valószínűségi hálóra

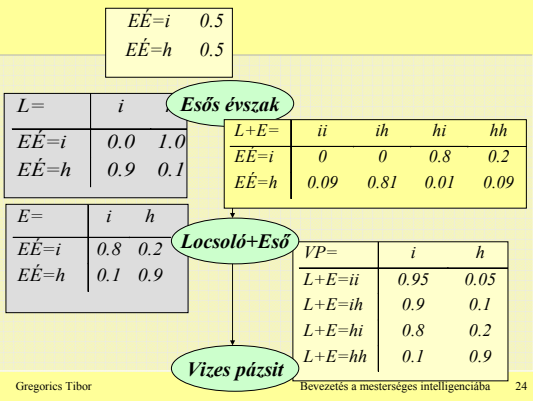
Russel-Norvig: AI



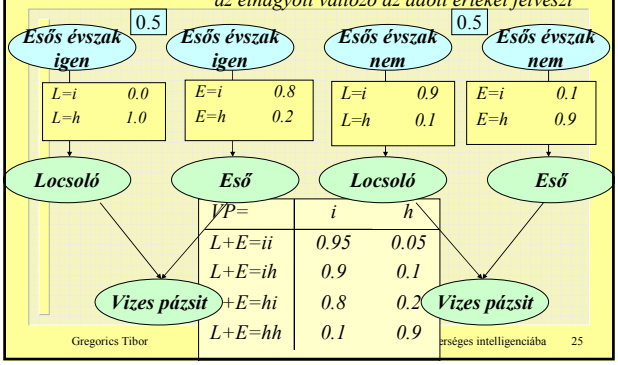
Következtetés többszörösen kötött hálókban

- Összevonásos eljárások
  - változók összevonásával fa-gráfot készítünk.
- Vágóhalmaz feltételezésen alapuló eljárások
  - a hálóból több fa-gráfot készítünk és a válasz az azokból kiszámított eredmények súlyozott átlaga lesz.
- Sztochasztikus szimulációs eljárások
  - Számítási modellnek tekintjük a hálót, és az abba írt valószínűségi értékeket figyelembe véve példákat generálunk. A feltett kérdésre a választ a jó példák relatív gyakorisága alapján számoljuk ki.

a)Változók összevonásával fa-gráfot készítünk



b) Vágóhalmaz feltételezés : A hálóból több fa-gráfot készítünk változók elhagyásával. Egy elhagyott változó minden értékéhez tartozik egy-egy háló, aminek súlya annak a valószínűsége, hogy az elhagyott változó az adott értéket felveszi



VP=	i	h
L+E=ii	0.95	0.05
L+E=ih	0.9	0.1
L+E=hi	0.8	0.2
L+E=hh	0.1	0.9

c) Sztochasztikus szimuláció: a háló alapkán példákat generálunk, és azok relatív gyakoriságot számoljuk

$$P(A \mid B) = \frac{P(A,B)}{P(B)} = \frac{\text{Jó hasznos példák száma}}{\text{Összes hasznos példa száma}}$$

Hasznos példa előállítása a $P(Vizes\ pázsit \mid Eső)$ számára 0.2 súllyal

- $EÉ=Random(0.5)$ mert $P(EÉ)=0.5$
  - TF: $EÉ=hamis$.
- $L=Random(0.9)$ mert $P(L \mid \neg EÉ)=0.9$
  - TF: $L=igaz$.
- E tényváltozó, értéke biztos igaz, és $P(E \mid \neg EÉ))= 0.2$
  - Ezért $E=igaz\ (0.2)$
- $VP= Random(0.95)$ mert $P(VP \mid E,L)=0.95$
  - TF: $VP=igaz$.

Valószínűségi hálók értékelése

- Kevesebb a priori valószínűséget kell benne tárolni, mintha az együttes valószínűségi eloszlásfüggvényt akarnánk ábrázolni.
- Egyszerűen bővíthető anélkül, hogy eddigi valószínűségeket újra kellene gondolni.
- A következtetés felhasználható magyarázatadásra.
- Matematikailag jól megalapozott, de - az erőfeszítéseink ellenére - igen számításigényes.

