

Logika és informatikai alkalmazásai

Iván Szabolcs

2023 tavasz

A kurzusról

- PTI BSc második féléven kötelező
 - mérnökinfó, gazdinfó spinoff másik kurzus ők azt vegyék fel
 - Előfeltétel: dimategy gyak easy
 - Jelenlétiben vagyunk, TIK pince hétfő 18-20 mondjuk idetalálatok
 - 2020-as ea playlist YouTube-on fenn van bejönni nem kötelező, nem óvoda ez
 - discord szoba, discord DM, coospace üzi, email
 - coospace hirdetések, discord tárgyas szoba pinek
 - Előadás tesztek olyanoknak kell megismételni a kurzust. . .
 - idén **nincs** kötelező eateszt kitöltés
 - időnként lesz, ha kitöltöd, megtudod, vizsgán mit érne, ha nem, nem
 - Vizsga csak ha megvan az átmenő gyakjegy
 - coospace beugró, 10 darab gondolkodós kérdés, 10×4 pont
 - mindent lehet használni, kivéve realtime másik személyt
 - nem lesz előadás/gyakanyag ctrlF kérdés
 - pass/fail 40 – 60 százalék közt lesz valahol curve-based, vizsgánként
 - erre kettes jár, vagy szóbeli 1-5 jegyért
- „kettesemet sétáltatom” vagy „sziauram, szóbeli érdekel?”

- Ítéletkalkulus: következtetési algoritmusok. SAT solverek.
- Elsőrendű logika: upgradelt következtetési algoritmusok. Code contractok.
- Másodrendű logika.

Ezen kívül abban is szerzünk némi rutint, hogy hogyan is lehet **bebizonyítani** állításokat.

Definíció

```
size_t my_size_of( char* c ){  
    size_t i = 0;  
    while ( c[i] ) { i++; }  
    return i;  
}
```

Állítás, lemma, segédteétel, tétel, whatever

A my_size_of függvény egy 0-terminált C string hosszát adja vissza.

legalábbis ha addig végig mienk a memória

Kommentbox

térkitöltő jelleggel

Példa

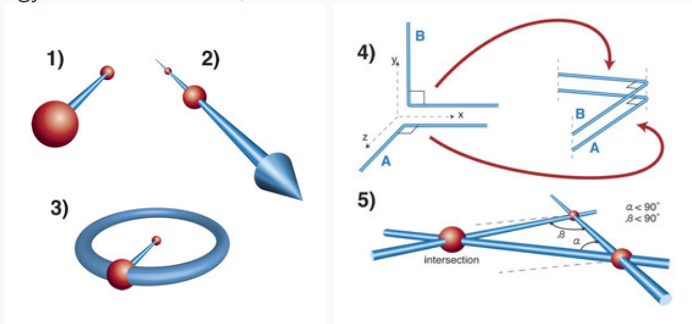
példa

Bizonyítás (-like érvelés, semiautomatizálható)

- A futás minden pillanatában igaz, hogy „minden $0 \leq j < i$ -re $c[j] \neq 0$ ”:
- Kezdetben $i = 0$, ilyen j nincs is, „triviálisan teljesül”
- Ha belépünk a ciklusmagba, akkor még ezen felül $c[i] \neq 0$ is igaz
- Tehát ekkor i -t megnövelve továbbra is igaz marad az állítás
- A ciklusmagból kilépéskor $c[i] = 0$ és az **invariáns** miatt ez az első ilyen index $\Rightarrow$ ha ekkor visszaadjuk, az épp a string hossza lesz

Background: Euklidesz

1. „bármely két pont összeköthető egy egyenes szakasszal”
2. „bármely egyenes szakasz tetszőlegesen meghosszabbítható”
3. „bármely pont körül bármekkora sugárral lehet kört rajzolni”
4. „két derékszög mindig egyforma”
5. „ha egy egyenest két másik összesen kisebb, mint 180 fokban metsz, akkor metszik egymást” **not so sure, but ok**



source: <https://plus.maths.org>

Background: Euklidesz

- A geometriai **tételek** ezekből az **axiómákból** következnek, **logikai következtetésekkel**.
- Axióma: (az adott terület) igaznak elfogadott alapigazságai
 - de tkp. bármit „kikiálthatunk” axiómának – ld Bolyai geometria
- Következtetés: stay tuned well, erről szól tkp a kurzus
- Ugyanez igaz pl. a **halmazelméletre**, **csoportelméletre** stb.
 - Kb. a XIX. század vége óta a matematika formális nyelve a logika
 - Cauchy „tétele” folytonos függvénytörvényekről
Cauchy felcserélt véletlen két kvantort. **Don't do that.**
 - Russel paradoxona az összes halmaz halmazáról
az „önmagukat nem tartalmazó halmazok halmaza” tartalmazza önmagát?
az összes halmaz halmazának a hatványhalmaza ugyanakkora, mint ő? Cantor says NO but yes
á ez akkor nem is halmaz, jelentsen ez bármit
- Informatika: a következtetést **algoritmikusan** végezzük, lehetőleg (legalább félig) **automatikusan**
 - ezt már Leibniz kigondolta a XVII. században
 - majd Hilbert is a XX. század elején
 - Gödel pedig megmutatta, hogy **ez nem megy**

- **Áramkörök** tervezése: egy áramkör egy logikai függvényt implementál
- **Automata**elmélet: „reguláris” nyelv mindig definiálható egy bizonyos („monadikus másodrendű”) logikában fonya, áfonya
- **Adatbázisok**: az SQL queryk mind elsőrendű logikai lekérdezések adatbé
- Logikai programozás: a **Prolog** egy elsőrendű logikai következtető motor, programozási nyelvnek álcázva prognyelvek
- Rendszerek **verifikáció**ja: egy metódus kielégíti-e a (valamilyen logikában megadott) specifikációt hsrv
- **Mesterséges intelligencia**: sok esetben (maybe fuzzy) logikát használ mestint
- **Bonyolultságelmélet**: a legtöbb „bonyolultsági osztály” pontosan karakterizálható egy-egy logikával bonya
- Programozási nyelvek **szemantikája**: az elemi utasítások elő- és utófeltételei, hatásuk formalizálva van alga
nem php, tho

Informatika: a SAT probléma

- a **SAT** (satisfiability) probléma: input egy (ítéletkalkulusbeli) logikai formula, adjuk meg a változóknak egy kielégítő értékadását! (Ha van.)
- A probléma **nehéz**: n változó – 2^n lehetőség bonyi: „NP-nehéz”
- 2^{100} művelet elvégzése kb. 4 évbe telne a Föld összes jelenlegi számítási kapacitását egyszerre használva (becslés, 2017)
 - 2^{120} : 4 millió év
 - 2^{200} : 5×10^{30} év – annyi nincs
- Heurisztikák kellene
- Minden évben van **SAT Competition**, amin SAT solvereket versenyeztetnek több tracken
- 2016 ősszel: egy laptopon kb. 900-változós formulát kb. 0.5sec alatt soon
- A XXI. században intenzíven fejlődik a terület
 - 1995-es évek: kb. 100 változóra 200 feltétel
 - 2010: kb. 1.000.000 változóra 5.000.000 feltétel

SAT, de minek?

- SAT Competition egyik track: **Application Benchmark**
 - Erre cégek küldik be az őket érdeklő nehéz problémákat **logikai formulával leírva**
 - 2016-ban pl. a francia vasúthálózat forgalomirányításáról is kérdezték, biztonságos-e
- IBM: egy hardware egység teljesíti-e a specifikációt
 - 2006-os SAT Race-re (ez Industrial Only) formulával leírva: 170.000 változó, 725.000 feltétel
 - erre egyébként 500+ ipari benchmark érkezett
- Az Intelnél, IBM-nél és Microsoftnál ma a SAT solving a domináns technológia a hardware tervek verifikációjára
- AI Planning: döntéshozatal egy vagy több cél elérése érdekében, erőforrást optimalizálva – ez is felírható formulával
- Handbook of SAT - 2009, 966 oldal

SAT, de minek?

- **Nagyon sok** „kombinatorikus” keresési problémát fel lehet írni SATként.
- Ekkor ahelyett, hogy mi írjunk keresőalgoritmust:
 - írunk egy kódot, ami átkonvertálja a problémát formulává
 - megoldatjuk a formulát egy SAT solverrel
 - a visszaadott eredményt visszakonvertáljuk az eredeti probléma megoldásává
 - ehhez jellemzően sokkal kevesebbet kell kódolni, mintha az eredeti problémára írnánk solver
 - debugolni is
- Egy use case-t látni fogunk hamarosan: a tatami coveringet.

Egy másik gyakori alkalmazás: a Code Contractok ellenőrzése

(ezen a kurzuson: „Hoare kalkulus”)

Code contractok – C + ACSL (ANSI C Spec Lang)

```
int f(int A, int B) {  
    int Q = 0, R = A;  
    while (R >= B) {  
        R -= B; Q++;  
    }  
    return R;  
}
```

A jobb oldali annotációk segítségével

automatikusan

ellenőrizhető, hogy a
függvény implementációja
helyes

```
//@ requires A >= 0 && B > 0;  
//@ ensures 0 <= \result < B;  
//@ ensures \exists integer i; i * B + \result == A  
int f(int A, int B) {  
    int Q = 0, R = A;  
    //@ assert A>=0 && B>0 && Q==0 && R==A;  
    while (R >= B) {  
        //@ assert A>=0 && B>0 && R>=B && A==Q*B+R;  
        R -= B; Q++;  
    }  
    //@ assert A>=0 && B>0 && R>=0 && R<B && A==Q*B+R;  
    return R;  
}
```

btw ez csak előadáson lesz

Code contractok – C + ACSL (ANSI C Spec Lang)

```
/*@  
  requires IsValidRange(a,n);  
  assigns \nothing;  
  behavior some:  
    assumes \exists integer i; 0 <= i < n && a[i] == val;  
    ensures 0 <= \result < n;  
    ensures a[\result] == val;  
    ensures \forall integer i; 0 <= i < \result ==> a[i] != val;  
  behavior none:  
    assumes \forall integer i; 0 <= i < n ==> a[i] != val;  
    ensures \result == n;  
  complete behaviors;  
  disjoint behaviors;  
*/  
size_type find(const value_type *a, size_type n, value_type val);
```

Code contractok – Java + ESC (Extended Static Checker)

```
public class OrderedArray {  
    int a[];  
    int nb;  
    //@invariant nb >= 0 && nb <= 20  
    //@invariant (\forall int i; (i >= 0 && i < nb-1) ==> a[i] <= a[i+1])  
    public OrderedArray() { a = new int[20]; nb = 0; }  
    public void add(int v) {  
        if (nb >= 20) return;  
        int i;  
        for (i=nb; i > 0 && a[i-1] > v; i--) a[i] = a[i-1];  
        a[i] = v; nb++;  
    }  
}
```

Az ESC **automatikusan** le tudja ellenőrizni, hogy egy OrderedArray a teljes élettartama során tényleg rendezett sorrendben fogja tárolni az elemeket

Dafny (by MS, compiles to C#, Java, JS, Go)

```
function fib(n: nat) : nat {  
  if ( n < 2 ) then n else fib(n - 1) + fib(n - 2)  
}  
  
method computeFib(n: nat) returns (x: nat)  
  ensures x == fib(n)  
{  
  var i := 0;  
  x := 0;  
  var y := 1;  
  while( i < n )  
    invariant 0 <= i <= n;  
    invariant x == fib(i);  
    invariant y == fib(i + 1);  
  {  
    x, y := y, x + y;  
    i := i + 1;  
  }  
}
```

Rekurzív, helyes $O(1.62^n)$

Iteratív, gyors $O(n)$

automatikusan ellenőrizhető:
ugyanazt adják

némi hinttel

Ítéletkalkulus vs. elsőrendű logika

A kurzus első felében **ítéletkalkulussal** fogunk foglalkozni:

- a változók a $\{0, 1\}$ halmazból kapnak értéket (0: hamis, 1: igaz – igazságértékek, bitek)
- a formulák változókból épülnek fel ítéletlogikai összekötő jelek (**konnektívák**, mint a $\neg$ és a $\vee$, sőt a $\downarrow$) alkalmazásával (zárójelezve)

A második felében **elsőrendű logikával**:

- a változók **objektumok** egy halmazából kapnak értékeket
- a konnektívákon kívül **kvantorok** is használhatóak lesznek (mint a $\forall$)
- az objektumokat **függvények** fogják újabb objektumokba, és **predikátumok** fogják igazságértékké transzformálni

Az ítéletkalkulust hívjuk még ítéletlogikának vagy nulladrendű, esetleg propozicionális logikának is.

Az elsőrendű logikát pedig predikátumkalkulusnak.

Szintaxis és szemantika

Ha precízen akarunk beszélni valamiről, általában megadjuk a nyelvünk **szintaxisát** és **szemantikáját**.

Szintaxis

Mi az, amit leírhatunk? Milyen stringek fognak jelentést hordozni? Melyek a „jól formált kifejezések”? Milyen típusúak lesznek?

- progalap: mi lehet egy azonosító? mi a függvény fejléc? mi egy függvénytörzs? mi egy értékadás?
- logika: mi a formula?

Szemantika

Mit jelent, amit leírhatunk a szintaxis szabályai szerint? Hogy értékeljük ki? Adott környezetben mi az eredménye / értéke?

- progalap: mit csinál egy értékadás? mit egy összeadás? mit egy függvényhívás?
- logika: mi a formula értéke egy adott változó-értékadás mellett?

Változók

Rögzítjük (ítélet)változóknak egy $\{p_1, p_2, \dots\}$ (végtelen) halmazát.

A változókat általában $p, q, r, p_1, p_2, p', \dots$ jelöli majd.

Logikai konstansjelek

A $\uparrow$ („igaz”) és $\downarrow$ („hamis”) jelek.

Logikai konnektívák

A $\wedge$ (konjunkció, és), $\vee$ (diszjunkció, vagy), $\neg$ (negáció, nem), $\rightarrow$ (implikáció, nyíl) és $\leftrightarrow$ (akkor és csak akkor, pontosan akkor, duplanyíl) jelek.

záróvizsgán ne mondd nyílnak, van aki nem szereti

Kényelmi okokból a konstansjeleket „nulla változós” konnektívaként kezeljük.

Formulák

- Minden változó és minden logikai konstans formula
- Ha F formula, akkor $(\neg F)$ is formula
- Ha F és G formulák, akkor $(F \wedge G)$, $(F \vee G)$, $(F \rightarrow G)$, $(F \leftrightarrow G)$ is formulák
- Más formula nincs

Az ehhez hasonló „ez meg ez meg ez X , más X nincs” alakú konstrukciókat úgy is szokták mondani, hogy „az X -ek halmaza a **legsűkebb olyan Y halmaz, melyre** ez meg ez meg ez Y ”.

Formulák például: p , $(\neg p)$, $((\neg p) \vee q)$, $(\neg \rightarrow p)$ és

$$((\neg(((\neg p) \vee q) \vee r)) \rightarrow (\neg q)).$$

Szintaxis: implicit zárójelek $((\neg(((\neg p) \vee q) \vee r)) \rightarrow (\neg q))$

Az olvashatóság kedvéért egyes zárójeleket elhagyunk:

- a legkülsőket $(\neg(((\neg p) \vee q) \vee r)) \rightarrow (\neg q)$
- a következő precedencia-sorrend szerint elhagyhatókat:
legerősebb a $\neg$, majd $\wedge$, $\vee$, $\rightarrow$ és végül a leggyengébb a $\leftrightarrow$
 $\neg((\neg p \vee q) \vee r) \rightarrow \neg q$
- a $\wedge$ és $\vee$ műveletek **asszociatívak**
pl. $(F \vee G) \vee H$ helyett $F \vee G \vee H$ -t írunk
de **nem** egymás közt! $(F \vee G) \wedge H$ így marad
 $\neg(\neg p \vee q \vee r) \rightarrow \neg q$
- a $\rightarrow$ művelet **jobb-asszociatív**
 $F \rightarrow G \rightarrow H$ az $F \rightarrow (G \rightarrow H)$ zárójelezést jelenti

Ezekkel az előző formulából ez lesz:

$$\neg(\neg p \vee q \vee r) \rightarrow \neg q$$

Hogy a konnektívák szemantikájáról tudjunk beszélni, mindhez rendelünk egy **Boole-függvényt**:

(n -változós) Boole-függvény

Bitvektort egy bitbe képző függvény: $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

Az f/n jelzi, hogy az f egy **n -változós** függvény.

A $\neg$ **unáris** (= egyváltozós) Boole-függvény: $\neg 0 = 1, \neg 1 = 0$.

A bináris (= kétváltozós) konnektívákhoz rendelt Boole-függvények **igazságtáblája**:

x	y	$x \vee y$	$x \wedge y$	$x \rightarrow y$	$x \leftrightarrow y$
0	0	0	0	1	1
0	1	1	0	1	0
1	0	1	0	0	0
1	1	1	1	1	1

Itt azért használunk x -et és y -t, mert **függvényeket** általában az $x, y, z, \dots$ változókkal szoktunk deklarálni.

Pl. $1 \vee 0 = 1, 0 \rightarrow 1 = 1, 1 \rightarrow 0 = 0$. (írhatnánk $\vee(1, 0) = 1$ -nek is, maradunk az **infix** jelölésnél.)

(Egy n -változós Boole-függvény igazságtáblája 2^n soros.)

Hogy egy formulát ki tudjunk értékelni, kell egy **(változó)értékadás**:

Értékadás

Egy $\mathcal{A}$ függvény, mely minden változóhoz egy igazságértéket (bitet: 0 vagy 1) rendel.

Az $\mathcal{A}$ értékadás mellett az F formula értékét $\mathcal{A}(F)$ jelöli:

$$\mathcal{A}(p) := \mathcal{A}(p) \quad \text{warning: not the same } \mathcal{A}$$

$$\mathcal{A}(\uparrow) := 1$$

$$\mathcal{A}(\downarrow) := 0$$

$$\mathcal{A}(\neg F) := \neg \mathcal{A}(F)$$

$$\mathcal{A}(F \vee G) := \mathcal{A}(F) \vee \mathcal{A}(G)$$

$$\mathcal{A}(F \wedge G) := \mathcal{A}(F) \wedge \mathcal{A}(G)$$

$$\mathcal{A}(F \rightarrow G) := \mathcal{A}(F) \rightarrow \mathcal{A}(G)$$

$$\mathcal{A}(F \leftrightarrow G) := \mathcal{A}(F) \leftrightarrow \mathcal{A}(G)$$

Magyarul

Tehát **rekurzívan** kiértékeljük az „eggyel egyszerűbb” formulákat és a **legkülső** konnektívának megfelelően kombináljuk az értékeket.

„**kiterjesztése**” az értékadásnak mert változókra továbbra is ugyanazt adja vissza, mint eddig, de most már több mindenre van értelmezve

Formula-kiértékelés példa

Ha $F = (p \rightarrow q) \vee (\neg r \leftrightarrow p)$ és $\mathcal{A} : p \mapsto 1, q \mapsto 0, r \mapsto 0$, akkor

- $\mathcal{A}(p \rightarrow q) = 1 \rightarrow 0 = 0$
- $\mathcal{A}(\neg r) = \neg 0 = 1$
- $\mathcal{A}(\neg r \leftrightarrow p) = 1 \leftrightarrow 1 = 1$
- tehát $\mathcal{A}(F) = 0 \vee 1 = 1$.

Ha az $\mathcal{A}$ értékadásra és az F formulára $\mathcal{A}(F) = 1$, azt úgy is írjuk, hogy $\mathcal{A} \models F$ és úgy is mondjuk, hogy $\mathcal{A}$ **kielégtíti** F -et vagy $\mathcal{A}$ egy **modellje** F -nek vagy $\mathcal{A} \in \text{Mod}(F)$.

Ha egy formulának van modellje, akkor azt mondjuk, **kielégtíthető**;
ha nincs neki, **kielégtíthetetlen**.

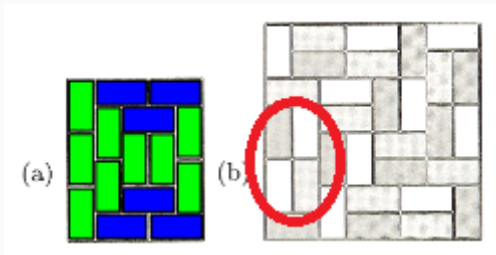
Ha az F formulának **minden** kiértékelés modellje, akkor **tautológia**.

Ennek jele $\models F$.

Honnan lesz formulánk?

Vegyük pl. a következő **kombinatorikus keresési feladatot**.

- Adott egy $n \times m$ -es téglalap, melyet 2×1 -es dominókkal (forgatni ér) szeretnénk lefedni.
- A lefedésnek hogy „szép” legyen, **tatami** lefedésnek kell lennie: négy dominó nem találkozhat egy sarkon.
- Néhány dominó előre fel van rakva a táblára.
- Adjunk meg egy tatami lefedését a táblának, melyben a megadott dominók a megadott módon szerepelnek!



Modellezés formula-kielégítési problémaként (SATként)

A táblán minden **szomszédos mezőket összekötő élhez** rendelünk egy változót, pl.

- $p_{i,j}$ legyen az i . sor j . oszlopából jobbra menő él,
- $q_{i,j}$ pedig a felfele menő él,

amilyen i, j -kre ilyen élek vannak. Pl. egy 6×5 -ös táblán:

$p_{6,1}$	$p_{6,2}$	$p_{6,3}$	$p_{6,4}$
$q_{5,1}$	$q_{5,2}$	$q_{5,3}$	$q_{5,4}$
$p_{5,1}$	$p_{5,2}$	$p_{5,3}$	$p_{5,4}$
$q_{4,1}$	$q_{4,2}$	$q_{4,3}$	$q_{4,4}$
$p_{4,1}$	$p_{4,2}$	$p_{4,3}$	$p_{4,4}$
$q_{3,1}$	$q_{3,2}$	$q_{3,3}$	$q_{3,4}$
$p_{3,1}$	$p_{3,2}$	$p_{3,3}$	$p_{3,4}$
$q_{2,1}$	$q_{2,2}$	$q_{2,3}$	$q_{2,4}$
$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$
$q_{1,1}$	$q_{1,2}$	$q_{1,3}$	$q_{1,4}$
$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$

Az elképzelés: azokat a változókat (éleket) állítsuk 1-re, amiket egy dominó fed le.

	$p_{6,2}$	$p_{6,4}$
$q_{5,1}$		$p_{5,3}$
	$q_{4,2}$	$q_{4,5}$
$q_{3,1}$	$q_{3,3}$	$q_{3,4}$
	$q_{2,2}$	$q_{2,5}$
$q_{1,1}$	$p_{2,3}$	
	$p_{1,2}$	$p_{1,4}$

Modellezés SATként

Azt kell megfogalmaznunk, hogy mikor felel meg egy **értékadás** egy megoldásnak, vagyis egy **tatami lefedésnek**.

- Minden **mezőt** lefed legalább egy dominó: vagyoljuk a rá illeszkedő éleket
- Minden mezőt legfeljebb egy dominó fed: az egy mezőre illeszkedő éleket **nand** kapcsolatba hozzuk: $\neg(x \wedge y)$
- A tatami feltétel: minden „sarok mellett” van igaz változó – ezeket is vagyoljuk

A fenti feltételeket pedig összeésszük. Részlet:

- $(p_{4,2} \vee q_{4,2} \vee p_{4,1} \vee q_{3,2})$ – a 4. sor 2. mezőjét fedi egy dominó;
- $\neg(p_{4,2} \wedge q_{4,2})$ – ezt a mezőt nem fedi egyszerre fentről és jobbról egy dominó;
- $(p_{3,1} \vee q_{2,1} \vee p_{2,1} \vee q_{2,2})$ – a 2. sor 1. oszlop sarkán nem érintkezik négy sarok

Plusz: az előre megadott dominóknak megfelelő változókat 1-re állítjuk (hozzáésselve őket az egész formulához)

- A megadott teszteken (30×30 -as tábla, kb 30 ledobott dominó) ez kb. 900 változó és 7.000 feltétel
- Ez egy mai SAT solvernek nem méret
 - unless ha a formula szándékosan „nehéz” SAT példánynak készült
- A formalizálás implementálása, oda-vissza konverzió tatami lefedés és formula közt: félóra, tops

A mai SAT solverek jók és egyre jobbak $\Rightarrow$ mindig jó ötlet elgondolkodni rajta, hogy az aktuális problémánkat formulává tudjuk-e konvertálni hasonló módon

More of this © bonya

A modellek halmaza

$\text{Mod}(F)$

Ha F egy formula, akkor $\text{Mod}(F)$ az F összes modelljének a halmaza.

Tehát hogy $\mathcal{A}(F) = 1$, vagy $\mathcal{A} \models F$, úgy is írhatjuk, hogy $\mathcal{A} \in \text{Mod}(F)$.

Pl. ha $\mathcal{A}(p) = 1$, $\mathcal{A}(q) = 0$, $\mathcal{A}(r) = 0$, akkor

$$\mathcal{A} \in \text{Mod}\left((p \rightarrow q) \vee (\neg r \leftrightarrow p)\right).$$

Így pl. F pontosan akkor kielégíthetetlen, ha $\text{Mod}(F) = \emptyset$.

Ha Σ formulák egy halmaza és $\mathcal{A}$ egy értékadás, akkor $\mathcal{A} \models \Sigma$ azt jelenti, hogy $\mathcal{A}$ kielégíti Σ összes elemét.

Hasonlóan $\text{Mod}(\Sigma)$ -ba azok az értékadások tartoznak, melyek kielégítik Σ összes elemét.

Pl. $\text{Mod}(\emptyset)$ -ba az összes értékadás beletartozik, mert egyik sem sért meg egy feltételt sem a nullából

Tautológiák és kielégíthetetlenség

Könnyű látni, hogy

Az F formula pontosan akkor tautológia, ha $\neg F$ kielégíthetetlen.

F tautológia	$\Leftrightarrow$ minden $\mathcal{A}$ -ra $\mathcal{A}(F) = 1$	tautológia def
	$\Leftrightarrow$ minden $\mathcal{A}$ -ra $\neg \mathcal{A}(F) = 0$	negálás függvény def
	$\Leftrightarrow$ minden $\mathcal{A}$ -ra $\mathcal{A}(\neg F) = 0$	negált formula szemantika def
	$\Leftrightarrow \neg F$ kielégíthetetlen.	kielégíthetetlenség def

A tautológiák persze kielégíthetők.

Logikai következmény

$A \models$ jel egy **másik** overloadja:

Ha F és G formulák, akkor $F \models G$ („ F -nek logikai következménye G ”) azt jelöli, hogy minden $\mathcal{A}$ -ra **ha** $\mathcal{A}(F) = 1$, **akkor** $\mathcal{A}(G) = 1$.

azaz: $\text{Mod}(F) \subseteq \text{Mod}(G)$

„ F modellje G -nek”, „ F kielégíti G -t”
viszont **hülyeség**

Például $(p \wedge q) \vee r \models \neg p \rightarrow r$:

p	q	r	$(p \wedge q) \vee r$	$\neg p \rightarrow r$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1

p	q	r	$(p \wedge q) \vee r$	$\neg p \rightarrow r$
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Logikai következmény formulahalmazok közt

Ugyanígy használhatjuk a $\Sigma \models F$, $\Sigma \models \Gamma$ jelöléseket is, ahol Σ , Γ formula

halmazok

:
pl. $\Sigma \models F$ akkor áll fenn, ha minden Σ minden modellje modellje F -nek is.

például

$$\{p, p \rightarrow q\} \models q$$

hiszen ha egy $\mathcal{A}$ értékadásban p is és $p \rightarrow q$ is igaz, akkor q is igaz.

Általában arra vagyunk kíváncsiak, hogy egy F formula következik-e axiómák egy Σ halmazából.

Az $F \equiv G$ („ F ekvivalens G -vel”) jelölés pedig azt jelenti, hogy $\text{Mod}(F) = \text{Mod}(G)$ (tehát $F \models G$ és $G \models F$).