2.2. Más bizonytalanság kezelő technikák

- Betörés-riasztó-szomszéd probléma szabályalapú megközelítése:
  - tény: Sz szabályok: $Sz \rightarrow R$,$R \rightarrow B$
  - Ekkor Sz , $Sz \rightarrow R \Rightarrow R$; R , $R \rightarrow B \Rightarrow B$
- Rendeljünk valószínűségeket a szabályokhoz:
  - $Sz \rightarrow R$ (0.95) hiszen $p(Sz \mid \neg R) = 0.05$
  - $R \rightarrow B$ (0.999) hiszen $p(R \mid \neg B) = 0.001$
- Ha T (p) és $T \rightarrow K$ (q) akkor K $(p*q)$
- Sz $(1) \Rightarrow R$ $(0.95) \Rightarrow B(0.95*0.999)$

Alternatív bizonytalanság kezelés

- MYCIN
  - Szabályalapú következtetéshez illesztett bizonytalanság kezelés
- Dempster-Shaffer
  - Az állítások valószínűségének van egy bizonytalansága, amit a tények ismeretének hiánya okoz. (Mi a valószínűsége annak, hogy egy 90%-os biztonsággal szabályosnak tartott érmével „fejet” dobunk?)
- Fuzzy
  - Elmosódott jelentésű állításokkal végzett következtetés. (Tudjuk, hogy HA tananyag nehézsége=közepes ÉS jegyzeteltsége=rossz AKKOR tanulási_idő=hosszú. Mennyi ideig tanuljuk a *45%*-ban nehéz, *31%*-osan jegyzetelt tananyagot?

3. Procedurális tudásábrázolás

- Végrehajtható tudás
- Deklaratív és procedurális szabályok
  - Ha <feltétel> akkor <következmény>
  - Ha <feltétel> akkor <tevékenység>

Deklaratív reprezentáció átírása procedurális formába

Tény:
Személy(Socrates)
Személy(Helena)
Szabályok:
Személy(x) → *Halandó(x)*
Kutya(x) → *Halandó(x)*
Cél:
Halandó(Socrates)
Halandó(x)

```
Procedure Személy(x) return logical;
   if x=Socrates or   x=Helena
      then return True
      else return False;
end;
```

```
Procedure Halandó(x) return logical;
   return (Személy(x) or Kutya(x));
end;
```


A procedurális tudásábrázolás tömörebb, de kevésbé általános

Tény:
Succ('A','B')
Succ('B','C')
...
Cél:
Succ('K',x)
Succ(y,'K')

```
Procedure Succ(y,x) return logical
   if y='Z' then return False
   else
        x:=chr(ord(y)+1);
        return True;
   endif;
end;
```

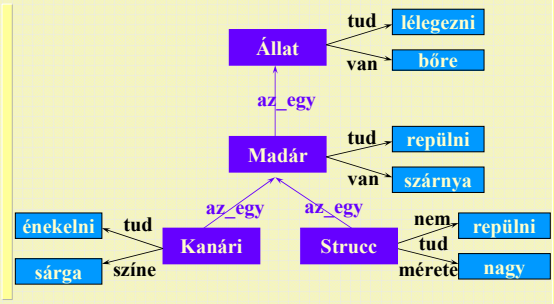

Procedurális hozzárendelés

- Egy adott modellben (interpretációban) egy függvényszimbólum alappéldánya kiértékelhető, és helyettesíthető az eredményt szimbolizáló konstansszimbólummal.
- Pl: Ha a $P(f(n,1))$ formula konkrét modellje az egész számokról szól, ahol f a kivonás művelete, akkor a $P(f(n,1))\{n|4\}$ vagy másképp $P(4\text{-}1)$ helyébe beírhatjuk a $P(3)$-at.

4. Strukturált objektumalapú reprezentáció

- Collins és Quillian kísérlete (emberek válaszidői):
  - Tud-e a kanári énekelni? *1.3 mp*
  - Képes-e a kanári repülni? *1.4 mp*
  - Van-e a kanárinak bőre? *1.5 mp*
  - A kanári egy kanári? *1.0 mp*
  - A kanári egy madár? *1.2 mp*
  - A kanári egy állat? *1.3 mp*
  - Tud-e a kanári repülni? *1.4 mp*
  - Tud-e a strucc repülni? *1.3 mp*

Az emberi információtárolás hierarchikus és objektumalapú

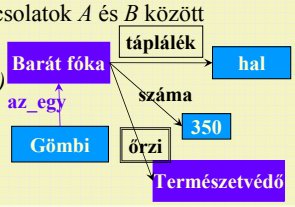


Szemantikus háló

- Egy olyan irányított gráf, amelynek csúcsai és élei címkézettek.
- Csúcs: objektum
  - a világ egy egyede (elem, példány)
  - egyedek csoportja (halmaz, osztály)
- Él: kapcsolat
  - objektum eleme-e ill. része-e egy csoportnak
  - objektum tulajdonsága