$\text{Mod}(\Sigma) \cap \text{Mod}(\Gamma)$

Hasznos tudnunk a következőt:

$$\text{Mod}(\Sigma \cup \Gamma) = \text{Mod}(\Sigma) \cap \text{Mod}(\Gamma).$$

- a bal oldalon szereplő halmazban azok az értékadások vannak, melyek kielégítik $\Sigma \cup \Gamma$ összes elemét
- azaz Σ összes elemét is és Γ összes elemét is
- azaz melyek benne vannak $\text{Mod}(\Sigma)$ -ban is és $\text{Mod}(\Gamma)$ -ban is
- ez pedig épp a jobb oldal

Nyilván az is igaz, hogy

tetszőleges $\mathcal{A}$ értékadás vagy $\text{Mod}(F)$ -ben, vagy $\text{Mod}(\neg F)$ -ben szerepel (pontosan az egyikükben), ha F egy formula.

how to speak math: $\text{Mod}(\Sigma) \cap \text{Mod}(\Gamma)$, mechanikusan

Memó: ha egy H halmazt a következőképp deklarálunk

$$H := \{x \in T : \text{valami feltétel, amiben szerepel az } x \text{ „szabadon”}\}$$

az azt jelenti, hogy H -ban pontosan azok a T -beli („ T típusú”, általában) dolgok vannak, amikre igaz a kettőspont utáni feltétel

$$\{n \in \mathbb{N}_0 : n^2 \equiv 3 \pmod{5}\}$$

igen, ez egy másik $\equiv$ jel

„azoknak a természetes számoknak a halmaza, amiknek a négyzete öttel osztva hármat ad maradékul”

ez épp üres halmaz – attól, hogy két halmazt más módon definiáltunk, még lehet, hogy valójában ugyanaz a halmaz

van, hogy az $\in T$ típusdeklarációt nem adjuk meg, ha kikövetkeztethető a feltételből

how to speak math: $\text{Mod}(\Sigma) \cap \text{Mod}(\Gamma)$, mechanikusan

$\mathcal{A}$ eleme $\text{Mod}(\Sigma \cup \Gamma)$ -nak pont akkor, ha

- minden F formulára ha $F \in \Sigma \cup \Gamma$, akkor $\mathcal{A} \models F$ Mod def
- minden F formulára ha $(F \in \Sigma \text{ vagy } F \in \Gamma)$, akkor $\mathcal{A} \models F$ unió def
- minden F formulára
 $(\text{ha } F \in \Sigma, \text{ akkor } \mathcal{A} \models F) \text{ és } (\text{ha } F \in \Gamma, \text{ akkor } \mathcal{A} \models F)$
mert $(p \vee q) \rightarrow r \equiv (p \rightarrow r) \wedge (q \rightarrow r)$
- (minden F formulára ha $F \in \Sigma$, akkor $\mathcal{A} \models F$)
és
(minden F formulára ha $F \in \Gamma$, akkor $\mathcal{A} \models F$)
mert $\forall x(F \wedge G) \equiv (\forall xF) \wedge (\forall xG)$ (laters – believe me for a moment, it's true)
- $\mathcal{A} \in \text{Mod}(\Sigma)$ és $\mathcal{A} \in \text{Mod}(\Gamma)$ Mod def
- $\mathcal{A} \in \text{Mod}(\Sigma) \cap \text{Mod}(\Gamma)$ metszet def

erre a két halmazra **nem** mondjuk, hogy „def alapján egyenlőek”, mert használunk mást is az egyik oldalból a másikká való transzformálás közben

Az indirekt bizonyítás

Ha következtető-motort akarunk fejleszteni, ahhoz elég a kielégíthetatlenséggel foglalkoznunk:

$\Sigma \models F$ pontosan akkor igaz, ha $\Sigma \cup \{\neg F\}$ kielégíthetetlen.

(gépies) bizonyítás: $\Sigma \models F$ pont akkor, ha

- $\text{Mod}(\Sigma) \subseteq \text{Mod}(F)$ $\models$ def
- minden $\mathcal{A}$ -ra: ha $\mathcal{A} \in \text{Mod}(\Sigma)$, akkor $\mathcal{A} \in \text{Mod}(F)$ részhalmoz def
- minden $\mathcal{A}$ -ra: ha $\mathcal{A} \in \text{Mod}(\Sigma)$, akkor $\mathcal{A}(F) = 1$ Mod def
- minden $\mathcal{A}$ -ra: ha $\mathcal{A} \in \text{Mod}(\Sigma)$, akkor $\neg \mathcal{A}(F) = 0$ $\neg$ függvény defből
- minden $\mathcal{A}$ -ra: ha $\mathcal{A} \in \text{Mod}(\Sigma)$, akkor $\mathcal{A}(\neg F) = 0$ $\neg$ szemantika def
- minden $\mathcal{A}$ -ra: ha $\mathcal{A} \in \text{Mod}(\Sigma)$, akkor $\mathcal{A} \notin \text{Mod}(\neg F)$ Mod def
- minden $\mathcal{A}$ -ra: **nem** $\left(\mathcal{A} \in \text{Mod}(\Sigma) \text{ és } \mathcal{A} \in \text{Mod}(\neg F) \right)$ $p \rightarrow \neg q \equiv \neg(p \wedge q)$
- minden $\mathcal{A}$ -ra: **nem** $\left(\mathcal{A} \in \text{Mod}(\Sigma) \cap \text{Mod}(\neg F) \right)$ metszet def
- $\text{Mod}(\Sigma) \cap \text{Mod}(\neg F) = \emptyset$ $x = \emptyset \leftrightarrow \forall y (y \notin x)$ halmazelméleti axióma
- $\text{Mod}(\Sigma \cup \{\neg F\}) = \emptyset$ előző „lemma”33

Konjunktív normálforma (CNF)

A következő algoritmusainkat (hogy ne legyen benne sok eset, uniforman működjön) valamilyen **normálformában** lévő formulákra szoktuk specifikálni. Az egyik leggyakrabban alkalmazott normálforma a **konjunktív normálforma**:

- Az **ítéleváltozókat** és **negáltjaikat** literáloknak nevezzük;
- Véges sok literál **diszjunkcióját** klóznak;
- Véges sok klóz **konjunkcióját** pedig **konjunktív normálformának**, CNF-nek.

$$(p \vee \neg q) \wedge (\neg p \vee \neg q \vee r) \wedge p$$

Itt

- p, q, r változók,
- $p, \neg q, \neg p, r$ literálok,
- $(p \vee \neg q)$, $(\neg p \vee \neg q \vee r)$ és p klózok.

Az egyelemű klózokat (mint itt a p) **egységklóznak** nevezzük.

Minden formula ekvivalens CNF alakra hozható.

- Először a $\rightarrow$ és $\leftrightarrow$ konnektívákat elimináljuk a formulából a következő ekvivalenciákkal:

$$F \rightarrow G \equiv \neg F \vee G$$

$$F \leftrightarrow G \equiv (\neg F \vee G) \wedge (F \vee \neg G)$$

ok, also $\uparrow \vee F \equiv \uparrow$, $\downarrow \vee F \equiv F$, $\downarrow \rightarrow F \equiv \uparrow$ stb.

- Majd a $\neg$ jeleket visszük le a változók mellé a **deMorgan azonosságokkal**:

$$\neg(F \vee G) \equiv \neg F \wedge \neg G$$

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

$$\neg\neg F \equiv F$$

(Ekkorra a formula **negáció normálformában**, NNF van.)

- Végül a $\vee$ jeleket visszük be a $\wedge$ jelek alá a **disztributivitás** alkalmazásával:

$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H), \quad (F \wedge G) \vee H \equiv (F \vee H) \wedge (G \vee H)$$

CNF példa

A formula:

$$(p \vee q) \leftrightarrow (q \rightarrow r)$$

Nyilak eliminálása:

$$(\neg(p \vee q) \vee (q \rightarrow r)) \wedge ((p \vee q) \vee \neg(q \rightarrow r))$$

$$(\neg(\underline{p \vee q}) \vee (\neg q \vee r)) \wedge ((p \vee q) \vee \neg(\underline{\neg q \vee r}))$$

Negálások bevitele:

$$((\neg p \wedge \neg q) \vee (\neg q \vee r)) \wedge ((p \vee q) \vee (\neg \neg q \wedge \neg r))$$

$$((\underline{\neg p} \wedge \underline{\neg q}) \vee (\underline{\neg q} \vee r)) \wedge ((\underline{p \vee q}) \vee (\underline{q} \wedge \underline{\neg r}))$$

Disztributivitás:

$$(\underline{\neg p} \vee \underline{\neg q} \vee r) \wedge (\underline{\neg q} \vee \underline{\neg q} \vee r) \wedge (\underline{p \vee q} \vee \underline{q}) \wedge (\underline{p \vee q} \vee \underline{\neg r})$$

klózok vagyolásokon, CNFek éseléseken terjednek keresztül

Halmazok

Tehát a probléma, amivel foglalkozunk:

SAT

Input: egy CNF.

Output: kielégíthető-e?

A CNF-eket nem stringként reprezentáljuk, hanem

- egy klózt a benne literálok **halmazaként**,
- egy CNF-et pedig klózainak **halmazaként**.

Ezt megtehetjük a $\vee$ és $\wedge$ műveletek kommutativitása, asszociativitása és idempotenciája miatt (azaz: sem a sorrend, sem a multiplicitás nem számít).

Az előző formula ebben a reprezentációban pl:

$$\Sigma = \left\{ \{ \neg p, \neg q, r \}, \{ \neg q, r \}, \{ p, q \}, \{ p, q, \neg r \} \right\}.$$

Üres klóz, üres CNF

Még ha az inputban nincs is, az algoritmusok generálhatnak **üres** klózt (jele a $\square$ lesz).

Az üres klóz minden értékadás mellett **hamis**.

(Pl. mert tetszőleges C és D klózokra és $\mathcal{A}$ értékadásra igaz kell legyen, hogy $\mathcal{A}(C \cup D) = \mathcal{A}(C) \vee \mathcal{A}(D)$ és ha $D = \square$, akkor ez $\mathcal{A}(C) = \mathcal{A}(C \cup \square) = \mathcal{A}(C) \vee \mathcal{A}(\square)$ -ot jelenti, ami akkor igaz, ha $\mathcal{A}(\square) = 0$.)
Az üres CNF (tehát 0 darab klóz halmaza) jele a szokásos üreshalmaz-jel, $\emptyset$ lesz.

Az üres CNF minden értékadás mellett **igaz**.

$\{\}$ igaz, ha CNF és hamis, ha klóz.

$\square$ hamis. $\emptyset$ igaz.

Közvetlen részformulák

Egy formula közvetlen részformulái az „eggyel lentebbi szinten lévő részei”:

Közvetlen részformula

- A változóknak és a logikai konstansoknak nincs közvetlen részformulája;
- A $(\neg F)$ alakú formulák közvetlen részformulája F ;
- Az $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$ és $(F \leftrightarrow G)$ alakú formulák közvetlen részformulái F és G .

A formulák kiértékelését úgy végeztük el, hogy

- rekurzívan kiértékeljük a közvetlen részformulákat;
- majd az eredményekből és a külső konnektívából számítjuk az egész formula értékét.

Az ilyen rendszerű definíciókat és bizonyításokat **a formula felépítése szerinti indukciónak** nevezzük.

Felépítés szerinti indukció

Definíciókban csak meg kell mondjuk, hogy az aktuálisan a formulához rendelt objektumot hogyan számítjuk ki a részformuláihoz rendelt objektumokból, ügyelve arra, hogy minden esetet pontosan egyszer vegyünk sorra.

Bizonyításokban kicsit összetettebb a feladat: minden esetre meg kell mutatnunk, hogy ha az állítás igaz a formula összes közvetlen részformulájára, akkor **miért igaz** az egész formulára is.

Láttunk már hasonlót: pl. a természetes számokon a **teljes indukció** is így működik.

Teljes indukció memó (és Peano axiómák a számokról)

A természetes számok konstruktorai:

- 0 természetes szám;
- ha n természetes szám, akkor n' is az;
- más természetes szám nincs.

Az n' helyett lehet írni $\text{succ}(n)$ -t
vagy $(n + 1)$ -et is

Az összeadás **definíciója**:

$$m + 0 := m$$

(az az eset, amikor a jobb oldali argumentum 0)

$$m + (n') := (m + n)'$$

(amikor n')

A második esetben hogy összeadjuk m -et $n + 1$ -gyel, előbb összeadtuk m -et az „egyszerűbb” n -nel (rekurzív hívás az egyszerűbb számra), majd az eredményt növeltük eggyel (a természetes számok konstruktorát hívva).

A **szorzásé**:

$$m \star 0 := 0$$

$$m \star (n') := (m \star n) + m$$

„definíció szerint ennyi”

$$3 + 2 = 5$$

$$\begin{aligned} 0''' + 0'' &= (0''' + 0')' \\ &= ((0''' + 0)')' \\ &= ((0''')')' \\ &= 0''''' \end{aligned}$$

$$2 \star 2 = 4$$

$$\begin{aligned} 0'' \star 0'' &= (0'' \star 0') + 0'' \\ &= ((0'' \star 0) + 0'') + 0'' \\ &= (0 + 0'') + 0'' \\ &= 0''''' \end{aligned}$$

Teljes indukció memó

Ez pedig egy **bizonyítás**a annak az állításnak, hogy

$$1 + 2 + \dots + m = \frac{m(m+1)}{2}$$

teljesül minden m természetes számra:

- $m = 0$ -ra igaz: mindkét oldal 0
- $m = n + 1$ -re: itt írhatnánk n' -t is
 - Az **indukciós feltevés szerint** $1 + \dots + n = \frac{n(n+1)}{2}$ igaz a nála egyszerűbb n -re;
 - $1 + \dots + (n + 1) = (1 + \dots + n) + (n + 1)$ ezek szerint $\frac{n(n+1)}{2} + (n + 1)$
 - ami tovább egyenlő $\frac{n(n+1)+2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$ -vel
 - ami pont a fenti képlet, ha a számunk az $m = n + 1$. Done.

Formulákra is pontosan így működik az indukció, azzal, hogy

- teljes indukció helyett **strukturális indukciónak** vagy felépítés szerinti indukciónak hívjuk,
- több aleset van (nem csak 0 és n' , hanem $p, \uparrow, \downarrow, \neg F, F \vee G$ stb).