Objektumok kapcsolataik

- Egyszerű kapcsolatok *A* és *B* között : *R(A,B)*
  - Standard: részhalmaza illetve eleme (az_egy)
  - Egyedek közötti egyéb kapcsolat
- Magasabb rendű kapcsolatok *A* és *B* között
  - *R* – $\forall x \in A : R(x,B)$
  - *R* – $\forall x \in A\ \exists y \in B : R(x,y)$



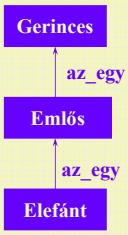
Predikátum alapú reprezentációtól az objektum alapú reprezentációig

- A szemantikus háló képes kifejezni
  - konstans-szimbólumokat → objektum
  - egy arg. predikátumokat → objektum
  - két arg. predikátumokat → kapcsolat
  - több arg. predikátumokat → átalakítás
  - logikai műveleteket

Implikáció, konjunkció, univerzális kvantor a szemantikus hálóban

Minden elefánt emlős és minden emlős gerinces

$\forall x(Elefánt(x) \rightarrow Emlős(x)) \wedge$
$\forall x(Emlős(x) \rightarrow Gerinces(x))$



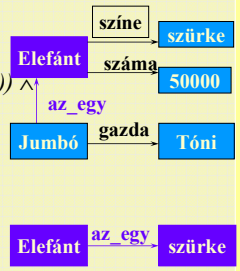
Jumbó egy elefánt.
Minden elefánt színe szürke.
Jumbó gazdája Tóni.

Példa

Elefánt(Jumbó) $\wedge$
$\forall x(Elefánt(x) \rightarrow színe(x,szürke)) \wedge$
gazda(Jumbó,Tóni)

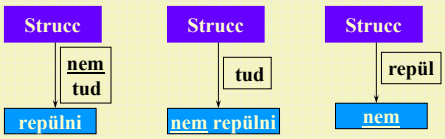
Az elefántok száma 50000.???

$\forall x(Elefánt(x) \rightarrow szürke(x))$



Tagadás lehetséges változatai

$\forall x(Strucc(x) \rightarrow \neg Repül(x))$

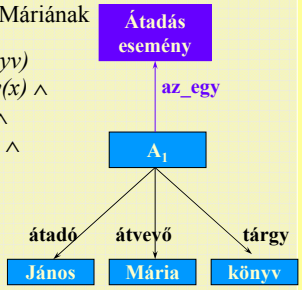


Három vagy több argumentumú predikátumok

János egy könyvet adott Máriának

AD(János, Mária, könyv)
= ∃x(Átadás_esemény(x) ∧
Átadó(x,János) ∧
Átvevő(x,Mária) ∧
Tárgy(x,könyv)

Skolemizálás:
A_1 egy átadás esemény



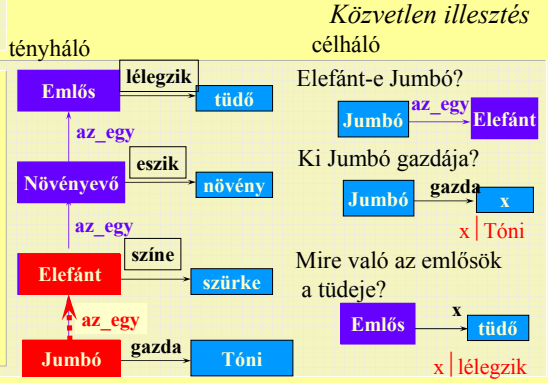
Szemantikus hálóval leírható feladatok

- Az univerzum szerkezetére valamilyen taxonómikus hierarchia jellemző, azaz az objektumok egymásba ágyazott halmazok rendszerében helyezkednek el.
- Az univerzum objektumai között kiterjedt, de logikailag egyszerű kapcsolatrendszer áll fenn.

Következtetés

- axiómák → tényháló
- célállítás → célháló
- Azt, hogy
  - axiómák ⇒ célállítás
- úgy bizonyítjuk, hogy megpróbáljuk a célhálót „ráhelyezni”, „beleilleszteni” a tényhálóba
- Algoritmus kell!

Közvetlen illesztés



Öröklődés

- A logikai következtetés alapvető tulajdonsága.
- Ha
  - $\forall x(Elefánt(x) \rightarrow Emlős(x))$
  - $\forall x(Elefánt(x) \rightarrow Színe(x,szürke))$
  - $Elefánt(Jumbó)$
- akkor
  - $Emlős(Jumbó)$
    hiszen $Elefánt(Jumbó) \rightarrow Emlős(Jumbó)$
  - $Színe(Jumbo,szürke)$
    hiszen $Elefánt(Jumbó) \rightarrow Színe(Jumbó,szürke)$

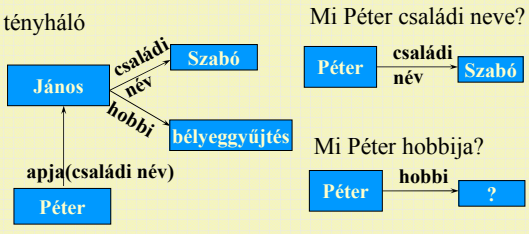
Öröklődés a szemantikus hálókban

- Az az_egy kapcsolat átnyúlik a többi az_egy kapcsolaton

- A magasabb rendű kapcsolatok öröklődnek az az_egy kapcsolatok mentén.