A CNF-re hozó algoritmus mindig megáll: megállási feltételek

- nyilak eliminálása: lehet a formulákhoz természetes szám súlyokat rendelni:

- $p, \uparrow, \downarrow$ súlya $w(p) = w(\uparrow) = w(\downarrow) = 1$
- $F \wedge G, F \vee G$ súlya $w(F) + w(G)$, $\neg F$ súlya $w(F)$
- $F \rightarrow G$ súlya $w(F) + w(G) + 1$, $F \leftrightarrow G$ súlya $2(w(F) + w(G)) + 1$

Ekkor átírás után mindig csökken a súly

pl $F \leftrightarrow G$ helyett $(\neg F \vee G) \wedge (F \vee \neg G)$ súlya $2(w(F) + w(G))$ lesz, eggyel kisebb

(és ha egy részformula súlya kisebbé válik, akkor az egésze is)

- de Morgan azonosságok: pl. $w(p) = 2$, $w(\neg F) = w(F)^2$,

$$w(F \vee G) = w(F \wedge G) = w(F) + w(G)$$

akkor pl. $w(\neg(F \vee G)) = (w(F) + w(G))^2$, $w(\neg F \wedge \neg G) = w(F)^2 + w(G)^2$, ami kisebb lesz

- disztributivitás: pl. $w(p) = 2$, $w(\neg F) = w(F)$, $w(F \wedge G) = w(F) + w(G) + 1$,
 $w(F \vee G) = w(F) \cdot w(G)$

akkor pl. $w(F \vee (G \wedge H)) = w(F) \cdot (w(G) + w(H) + 1)$,

$w((F \vee G) \wedge (F \vee H)) = w(F) \cdot (w(G) + w(H)) + 1$, ami kisebb lesz, mert $w(F) > 1$ biztos igaz

note: lehet csinálni egyetlen nagy súlyfüggvényt is 3 helyett, ami bármelyik átírást is csináljuk, csökken
tehát bármilyen sorrendben is végezzük az átírásokat, előbb-utóbb egy CNF lesz a vége

$$(p \vee q) \leftrightarrow (q \rightarrow r)$$

Nyilak eliminálása (súly alapból $2(w(p \vee q) + w(q \rightarrow r)) + 1 = 2((1 + 1) + (1 + 1 + 1)) + 1 = 11$):

$$(\neg(p \vee q) \vee (q \rightarrow r)) \wedge ((p \vee q) \vee \neg(q \rightarrow r)) \quad \text{súly } 1 + 1 + 3 + 1 + 1 + 3 = 10$$

$$(\neg(\underline{p \vee q}) \vee (\neg q \vee r)) \wedge ((p \vee q) \vee \underline{\neg(\neg q \vee r)}) \quad \text{súly } 8$$

Negálások bevitele (negálásos súly ennek $(2 + 2)^2 + 2^2 + 2 + 2 + 2 + (2^2 + 2)^2 = 62$):

$$((\neg p \wedge \neg q) \vee (\neg q \vee r)) \wedge ((p \vee q) \vee (\underline{\neg q} \wedge \neg r)) \quad \text{súly } 2^2 + 2^2 + 2^2 + 2 + 2 + 2 + (2^2)^2 + 2^2 = 38$$

$$((\underline{\neg p} \wedge \underline{\neg q}) \vee (\underline{\neg q} \vee r)) \wedge ((\underline{p \vee q}) \vee (\underline{q} \wedge \underline{\neg r})) \quad \text{súly } 2^2 + 2^2 + 2^2 + 2 + 2 + 2 + 2 + 2^2 = 24$$

Disztributivitás: diszt. súly alapból $(2 + 2 + 1) \cdot (2 \cdot 2) + (2 \cdot 2) \cdot (2 + 2 + 1) + 1 = 41$

$$(\underline{\neg p} \vee \underline{\neg q} \vee r) \wedge (\underline{\neg q} \vee \underline{\neg q} \vee r) \wedge (\underline{p \vee q} \vee \underline{q}) \wedge (\underline{p \vee q} \vee \underline{\neg r})$$

$$\text{súly } 2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 2 + 1 + 2 \cdot 2 \cdot 2 + 2 \cdot 2 \cdot 2 + 1 + 1 = 35$$

a lényeg: mindig csökkennek a súlyok és természetes számok $\Rightarrow$ nem eshet végtelen ciklusba

Formulák által indukált Boole-függvények

Indukált Boole függvény

Ha az F formulában csak a $\{p_1, \dots, p_n\}$ változók szerepelnek, akkor F **indukál** egy n -változós Boole-függvényt, melyet szintén F -fel jelölünk:

$$p_i(x_1, \dots, x_n) := x_i \quad (\text{ezt } \textbf{projekciónak} \text{ hívjuk}) \quad \text{tkp tömbelem-kiválasztás}$$

$$(\neg F)(x_1, \dots, x_n) := \neg(F(x_1, \dots, x_n))$$

$$(F \vee G)(x_1, \dots, x_n) := F(x_1, \dots, x_n) \vee G(x_1, \dots, x_n)$$

...

(ez ismét egy formula felépítése szerinti indukcióval **definiált** fogalom)

az $F = p_1 \vee (\neg p_1 \wedge p_2)$ formulára $F(0, 1) = 1$:

$$p_1(0, 1) = 0$$

$$p_2(0, 1) = 1$$

$$(\neg p_1)(0, 1) = \neg 0 = 1$$

$$(\neg p_1 \wedge p_2)(0, 1) = 1 \wedge 1 = 1$$

$$(p_1 \vee (\neg p_1 \wedge p_2))(0, 1) = 0 \vee 1 = 1.$$

Boole-függvények megszorításai

Legyen f/n Boole-függvény, $n > 0$. Ha $b \in \{0, 1\}$ igazságérték, úgy $f|_{x_n=b}$ jelöli azt az $(n-1)$ -változós Boole-függvényt, melyet úgy kapunk, hogy f inputjában x_n értékét b -re rögzítjük.

Formálisan

$$f|_{x_n=b}(x_1, \dots, x_{n-1}) := f(x_1, \dots, x_{n-1}, b).$$

Példa

- $\vee|_{x_2=1}$ a konstans 1 függvény: $\vee|_{x_2=1}(x_1) = \vee(x_1, 1) = 1$.
- $\wedge|_{x_2=0}$ a konstans 0 függvény
- $\wedge|_{x_2=1}$ az identikus $(x_1 \mapsto x_1)$ függvény

stb.

(Hasonlóan, rögzíthetjük bármelyik koordinátát, nem feltétlenül az utolsót.)

Shannon expanzió

A következőt könnyű látni:

$$f(x_1, \dots, x_n) = (x_n \wedge f|_{x_n=1}(x_1, \dots, x_{n-1})) \vee (\neg x_n \wedge f|_{x_n=0}(x_1, \dots, x_{n-1})).$$

note: ez nem ekvivalencia, hanem egyenlőség, mert nem formula, hanem függvény!

két függvény akkor egyenlő, ha minden inputra megegyezik az eredményük

Hiszen ha $x_n = 1$, akkor a jobb oldali tag hamis lesz, a bal oldali pedig épp $f|_{x_n=1}(x_1, \dots, x_{n-1}) = f(x_1, \dots, x_{n-1}, 1)$, ami megegyezik $f(x_1, \dots, x_n)$ -nel ebben az esetben; az $x_n = 0$ eset hasonló.

Ebből a következőt kapjuk:

Minden $n \geq 1$ -változós Boole-függvény előáll a **projekciók** és a $\{\neg, \vee, \wedge\}$ Boole-függvények alkalmas **kompozíciójaként**.

Ezt úgy is mondjuk, hogy a $\{\neg, \vee, \wedge\}$ **rendszer teljes**.

Shannon expanzió

Minden $n \geq 1$ -változós Boole-függvény előáll a **projekciók** és a $\{\neg, \vee, \wedge\}$ Boole-függvények alkalmas kompozíciójaként.

Bizonyítás

n szerinti teljes indukciót alkalmazunk.

- Ha $n = 1$, akkor az $f/1$ függvény vagy a konstans $0 = x_1 \wedge \neg x_1$, vagy a konstans $1 = x_1 \vee \neg x_1$, vagy az x_1 , vagy a $\neg x_1$, mind a négy előállítható így.
- Ha $n > 1$, akkor az indukciós feltevés szerint az $f|_{x_n=b}(x_1, \dots, x_{n-1})$ Boole-függvények $b \in \{0, 1\}$ -re előállnak ilyen alakban, a Shannon expanzióban pedig szintén csak ezt a három műveletet alkalmazzuk.

Következmény

Minden $n \geq 1$ -változós Boole-függvény indukálható olyan formulával, melyben csak a $\{\neg, \vee, \wedge\}$ konnektívák szerepelnek.

Shannon-expanziós példa

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

$$= (x_3 \wedge f|_{x_3=1}) \vee (\neg x_3 \wedge f|_{x_3=0})$$

x_1	x_2	$f _{x_3=0}$
0	0	0
1	0	1
0	1	1
1	1	0

x_1	x_2	$f _{x_3=1}$
0	0	0
1	0	1
0	1	1
1	1	1

Shannon-expanziós példa

x_1	x_2	$f _{x_3=1}$
0	0	0
1	0	1
0	1	1
1	1	1

$$= (x_2 \wedge f|_{x_3=1, x_2=1}) \vee (\neg x_2 \wedge f|_{x_3=1, x_2=0})$$

x_1	$f _{x_3=1, x_2=0}$
0	0
1	1

x_1	$f _{x_3=1, x_2=1}$
0	1
1	1

$$f|_{x_3=1, x_2=0} = x_1$$

$$f|_{x_3=1, x_2=1} = \uparrow \quad \text{ez csak ideiglenes, egyszerűsíteni akarunk vele}$$

$$f|_{x_3=1} = (x_2 \wedge \uparrow) \vee (\neg x_2 \wedge x_1)$$

$$= x_2 \vee (\neg x_2 \wedge x_1)$$

Shannon-ekspandíós példa

x_1	x_2	$f _{x_3=0}$
0	0	0
1	0	1
0	1	1
1	1	0

$$= (x_2 \wedge f|_{x_3=0, x_2=1}) \vee (\neg x_2 \wedge f|_{x_3=0, x_2=0})$$

x_1	$f _{x_3=0, x_2=0}$
0	0
1	1

x_1	$f _{x_3=0, x_2=1}$
0	1
1	0

$$f|_{x_3=0, x_2=1} = \neg x_1$$

$$f|_{x_3=0, x_2=0} = x_1$$

$$f|_{x_3=0} = (x_2 \wedge \neg x_1) \vee (\neg x_2 \wedge x_1)$$

$$f = (x_3 \wedge (x_2 \vee (\neg x_2 \wedge x_1))) \vee (\neg x_3 \wedge ((x_2 \wedge \neg x_1) \vee (\neg x_2 \wedge x_1)))$$

CNF-ek megszorításai

Ha ℓ egy literál, akkor $\bar{\ell}$ jelöli ℓ **komplementerét**: $\bar{p} := \neg p$ és $\overline{\neg p} := p$.

Hasonlóan a Boole-függvények megszorításaihoz, CNF-ekre is van megszorítás.

Ha Σ klózok egy halmaza és ℓ egy literál, akkor a $\Sigma|_{\ell=1}$ is egy klózhalmoz, mégpedig:

- Σ -ból elhagyjuk az ℓ -t tartalmazó klózokat,
- az eredmény klózaiból pedig elhagyjuk az $\bar{\ell}$ -eket.

Ha $\Sigma = \{\{p\}, \{p, q\}, \{\neg p, \neg q\}, \{p, \neg p, r\}\}$ és $\ell = \neg p$, akkor $\Sigma|_{\neg p=1} = \{\square, \{q\}\}$.

A $\Sigma|_{\ell=0}$ pedig legyen $\Sigma|_{\bar{\ell}=1}$.

note: ezek csak szintaktikai definíciók! csak a CNF-et változtatják, értékadásokhoz nincs by def köztük

CNF-ek megszorításai

Ha a Σ CNF az f Boole-függvényt indukálja, akkor $\Sigma|_{p_i=b}$ az $f|_{x_i=b}$ függvényt.

Legyen $\mathcal{A}$ értékadás, Σ CNF és ℓ literál.

$\mathcal{A} \models \Sigma$ pontosan akkor igaz, ha

- $\mathcal{A}(\ell) = 1$ és $\mathcal{A} \models \Sigma|_{\ell=1}$
- vagy $\mathcal{A}(\ell) = 0$ és $\mathcal{A} \models \Sigma|_{\ell=0}$.

ez pont megfelel a Shannon-expanziós képletnek

Röviden: ha $\mathcal{A} \models \Sigma|_{\ell=\mathcal{A}(\ell)}$.

Mert

Ha $\mathcal{A}(\ell) = 1$, akkor $\mathcal{A}$ kielégíti az összes Σ -beli klózt, melyekben ℓ szerepel.

A többi klózból a $\bar{\ell}$ literál értéke $\mathcal{A}$ mellett hamis, elhagyható belőlük.

A kapott klózhalmaz épp $\Sigma|_{\ell=1}$.

(Az $\ell = 0$ eset is kész ezzel, hiszen az ugyanaz, mint az $\bar{\ell} = 1$ eset.)

Az első algoritmus

Az előzőnek következménye:

Legyen Σ klózalmaz, ℓ literál.

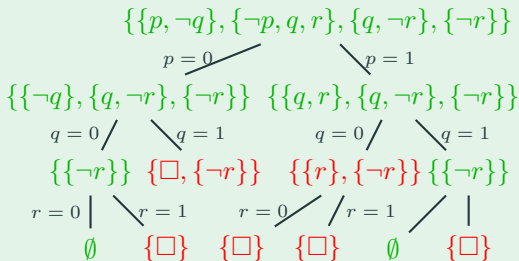
A Σ pontosan akkor kielégíthető, ha a $\Sigma|_{\ell=0}$ vagy a $\Sigma|_{\ell=1}$ klózalmazok valamelyike kielégíthető.

Ez ad egy algoritmust:

```
function A( $\Sigma$ )  
  if  $\square \in \Sigma$  then return false  
  if  $\Sigma = \emptyset$  then return true  
  (válasszunk egy  $p$  változót)  
  return  $A(\Sigma|_{p=0}) \vee A(\Sigma|_{p=1})$ 
```

(Könnyű módosítani, hogy
visszaadjon egy kielégítő értékadást,
ha van.)

Példa



zöld: az a node true-t ad vissza, piros: false

A keresési tér vágása: Unit propagation

Ha van egységklóz a CNF-ben, az kényszeríti a benne lévő változó értékét:

Ha Σ -ban van egy $\{\ell\}$ egységklóz, akkor Σ minden $\mathcal{A}$ modelljében $\mathcal{A}(\ell) = 1$.

Tehát ekkor Σ pontosan akkor kielégíthető, ha $\Sigma|_{\ell=1}$ az (hiszen $\Sigma|_{\ell=0}$ -ban lesz egy üres klóz, így kielégíthetetlen).

Unit propagation

Ha $\{\ell\} \in \Sigma$, akkor elég $\Sigma|_{\ell=1}$ -re hívni rekurzívan.

A keresési tér vágása: Pure literal elimination

Ha ℓ olyan literál, melynek komplementere nem fordul elő Σ -ban, akkor Σ pontosan akkor kielégíthető, ha $\Sigma|_{\ell=1}$ az.

Tehát: ha $\bar{\ell}$ nem szerepel Σ -ban, akkor ℓ -t biztonsággal 1-re állíthatjuk.

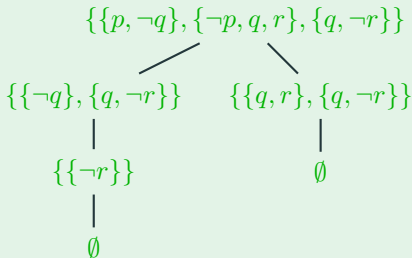
Hiszen ha $\mathcal{A} \models \Sigma$, akkor az a $\mathcal{A}[\ell = 1]$ értékadás, mely ℓ értékét 1-re állítja, az ℓ -ben nem szereplő változókon pedig megegyezik $\mathcal{A}$ -val, kielégíti $\Sigma|_{\ell=1}$ -et:

- ha $\ell \in C$, akkor az $[\ell = 1]$ miatt garantáltan;
- ha $\ell \notin C$, akkor $\mathcal{A}(C) = \mathcal{A}[\ell = 1](C)$, hiszen ekkor C értéke nem függ ℓ értékétől.

```

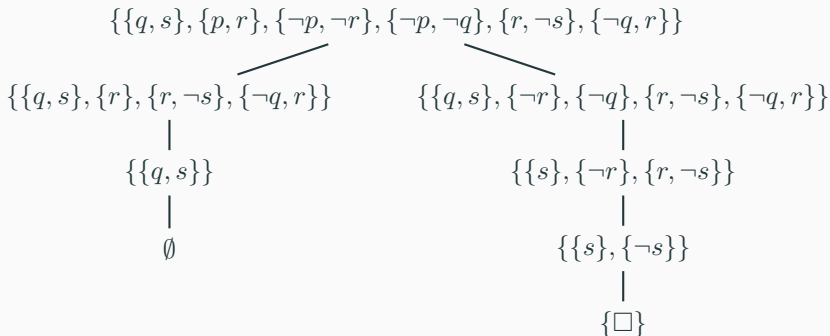
function dpll( $\Sigma$ )
  if  $\Sigma = \emptyset$  then return true
  if  $\square \in \Sigma$  then return false
  if valamilyen  $\ell$ -re  $\{\ell\} \in \Sigma$  then
    return dpll( $\Sigma|_{\ell=1}$ )
  if valamilyen  $\ell$ -re
     $\bar{\ell}$  nem szerepel  $\Sigma$ -ban then
      return dpll( $\Sigma|_{\ell=1}$ )
  (válasszunk egy  $p$  változót)
  return dpll( $\Sigma|_{p=0}$ )  $\vee$  dpll( $\Sigma|_{p=1}$ )
    
```

Példa



A mai leggyorsabb SAT solverok ennek változatait implementálják (később látjuk, milyeneket).

Másik DPLL példa



```

{ {q,s}, {p,r}, {-p,-r}, {-p,-q}, {r,-s}, {-q,r} }
--p=0-- { {q,s}, {r}, {r,-s}, {-q,r} }
--p=0-- --r=1-- { {q,s} }
--p=0-- --r=1-- --q=1-- {}
--p=1-- { {q,s}, {-r}, {-q}, {r,-s}, {-q,r} }
--p=1-- --q=0-- { {s}, {-r}, {r,-s} }
--p=1-- --q=0-- --r=0-- { {s}, {-s} }
--p=1-- --q=0-- --r=0-- --s=1-- { {} }

```

a formula kielégítheto, mert van {} címkéju levél

Rezolúció

A **rezolúciós következtetés**:

$$\{F \vee G, \neg F \vee H\} \models G \vee H.$$

Hiszen ha $\mathcal{A}(F) = 1$, akkor $\mathcal{A}(\neg F \vee H) = 1$ -ből $\mathcal{A}(H) = 1$ és ezért $\mathcal{A}(G \vee H) = 1$; ha pedig $\mathcal{A}(F) = 0$, akkor $\mathcal{A}(F \vee G) = 1$ miatt $\mathcal{A}(G) = 1$ és így $\mathcal{A}(G \vee H) = 1$.

Ennek megfelelően két klóz **rezolvensét** így definiáljuk:

Rezolvens

Ha C és D klózok, $p \in C$ és $\neg p \in D$, akkor C és D (p menti) rezolvense a

$$(C - \{p\}) \cup (D - \{\neg p\})$$

klóz.

Például $\{p, q, s\}$ és $\{\neg p, \neg r, s\}$ rezolvense (p mentén) $\{q, s\} \cup \{\neg r, s\} = \{q, \neg r, s\}$.

Rezolúció

A **rezolúciós algoritmus** a következő:

Input: klózok Σ halmaza.

Output: kielégíthetetlen-e Σ ?

Algoritmus: listát vezetünk klózokról. Egy klózt felvehetünk, ha

- Σ -beli vagy
- két, a listán már szereplő klóz rezolvense.

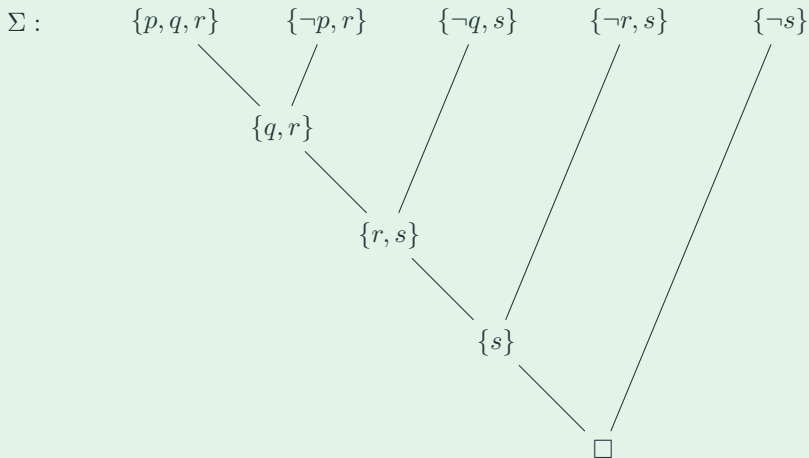
Ha az $\square$ üres klóz rákerül a listára, Σ kielégíthetetlen.

Ha már nem tudunk új klózt felvenni és $\square$ nincs köztük, Σ kielégíthető.

Példa: $\Sigma = \left\{ \{p, q, r\}, \{\neg p, r\}, \{\neg q, s\}, \{\neg r, s\}, \{\neg s\} \right\}$

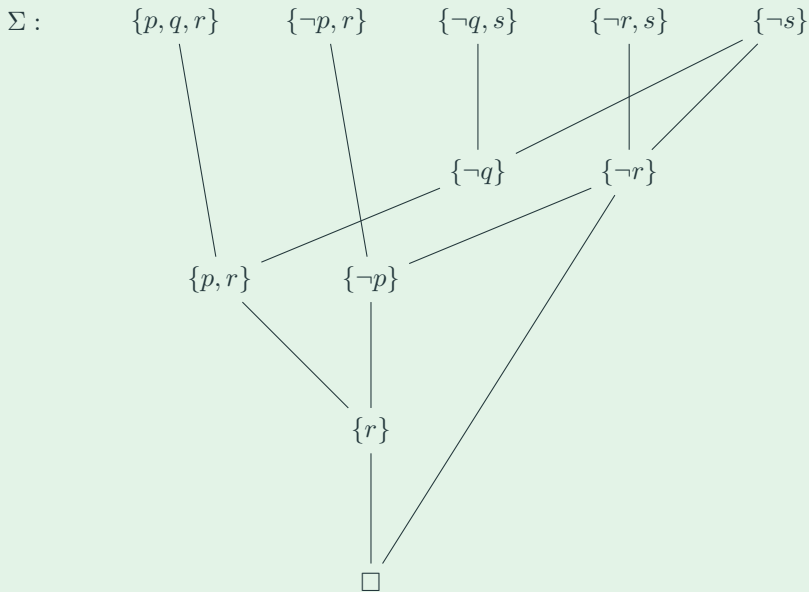
- | | | | | |
|----|----------------------------|----|----------------------------|-----------|
| 1. | $\{p, q, r\} \in \Sigma$ | 5. | $\{r, s\}$ | Res(3, 4) |
| 2. | $\{\neg p, r\} \in \Sigma$ | 6. | $\{\neg r, s\} \in \Sigma$ | |
| 3. | $\{q, r\}$ | 7. | $\{s\}$ | Res(5, 6) |
| 4. | $\{\neg q, s\} \in \Sigma$ | 8. | $\{\neg s\} \in \Sigma$ | |
| | | 9. | $\square$ | Res(7, 8) |

Példa megint



Többször is szabad felhasználni ugyanazt

Példa megint



Helyesség és teljesség

A rezolúciós algoritmus

- **Helyes:** ha az algoritmus „kielégíthetetlen” válasszal áll meg, akkor az input Σ valóban kielégíthetetlen;
- **Teljes:** ha Σ kielégíthetetlen, akkor az algoritmus mindig a „kielégíthetetlen” válasszal áll meg.

DON'T DO THIS

1. $\{p, q\} \in \Sigma$
2. $\{\neg p, \neg q\} \in \Sigma$
3. $\square \quad \text{Res}(1, 2)$

Miért

Ha több literál mentén rezolválunk egyszerre, az **nem helyes** következtetés: úgy is kijöhet az üres klóz, ha a formula kielégíthetetlen.

Mint ahogy a bal oldalon is kielégíthető a $(p \vee q) \wedge (\neg p \vee \neg q)$ formula.

Helyesség

Annyit elég belátnunk, hogy minden klóz, mely a listára kerül, logikai következménye Σ -nak. Ezt indukcióval tesszük: ha a C klóz lista n . elemeként kerül a listára, akkor:

- Ha $C \in \Sigma$, akkor $\Sigma \models C$ mindig teljesül. (Ha $\Gamma \subseteq \Sigma$, akkor $\Sigma \models \Gamma$.)
- Ha C a korábban már felvett C_1 és C_2 klózok rezolvense, akkor
 - indukciós feltevés szerint $\Sigma \models C_1$ és $\Sigma \models C_2$
 - tehát $\Sigma \models \{C_1, C_2\}$ (Ha $\Sigma \models \Gamma_1$ és $\Sigma \models \Gamma_2$, akkor $\Sigma \models \Gamma_1 \cup \Gamma_2$.)
 - a rezolúciós következtetés szerint pedig $\{C_1, C_2\} \models C$
 - így a $\models$ tranzitivitása miatt $\Sigma \models C$. (Ha $\Sigma \models \Gamma$ és $\Gamma \models \Delta$, akkor $\Sigma \models \Delta$.)

Ha pedig $\Sigma \models \square$, akkor Σ valóban kielégíthetetlen, hiszen $\text{Mod}(\square) = \emptyset$ és ekkor $\text{Mod}(\Sigma) \subseteq \text{Mod}(\square)$ miatt $\text{Mod}(\Sigma) = \emptyset$.

(Egy Σ formulahalmaz pontosan akkor kielégíthetetlen, ha $\Sigma \models \downarrow$.)

gyakon

Lineáris rezolúció

A teljességnél többet fogunk megmutatni: azt, hogy ha Σ kielégíthetetlen, akkor $\square$ levezethető

- bármelyik, a „kielégíthetetlenségért felelős” klózból indulva
- úgy, hogy minden további lépésben a listára utoljára felvett klózt rezolváljuk egy, vagy a listán, vagy Σ -ban szereplő klózzal.

Ezt egy kicsit precízebben:

A Σ kielégíthetetlen klózalmaznak a $\Sigma' \subseteq \Sigma$ egy **minimális kielégíthetetlen** részalmaz, ha Σ' is kielégíthetetlen, de Σ' bármelyik valódi részalmaz már kielégíthető.

Ha $\Sigma = \{ \{p, q\}, \{\neg p, q\}, \{\neg q\}, \{p, r\}, \{\neg p, \neg r\}, \{s\}, \{\neg s\} \}$, akkor pl.

$\{ \{p, q\}, \{\neg p, q\}, \{\neg q\} \}$ egy minimális kielégíthetetlen részalmaz, $\{ \{s\}, \{\neg s\} \}$ pedig egy másik. A $\{p, r\}$ és $\{\neg p, \neg r\}$ klózek nem elemei egyetlen minimális kielégíthetetlen részalmaznak sem.

Lineáris rezolúció

A lineáris rezolúciós algoritmus pedig a következő:

Input: Σ klózhalmoz.

Output: kielégíthetetlen-e Σ ?

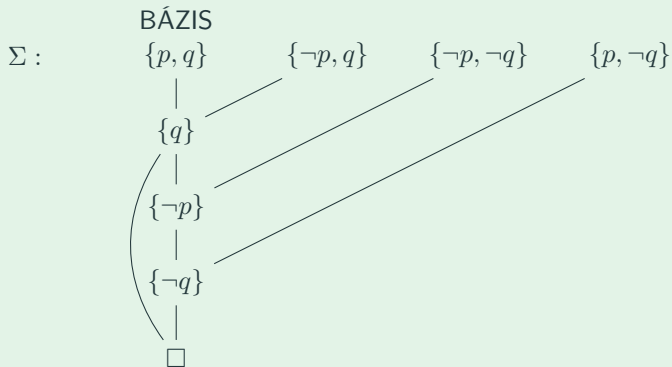
Algoritmus: Listát vezetünk klózokról.