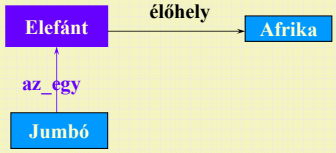
Örökölhető egyszerű kapcsolatok

- Speciális öröklődést biztosító kapcsolatoknál meg lehet adni, mely tulajdonságok öröklődhetnek.



Nem örökölhető egyszerű kapcsolatok

- A magasabb rendű kapcsolatok mindig örökölhetők, de az egyszerű kapcsolatok ritkán.

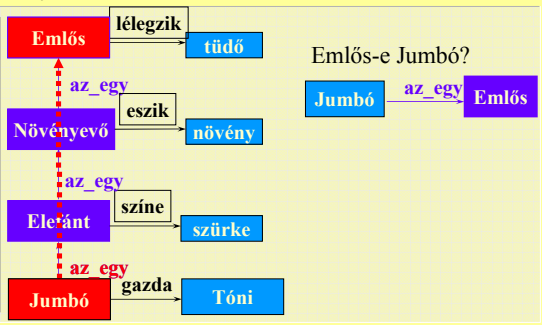


Öröklődést biztosító kapcsolatok

- Az *az_egy* általános öröklődést biztosító kapcsolat.
- Bevezethetünk olyan speciális kapcsolatokat is, amelyek mentén csak bizonyos tulajdonságok öröklődhetnek.
- Összekapcsolódó öröklődést biztosító élek egy öröklődést biztosító láncot alkotnak.

Közvetett illesztés

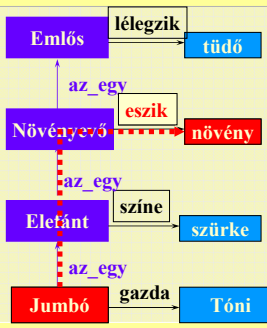
tényháló célháló



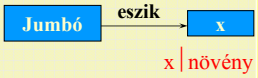
Emlős-e Jumbó?

Közvetett illesztés

tényháló célháló

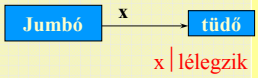


Mit eszik Jumbó?



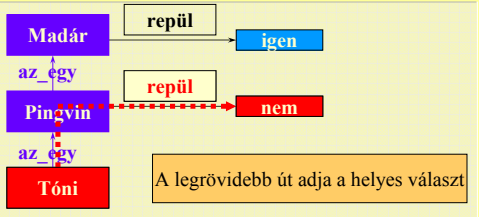
x | növény

Mire való Jumbó tüdeje?



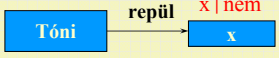
x | lélegzik

Kivételek kezelése



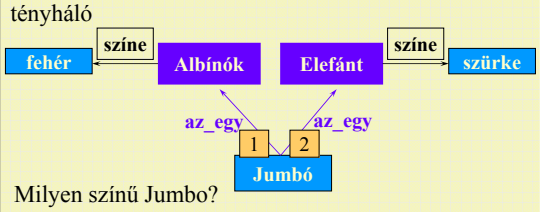
A legrövidebb út adja a helyes választ

Repül-e Tóni?

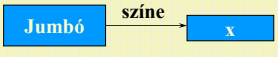


Prioritás vagy alapértelmezés

tényháló



Milyen színű Jumbo?

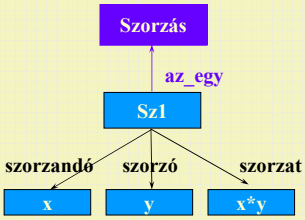


Bizonytalanság kezelés

Semmi akadálya bizonytalansági mértékek feltűntetésének:

Procedurális hozzárendelés

Démonok: procedurális hozzárendelést végző algoritmusok



Értékelés

- A szemantikus hálók kifejező ereje korlátozott az első rendű logikához képest,
- a vele végzett következtetés általánosabb, hiszen tartalmazza az alternatív következtetési technikákat
- Megvalósítás: frame alapú nyelvek