- Az első lépésben felvehetjük Σ bármelyik elemét, ez lesz a levezetés **bázisa**.
- Minden további lépésben felvehetjük az előző lépésben felvett klóznak és egy vagy már a listán szereplő, vagy Σ -beli klóznak a rezolvensét. Ezt a másik klózt hívjuk ennek a lépésnek az **oldaklózának**.

Ha pl. $\Sigma = \{\{p, q\}, \{\neg p, q\}, \{\neg p, \neg q\}, \{p, \neg q\}\}$:

- | | | | | | |
|----|-----------------------|-------------------------------|--------------|--------------------------|-----------|
| 1. | $\{p, q\} \in \Sigma$ | 4. | $\{\neg q\}$ | Res(3, $\{p, \neg q\}$) | |
| 2. | $\{q\}$ | Res(1, $\{\neg p, q\}$) | 5. | $\square$ | Res(4, 2) |
| 3. | $\{\neg p\}$ | Res(2, $\{\neg p, \neg q\}$) | | | |

Lineáris rezolúció rajzban



Nyilván igaz, hogy ami lineáris rezolúcióval levezethető Σ -ból, az rezolúcióval is levezethető, tehát a lineáris rezolúciós algoritmus is helyes.

Teljes is:

Ha Σ kielégíthetetlen és $C \in \Sigma$ benne van a Σ egy minimális kielégíthetetlen részhalmazában, akkor Σ -ból levezethető az üres klóz olyan lineáris rezolúciós levezetéssel, melynek bázisa C .

Tehát az állítás az, hogy ha Σ kielégíthetetlen, akkor lineáris rezolúcióval is le lehet vezetni az üres klózt, bárből kiindulva, aki a kielégíthetlenségért „felelős”.

a terv: szétvágjuk egy p változó mentén $\Sigma|_{p=1}$ -re és $\Sigma|_{p=0}$ -ra, mindkettő $\models \downarrow$,
 levezetjük mindkettőben az üres klózt, „visszavetítjük” ezt a két levezetést Σ -ra és
 „összeragasztjuk” őket

Ha

$$\Sigma = \{\{p, \neg q, r\}, \{p, q\}, \{\neg p, r\}, \{\neg q, \neg r\}, \{q, \neg r\}\}$$

és mondjuk a p változót választjuk, akkor

$$\Sigma|_{p=1} = \{\{r\}, \{\neg q, \neg r\}, \{q, \neg r\}\} \quad \Sigma|_{p=0} = \{\{\neg q, r\}, \{q\}, \{\neg q, \neg r\}, \{q, \neg r\}\}$$

Lineáris levezetések:

- | | | | | | | | |
|----|--------------|---------------------------------|----|----------------------|---------------------------------|----|--|
| 1. | $\{r\}$ | $\in \Sigma _{p=1}$ | 1. | $\{\neg p, r\}$ | $\in \Sigma$ | | |
| 2. | $\{\neg q\}$ | $\leftarrow \{\neg q, \neg r\}$ | 2. | $\{\neg p, \neg q\}$ | $\leftarrow \{\neg q, \neg r\}$ | | |
| 3. | $\{\neg r\}$ | $\leftarrow \{q, \neg r\}$ | 3. | $\{\neg p, \neg r\}$ | $\leftarrow \{q, \neg r\}$ | | |
| 4. | $\square$ | $\leftarrow \{r\}$ | 4. | $\{\neg p\}$ | $\leftarrow \{\neg p, r\}$ | | |
| | | | 5. | $\{q\}$ | $\leftarrow \{p, q\}$ | 1. | $\{q\}$ $\in \Sigma _{p=0}$ |
| | | | 6. | $\{\neg r\}$ | $\leftarrow \{\neg q, \neg r\}$ | 2. | $\{\neg r\}$ $\leftarrow \{\neg q, \neg r\}$ |
| | | | 7. | $\{p, \neg q\}$ | $\leftarrow \{p, \neg q, r\}$ | 3. | $\{\neg q\}$ $\leftarrow \{\neg q, r\}$ |
| | | | 8. | $\{p\}$ | $\leftarrow \{q\}$ | 4. | $\square$ $\leftarrow \{q\}$ |
| | | | 9. | $\square$ | $\leftarrow \{\neg p\}$ | | |

Az állítást a Σ -beli változók n száma szerinti indukcióval látjuk be.

- Ha $n = 0$, azaz Σ -ban nincs változó, akkor vagy $\Sigma = \{\}$, vagy $\Sigma = \{\square\}$.
 - A kettő közül $\Sigma = \{\square\}$ a kielégíthetetlen.
 - Ennek $\square$ az egyetlen eleme, ez egy minimális kielégíthetetlen részhalmazának is eleme.
 - Ha felvesszük bázisként, már le is vezettük az üres klózt.
- Ha $n > 0$, akkor vegyünk egy C klózt, mely szerepel Σ egy minimális kielégíthetetlen részhalmazában. Legyen ez a részhalmaz Σ' .
 - Ha $C = \square$, kész vagyunk: vegyük fel bázisnak.
 - Különbön legyen $\ell \in C$ egy C -beli literál.
 - Vegyük észre: minimális kielégíthetetlen részhalmazban nincs pure literál, hiszen ha ℓ pure lenne, akkor a Σ' -nak egy valódi részhalmaza, $\Sigma'|_{\ell=1}$ is kielégíthetetlen lenne. Tehát Σ' -ben $\bar{\ell}$ is szerepel valahol.

(nincs vége, csak betelt a fólia)

- Vegyük a $\Sigma'|_{\ell=0}$ és $\Sigma'|_{\ell=1}$ klózalmazokat.
- Mivel Σ' kielégíthetetlen, ezek is azok.
- Bennük csak legfeljebb $n - 1$ változó szerepel (mert ℓ változója kiesik), így alkalmazhatjuk az indukciós feltevést.
- A $\Sigma'|_{\ell=0}$ klózalmaznak $C - \{\ell\}$ is eleme, sőt egy minimális kielégíthetetlen részalmazának is eleme (mert különben $\Sigma' - \{C\}$ is kielégíthetetlen lenne).
- Tehát $\Sigma'|_{\ell=0}$ -ból az indukciós feltevés szerint van $\square$ -nak egy $C_1, C_2, \dots, C_m$ lineáris rezolúciós levezetése, melynek $C_1 = C - \{\ell\}$ a bázisa.

(még mindig nincs vége)

Lineáris rezolúció

- Az előző példa szerint „visszaemelve” a $\Sigma|_{\ell=0}$ cáfolatot Σ' fölötti levezetéssé láthatjuk, hogy az új levezetésben minden klózba bekerül az ℓ literál.
- Ez igaz a bázisra, és minden lépésben az eredeti C_1 és C_2 klózik rezolvense helyett a $C_1 \cup \{\ell\}$ és C_2 vagy $C_2 \cup \{\ell\}$ klózik rezolvensét kapjuk, ami a rezolvens, plusz ℓ .
- Tehát ennek a konstrukciónak a végén az $\{\ell\}$ egységklóznál jár a lineáris rezolúciós levezetés.
- Mivel Σ' minimális kielégíthetetlen, kell legyen benne olyan C klóz is, mely $\bar{\ell}$ -t tartalmazza. (Itt használjuk, hogy Σ' -ben ℓ nem lehetett pure.)
- Akkor $\Sigma'|_{\ell=1}$ -nek egy minimális kielégíthetetlen részhalmazában szerepel $C - \{\bar{\ell}\}$.
- Ebből a klózból indulva az indukciós feltevés szerint van $\Sigma'|_{\ell=1}$ -nek lineáris rezolúciós cáfolata. A terv az, hogy ezt a második cáfolatot „mögéragszjuk” az eddiginek.

már majdnem kész a teljességi bizonyítás

- Az előző fázisban kapott $\{\ell\}$ egységklózt tudjuk rezolválni ezzel a C klózzal, tehát a $\Sigma'|_{\ell=1}$ cáfolatát „fel tudjuk emelni” Σ' fölötti levezetéssé.
- A felemelt levezetés végén vagy $\square$ -t, vagy $\{\bar{\ell}\}$ -t kapunk. Utóbbi esetben még egyszer rezolválunk $\{\ell\}$ -lel mint oldalklózzal és kész vagyunk.

done! easy. a lineáris rezolúció teljes.

Horn-formulák

A rezolúciós módszer helyes és teljes, de vannak olyan formulák, melyeknek a **legrövidebb** rezolúciós cáfolatuk is **exponenciális sok** lépést igényel.

Felmerül a kérdés, tudunk-e mondani olyan formulacsaládot, melyre a módszer garantáltan **gyors**?

Egy klóz **Horn-klóz**, ha benne **legfeljebb egy pozitív** literál szerepel.

Egy CNF **Horn-formula**, ha benne minden klóz Horn-klóz.

Horn-formula pl a szokásos halmazos reprezentációban

$$\left\{ \{p, \neg q\}, \{q\}, \{\neg q, r\}, \{\neg p, \neg r, s\}, \{\neg p, \neg s\} \right\}$$

Honnan kaphatunk épp ilyet? A logikai programok „utasításai” **Definit klóz**, avagy program klóz: egy pozitív literál van benne és „lekérdezései” **Kérdés klóz**, avagy negatív klóz: minden benne levő literál negatív épp ilyenek: (erre még később visszatérünk)

$$q \rightarrow p, \quad q, \quad q \rightarrow r, \quad (p \wedge r) \rightarrow s; \quad :- (p \wedge s)? \quad \text{ez egy prolog program}$$

Horn-formulák

Egy Horn-formula pontosan akkor kielégíthetetlen, ha belőle levezethető az üres klóz úgy, hogy **minden** rezolvensképzésben az egyik résztvevő klóz **pozitív egységklóz**.

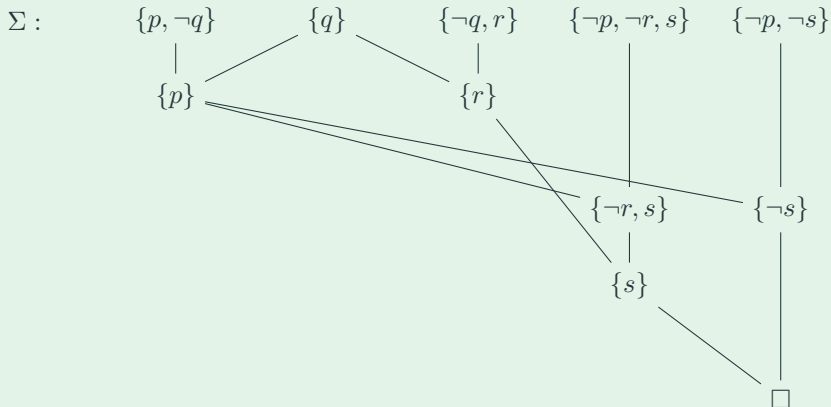
$$\left\{ \{p, \neg q\}, \{q\}, \{\neg q, r\}, \{\neg p, \neg r, s\}, \{\neg p, \neg s\} \right\}$$

- | | | | | | |
|----|-------------------------|--------------------|-----|----------------------|---------------------|
| 1. | $\{p, \neg q\}$ | $\in \Sigma$ | 7. | $\{\neg r, s\}$ | Res(3 , 6) |
| 2. | $\{q\}$ | $\in \Sigma$ | 8. | $\{s\}$ | Res(5 , 7) |
| 3. | $\{p\}$ | Res(1, 2) | 9. | $\{\neg p, \neg s\}$ | $\in \Sigma$ |
| 4. | $\{\neg q, r\}$ | $\in \Sigma$ | 10. | $\{\neg s\}$ | Res(3 , 9) |
| 5. | $\{r\}$ | Res(2 , 4) | 11. | $\square$ | Res(8 , 10) |
| 6. | $\{\neg p, \neg r, s\}$ | $\in \Sigma$ | | | |

Erre tudunk implementálni lineáris időben futó algoritmust.

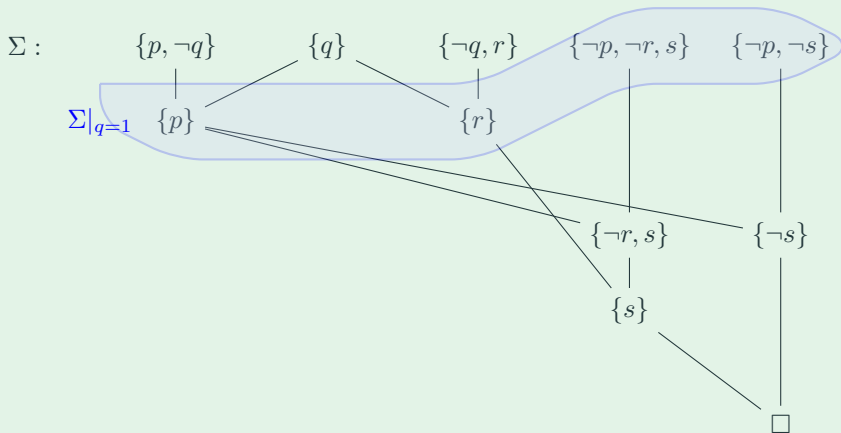
Egy gyors algoritmus Horn-formulákra

választunk egy pozitív egységklózt és végigrezolválunk vele akit csak lehet



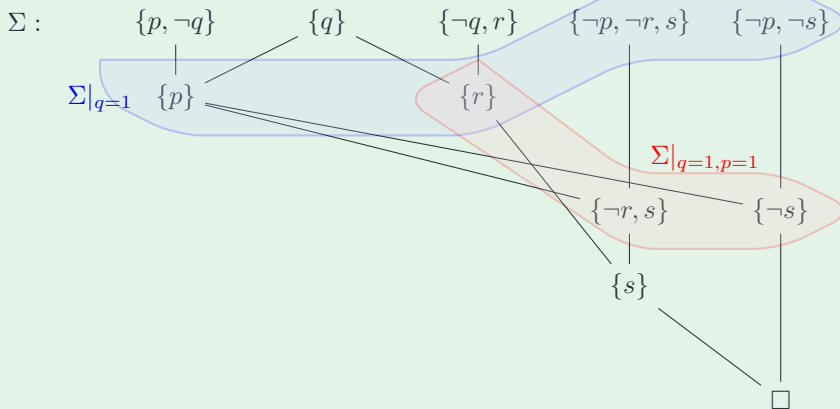
Egy gyors algoritmus Horn-formulákra

választunk egy pozitív egységklózt és végigrezolválunk vele akit csak lehet



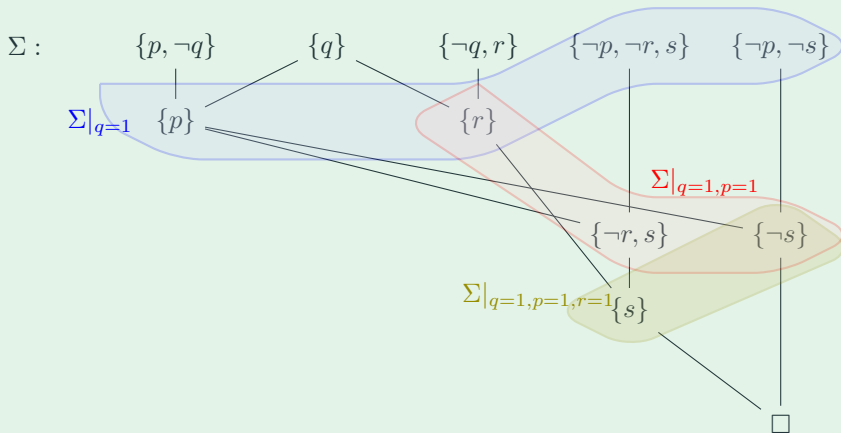
Egy gyors algoritmus Horn-formulákra

választunk egy pozitív egységklózt és végigrezolválunk vele akit csak lehet



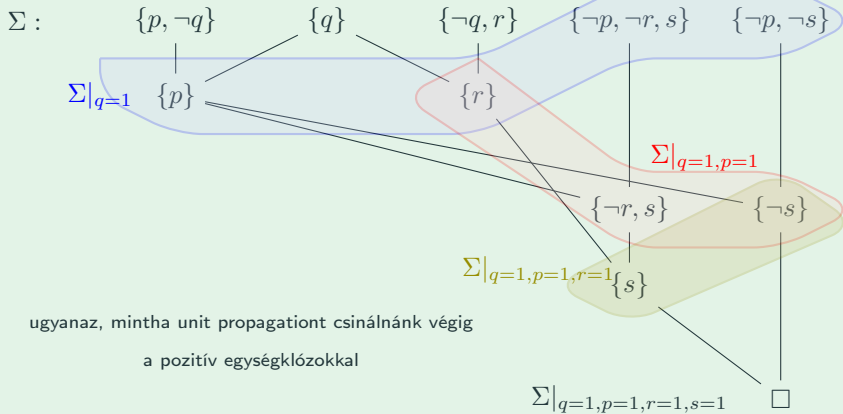
Egy gyors algoritmus Horn-formulákra

választunk egy pozitív egységklózt és végigrezolválunk vele akit csak lehet



Egy gyors algoritmus Horn-formulákra

választunk egy pozitív egységklózt és végigrezolválunk vele akit csak lehet



Horn-formulák

Tegyük fel, hogy Σ kielégíthetetlen Horn-formula. Megmutatjuk, hogy $\square$ levezethető belőle így.

- A Σ -beli változók n száma szerinti indukciót alkalmazunk.
- Ha $n = 0$, akkor Σ -ban nincs változó és kielégíthetetlen $\Rightarrow \Sigma = \{\square\}$ és egy lépésben kész vagyunk.
- Ha $n > 0$ és $\square \in \Sigma$, akkor megint csak egy lépésben kész vagyunk.
- Ha $n > 0$ és $\square \notin \Sigma$, akkor Σ -ban **kell legyen** egy $\{p\}$ pozitív egységklóz: ha nincs, akkor az az értékadás, mely minden változót 0-ra állít, kielégíti Σ -t.

(Hiszen ekkor minden klózban van legalább egy negatív literál.)

- Ekkor $\{p\}$ -vel rezolválva minden Σ -beli klózt, melyben $\neg p$ szerepel, kapjuk, hogy $\Sigma|_{p=1}$ minden klóza levezethető ily módon:
- amik egy $\neg p$ literált tartalmaznak, azok rezolválással $\{p\}$ -vel,
- amik nem, azok elemei Σ -nak is
- $\Sigma|_{p=1}$ is Horn-formula

mert úgy kapjuk, hogy a Σ Horn-formula egyes klózaiból elhagyunk egy negatív literált

- Az indukciós feltevés szerint tehát $\Sigma|_{p=1}$ -ből levezethető így $\square$.

SLD rezolúció

Selective Linear Definite

Egy rezolúciós levezetést **input rezolúciónak** nevezünk, ha minden rezolúciós lépésben a legutóbbi lépésben kapott klózt egy Σ -belivel rezolválunk. Ha Σ Horn-formula, és **negatív** bázisból indulunk, akkor **SLD rezolúciónak** nevezzük ezt.

(Tehát olyan **lineáris** rezolúció, ahol nem rezolválunk a listán korábban szereplő kló-zokkal).

$$\left\{ \{p, \neg q\}, \{q\}, \{\neg q, r\}, \{\neg p, \neg r, s\}, \{\neg p, \neg s\} \right\}$$

- | | |
|--|--|
| 1. $\{\neg p, \neg s\} \in \Sigma$ | 5. $\{\neg q, \neg r\} \leftarrow \{p, \neg q\}$ |
| 2. $\{\neg q, \neg s\} \leftarrow \{p, \neg q\}$ | 6. $\{\neg q\} \leftarrow \{\neg q, r\}$ |
| 3. $\{\neg s\} \leftarrow \{q\}$ | 7. $\square \leftarrow \{q\}$ |
| 4. $\{\neg p, \neg r\} \leftarrow \{\neg p, \neg r, s\}$ | |

Legyen Σ Horn-formula.

- Tudjuk, hogy a lineáris rezolúció teljes.
- Sőt: Σ bármelyik olyan klózából indulhatunk bázisként, mely szerepel egy minimális kielégíthetetlen részhalmazban.
- Ha Σ kielégíthetetlen, akkor van benne olyan negatív klóz, mely szerepel egy minimális kielégíthetetlen részhalmazban.
hiszen ha csak a definit klózokat vesszük, azoknak a halmaza kielégíthető $\Rightarrow$ kell negatív klóz is
- Tehát indulhatunk egy negatív bázisból.
- Az első lépésben csak input rezolúciót tudunk végezni $\Rightarrow$ megint negatív klózt kapunk.
- Tehát minden lépés után igaz, hogy csak negatív klózok szerepelnek a listán.
- Tehát nincs is más választásunk, mint input rezolválni egy lineáris rezolúciós levezetésben $\Rightarrow$ SLD rezolúciót végzünk.
- A $\Box$ viszont garantáltan kijön lineáris rezolúcióval, ha Σ kielégíthetetlen.

Tehát: ha Σ Horn-formula, akkor SLD rezolúcióval is megkaphatjuk $\Box$ -t.

SLD rezolúció

- Az SLD rezolúció Horn-formulákra még akkor is teljes marad, ha a literálokat LIFO adatszerkezetben (veremben) tároljuk az aktuális munkaklózunkban.

Tehát mindig az utolsóként bekerült literált resolváljuk tovább. Ettől „selective”.

- Viszont backtrack néha szükséges lehet:

$$\{\{\neg p, \neg q\}, \{p, \neg r\}, \{p, \neg s\}, \{q\}, \{s\}\}$$

$$1. \quad \{\neg p, \neg q\} \in \Sigma$$

$$2. \quad \{\neg r, \neg q\} \leftarrow \{p, \neg r\}$$

$$3. \quad \text{stuck}$$

$$1. \quad \{\neg p, \neg q\} \in \Sigma$$

$$2. \quad \{\neg s, \neg q\} \leftarrow \{p, \neg s\}$$

$$3. \quad \{\neg q\} \leftarrow \{s\}$$

$$4. \quad \square \leftarrow \{q\}$$

Lehet egy stratégia, hogy mindig a legutolsóként bekerült literált resolváljuk, a szóba jövő klózok közül pedig az elsőt, majd backtrack esetén a másodikat,... próbáljuk ki.

Ebből még így lehet végtelen ciklus.

Amit így kapunk, a Prolog. Elsőrendű logikára. Még visszatérünk rá.

Egy CNF-et 2CNF-nek hívunk, ha benne minden klóz **legfeljebb kételemű**.

Legfeljebb kételemű klózok rezolvense is legfeljebb kételemű.

$$\frac{\{\ell_1, p\}, \{\ell_2, \neg p\}}{\{\ell_1, \ell_2\}}$$

Tehát egy 2CNF-en indított rezolúció során **minden**, a listára felkerülő klóz legfeljebb kételemű lesz.

Ebből pedig csak $O(n^2)$ sok van, ha n a változók száma.

$$f(n) = O(g(n)), \text{ ha } \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} < \infty$$

(A 2SATra is implementálható lineáris időigényű algoritmus.)

ezt majd bonyán

Conflict-driven clause learning, CDCL

A mai vezető SAT solverek több módszert kombinálnak:

- Futtatják a DPLL algoritmust
- Minden új értékadásnál a branching változó és az általa indukált unit propagation alapján felépítenek egy „implikációs gráfot”
- Ellentmondás esetén ez alapján a gráf alapján **tanulnak** egy új klózt
- Ezzel az új klózzal tovább szűkül a bejárandó keresési tér
- Ha az aktuális halmaz speciális alakú, arra célalgoritmust futtatnak

Az előző részben a bizonyításokban felhasználtunk néhány állítást (amiknek a bizonyítása, vagy annak vázlata legalább szóban elhangzott), például:

- Ha $\Sigma \subseteq \Delta$, akkor $\text{Mod}(\Delta) \subseteq \text{Mod}(\Sigma)$. (Tehát akkor $\Delta \models \Sigma$.)
- $\text{Mod}(\Sigma \cup \Delta) = \text{Mod}(\Sigma) \cap \text{Mod}(\Delta)$.
- Ha $\Sigma \models \Delta$ és $\Delta \models \Gamma$, akkor $\Sigma \models \Gamma$.
- Ha $\Sigma \models \Delta$ és $\Sigma \models \Gamma$, akkor $\Sigma \models \Delta \cup \Gamma$.

Most ezeket az állításokat (és még néhány további) bebizonyítjuk **általánosabban**.

Általánosítás

„Írj egy Int lista osztályt!”

```
trait IntList
case object EmptyIntList extends IntList
case class NonemptyIntList(head: Int, tail: IntList) extends IntList
```

„Írj egy Double lista osztályt!”

```
trait DoubleList
case object EmptyDoubleList extends DoubleList
case class NonemptyDoubleList(head: Double, tail: DoubleList) extends DoubleList
```

„függvényt, ami összeadja/szorozza egy Int/Double lista elemeit!”

<pre>def sum(list: IntList): Int = list match { case EmptyIntList => 0 case NonemptyIntList(head, tail) => head + sum(tail) }</pre>	<pre>def sum(list: DoubleList): Double = list match { case EmptyDoubleList => 1 case NonemptyDoubleList(head, tail) => head * sum(tail) }</pre>
---	---

Általánosítás

Inkább:

```
trait MyList[+T] //generikus típusparaméter, kovariáns
object EmptyList extends MyList[Nothing] //az üres lista az üres lista, minden típusnak jó
case class NonemptyList[T](head: T, tail: MyList[T]) extends MyList[T] {
  override def equals(that: Any): Boolean = that match {
    case NonemptyList(h, t) => h == head && t == tail
    case _ => false
  }
}

//...
type IntList = MyList[Int]           //ha nagyon el kell nevezni
type DoubleList = MyList[Double]     //ha nagyon el kell nevezni

def fold[T](list: MyList[T], op: (T,T)=>T, base: T): T = list match {
  case EmptyList => base
  case NonemptyList(head, tail) => op(head, fold(tail, op, base))
}

//ha pl list egy MyList[Int] és összegezni kell:
println(fold[Int](list, { _+_ }, 0)) //nem cica
//ha double lista és szorozni:
println(fold[Double](list, { _*_ }, 1)) //member functionként much jobb, de ok
```

Általánosítás

A matematikai állításokban is az általánosítással a **modularitást** támogatjuk: ha kiemelünk néhány tulajdonságot általánosabb szintre és ezen a szinten bizonyítunk be valamit, az más „hasznló” helyzetben is alkalmazható lehet.

Nézzük a $\text{Mod}(\Sigma)$ definícióját:

$$\text{Mod}(\Sigma) := \{\mathcal{A} : \forall F \in \Sigma \mathcal{A} \models F\}.$$

Ebből amit kiemelhetünk:

- vannak **formulák** (ez egy „típus”)
- vannak **értékadások** (ez egy „másik típus”)
- formulák és értékadások közt van egy $\models$ **reláció**
- ezt a relációt arra használjuk, hogy formula**halmazok** és értékadás**halmazok** közt definiáljunk kapcsolatot
- és erről a kapcsolatról szeretnénk mondani dolgokat.

Galois-kapcsolat

Általánosítsuk az előzőt.

Legyen A és B két **halmaz**, R pedig egy **reláció** köztük ($R \subseteq A \times B$).

Akkor R meghatároz egy $f : P(A) \rightarrow P(B)$ és egy $g : P(B) \rightarrow P(A)$ leképezést a következőképpen:

- $f(X) := \{y \in B : \forall x \in X \ xRy\}$
- $g(Y) := \{x \in A : \forall y \in Y \ xRy\}$

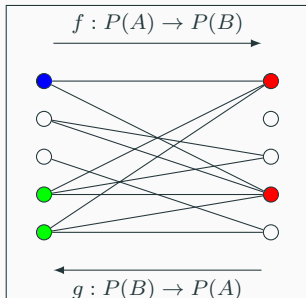
egy (azonos típusú elemeket tartalmazó)

halmaz képe: a másik típusból azok

halmaza, akik a halmaz minden elemével

össze vannak kötve

Azt mondjuk ekkor, hogy f és g **Galois-kapcsolatot** alkotnak.



$f(\text{zöldek}) = \text{pirosak}$

$g(\text{pirosak}) = \text{zöldek} + \text{kék}$

$f(\text{zöldek} + \text{kék}) = \text{pirosak}$

Az előbb: A – értékadások, B – formulák

$R \models, g : \text{Mod}$

Galois-kapcsolat

Általánosítsuk az előzőt.

Legyen A és B két **halmaz**, R pedig egy **reláció** köztük ($R \subseteq A \times B$).

Akkor R meghatároz egy $f : P(A) \rightarrow P(B)$ és egy $g : P(B) \rightarrow P(A)$ leképezést a következőképpen:

- $f(X) := \{y \in B : \forall x \in X \ xRy\}$
- $g(Y) := \{x \in A : \forall y \in Y \ xRy\}$

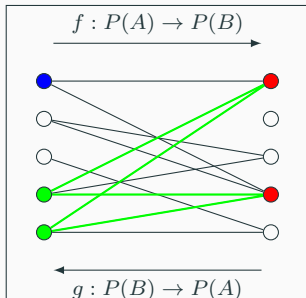
egy (azonos típusú elemeket tartalmazó)

halmaz képe: a másik típusból azok

halmaza, akik a halmaz minden elemével

össze vannak kötve

Azt mondjuk ekkor, hogy f és g **Galois-kapcsolatot** alkotnak.



$f(\text{zöldek}) = \text{pirosak}$

$g(\text{pirosak}) = \text{zöldek} + \text{kék}$

$f(\text{zöldek} + \text{kék}) = \text{pirosak}$

Az előbb: A – értékadások, B – formulák

$R \models, g : \text{Mod}$

Galois-kapcsolat

Általánosítsuk az előzőt.

Legyen A és B két **halmaz**, R pedig egy **reláció** köztük ($R \subseteq A \times B$).

Akkor R meghatároz egy $f : P(A) \rightarrow P(B)$ és egy $g : P(B) \rightarrow P(A)$ leképezést a következőképpen:

- $f(X) := \{y \in B : \forall x \in X \ xRy\}$
- $g(Y) := \{x \in A : \forall y \in Y \ xRy\}$

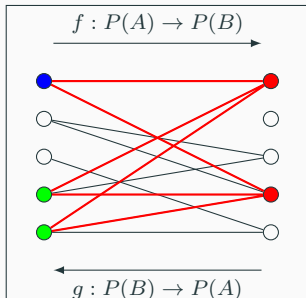
egy (azonos típusú elemeket tartalmazó)

halmaz képe: a másik típusból azok

halmaza, akik a halmaz minden elemével

össze vannak kötve

Azt mondjuk ekkor, hogy f és g **Galois-kapcsolatot** alkotnak.



$f(\text{zöldek}) = \text{pirosak}$

$g(\text{pirosak}) = \text{zöldek} + \text{kék}$

$f(\text{zöldek} + \text{kék}) = \text{pirosak}$

Az előbb: A – értékadások, B – formulák

$R \models, g : \text{Mod}$

Galois-kapcsolat

Általánosítsuk az előzőt.

Legyen A és B két **halmaz**, R pedig egy **reláció** köztük ($R \subseteq A \times B$).

Akkor R meghatároz egy $f : P(A) \rightarrow P(B)$ és egy $g : P(B) \rightarrow P(A)$ leképezést a következőképpen:

- $f(X) := \{y \in B : \forall x \in X \ xRy\}$
- $g(Y) := \{x \in A : \forall y \in Y \ xRy\}$

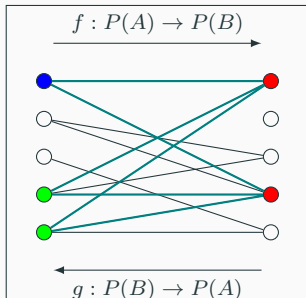
egy (azonos típusú elemeket tartalmazó)

halmaz képe: a másik típusból azok

halmaza, akik a halmaz minden elemével

össze vannak kötve

Azt mondjuk ekkor, hogy f és g **Galois-kapcsolatot** alkotnak.



$f(\text{zöldek}) = \text{pirosak}$

$g(\text{pirosak}) = \text{zöldek} + \text{kék}$

$f(\text{zöldek} + \text{kék}) = \text{pirosak}$

Az előbb: A – értékadások, B – formulák

$R \models, g : \text{Mod}$

A mi esetünkben mi most akkor f ?

- Értékadásoknak egy $\mathcal{K}$ halmazához rendeli formulák egy Σ halmazát
- Az F formula akkor kerül bele ebbe a halmazba, ha $\mathcal{A} \models F$ teljesül minden $\mathcal{A} \in \mathcal{K}$ -ra

Ezt az operátort el is nevezzük: $\mathcal{K}$ elmélete („theory”), jele $\text{Th}(\mathcal{K})$ lesz:

Ha $\mathcal{K}$ értékadások egy halmaza, akkor

$$\text{Th}(\mathcal{K}) := \{F : \forall \mathcal{A} \in \mathcal{K} \mathcal{A} \models F\}$$

a $\mathcal{K}$ elmélete.

(tehát: az összes olyan formula, melyeket $\mathcal{K}$ minden egyes eleme kielégít).

Eddig tehát azt tudjuk, hogy Mod és Th Galois-kapcsolatban vannak.

Példa

ha $\mathcal{A} : p \mapsto 1, q \mapsto 0, r \mapsto 1$ és $F = (p \rightarrow q) \vee (q \rightarrow r)$, akkor

- $\mathcal{A} \in \text{Mod}(F)$
- $F \in \text{Th}(\mathcal{A})$

ha $\mathcal{A}_2 : p \mapsto 0, q \mapsto 1, r \mapsto 0$, akkor

- $\mathcal{A}_2 \in \text{Mod}(F)$ is és
- $F \in \text{Th}(\mathcal{A}_2)$, ezért
- $F \in \text{Th}(\{\mathcal{A}, \mathcal{A}_2\})$.

note: olyan F és $\mathcal{A}$ nincs, melyekre $\mathcal{A} \in \text{Mod}(\{F, \neg F\})$

a $\neg$ egy olyan művelet lesz, amivel nem tudunk mit kezdeni, csak a $\models$ -t fogjuk nézni mint relációt

Galois-kapcsolat

Legyenek A és B halmazok, $f : P(A) \rightarrow P(B)$ és $g : P(B) \rightarrow P(A)$ pedig az $R \subseteq A \times B$ reláció által meghatározott Galois-kapcsolat függvényei.

Ha $X_1 \subseteq X_2 \subseteq A$, akkor $f(X_2) \subseteq f(X_1)$. „ f **antimonoton**” (antitone).

- Legyen $y \in f(X_2)$. azt akarjuk belátni, hogy akkor $y \in f(X_1)$ is
- Akkor f definíciója szerint $x_2 R y$ minden $x_2 \in X_2$ -re. Galois def
- Mivel $X_1 \subseteq X_2$, ezért tehát $x_1 R y$ minden $x_1 \in X_1$ -re is.
$$\forall x(x \in X_1 \rightarrow x \in X_2) \wedge \forall x(x \in X_2 \rightarrow x R y) \models \forall x(x \in X_1 \rightarrow x R y)$$
- Tehát ekkor $y \in f(X_1)$ is igaz $\Rightarrow f(X_2) \subseteq f(X_1)$. $\subseteq$ def

Persze ugyanígy g is antimonoton.

Most tehát megkaptuk azt is, hogy **ha** $\Sigma \subseteq \Delta$, **akkor** $\text{Mod}(\Delta) \subseteq \text{Mod}(\Sigma)$, vagyis $\Delta \models \Sigma$, hiszen Mod egy Galois-kapcsolatban az egyik függvény.

Továbbá, azt is megkaptuk, hogy ha $\mathcal{K}_1 \subseteq \mathcal{K}_2$ értékadások halmazai, akkor $\text{Th}(\mathcal{K}_2) \subseteq_{90} \text{Th}(\mathcal{K}_1)$.

Galois-kapcsolat

Tetszőleges $X_1, X_2 \subseteq A$ -ra $f(X_1 \cup X_2) = f(X_1) \cap f(X_2)$.

Bizonyítás

$$y \in f(X_1 \cup X_2)$$

$$\Leftrightarrow xRy \text{ minden } x \in X_1 \cup X_2\text{-re}$$

Galois def

$$\Leftrightarrow xRy \text{ minden } x \in X_1\text{-re és } xRy \text{ minden } x \in X_2\text{-re}$$

unió

$$\Leftrightarrow y \in f(X_1) \text{ és } y \in f(X_2)$$

Galois def

$$\Leftrightarrow y \in f(X_1) \cap f(X_2).$$

metszet def

$$\begin{aligned} \text{note unió: } \forall x((x \in X_1 \vee x \in X_2) \rightarrow xRy) &\equiv \forall x((x \in X_1 \rightarrow xRy) \wedge (x \in X_2 \rightarrow xRy)) \\ &\equiv \forall x(x \in X_1 \rightarrow xRy) \wedge \forall x(x \in X_2 \rightarrow xRy) \end{aligned}$$

Ugyanígy, $g(Y_1 \cup Y_2) = g(Y_1) \cap g(Y_2)$.

Ezzel megkaptuk, hogy $\text{Mod}(\Sigma \cup \Delta) = \text{Mod}(\Sigma) \cap \text{Mod}(\Delta)$.

$$X \subseteq g(Y) \Leftrightarrow Y \subseteq f(X).$$

Bizonyítás

$X \subseteq g(Y) \Leftrightarrow$ minden $x \in X$ -re $x \in g(Y)$	részalmaz def
$\Leftrightarrow$ minden $x \in X$ -re minden $y \in Y$ -ra xRy	Galois def
$\Leftrightarrow$ minden $y \in Y$ -ra minden $x \in X$ -re xRy	ekvivalencia
$\Leftrightarrow$ minden $y \in Y$ -ra $y \in f(X)$	Galois def
$\Leftrightarrow Y \subseteq f(X).$	részalmaz def

Tehát így pl. $\Sigma \subseteq \text{Th}(\mathcal{K}) \Leftrightarrow \mathcal{K} \subseteq \text{Mod}(\Sigma).$

$$X \subseteq g(f(X)) \text{ és } Y \subseteq f(g(Y)).$$

Írjunk az előzőbe $Y = f(X)$ -et: $X \subseteq g(f(X)) \Leftrightarrow f(X) \subseteq f(X)$. A jobb oldal nyilván igaz, tehát akkor a bal is. A másik irányhoz X legyen $g(Y)$.

Galois-kapcsolat

Jelölje $f \circ g : P(B) \rightarrow P(B)$ az $Y \mapsto f(g(Y))$, $g \circ f : P(A) \rightarrow P(A)$ pedig az $X \mapsto g(f(X))$ függvényt. „odamegyünk, visszajövünk”

Tetszőleges $X_1 \subseteq X_2 \subseteq A$ -ra $(g \circ f)(X_1) \subseteq (g \circ f)(X_2)$, vagyis $g \circ f$ **monoton**.

$$X_1 \subseteq X_2 \Rightarrow f(X_2) \subseteq f(X_1)$$

mert f antimonoton

$$\Rightarrow g(f(X_1)) \subseteq g(f(X_2))$$

mert g antimonoton

Teljesen hasonló módon $f \circ g$ is monoton.

Jelölje Cons ezt a $\text{Th} \circ \text{Mod}$ függvényt (azaz $\text{Cons}(\Sigma) := \text{Th}(\text{Mod}(\Sigma))$).

lesz értelme, nagyon is, stay tuned

Eddig azt kaptuk, hogy $\Sigma \subseteq \text{Cons}(\Sigma)$, és ha $\Sigma \subseteq \Delta$, akkor $\text{Cons}(\Sigma) \subseteq \text{Cons}(\Delta)$.

Galois-kapcsolat

$$f(g(f(X))) = f(X).$$

„ha hármat lépünk, mintha egyet lépnénk”

$$X \subseteq g(f(X)) \subseteq g(f(g(f(X))))$$

volt ilyen „bővít lemma”

$$\Leftrightarrow f(g(f(X))) \subseteq f(X)$$

$$X \subseteq g(Y) \Leftrightarrow Y \subseteq f(X) \text{ lemma}$$

és $f(X) \subseteq f(g(f(X)))$ is (megint a „bővít lemma”)

$$g(f(g(Y))) = g(Y).$$

(hasonlóan)

Tehát a példában nem véletlen kaptuk a harmadik lépésben vissza az első lépés eredményét: mindig így van.

Tehát $f \circ g$ és $g \circ f$ **idempotens** függvények: $(f \circ g)(f \circ g)(Y) = (f \circ g)(Y)$ és $(g \circ f)(g \circ f)(X) = (g \circ f)(X)$. „ha lépünk kettőt, aztán még kettőt, ugyanaz, mint csak kettőt”

Tehát, $\text{Cons}(\text{Cons}(\Sigma)) = \text{Cons}(\Sigma)$ is igaz például.

Lezárási operátor

Egy $h : P(A) \rightarrow P(A)$ függvényt **lezárási operátornak** nevezünk, ha

- monoton: $X \subseteq Y \Rightarrow h(X) \subseteq h(Y)$;
- bővítő: $X \subseteq h(X)$
- és idempotens: $h(h(X)) = h(X)$.

Tehát azt kaptuk eddig, hogy

Ha f és g Galois-kapcsolatban vannak, akkor $f \circ g$ is és $g \circ f$ is lezárási operátor.

Általában a lezárási operátorok is „jelentenek” valamit, ha f és g is. Sőt, sokszor maga a lezárási operátor a „legmotiváltabb” művelet az összes közül.

A $\models$ által generált Galois-kapcsolatban mi a Cons lezárási operátor jelentése?

Következmények

A lezárási operátorra általában igaz, hogy:

$$\begin{aligned} a \in (g \circ f)(X) &\Leftrightarrow \{a\} \subseteq g(f(X)) && \text{egyelemű halmaz } \subseteq \\ &\Leftrightarrow f(X) \subseteq f(\{a\}). && X \subseteq g(Y) \Leftrightarrow Y \subseteq f(X) \text{ lemma} \end{aligned}$$

Mit jelent ez a $\models$ esetében?

$F \in \text{Cons}(\Sigma)$ pontosan akkor igaz, ha $\Sigma \models F$.

Azaz: $\text{Cons}(\Sigma)$ -ban épp a Σ logikai következményei vannak!

Az is igaz, hogy minden $\text{Th}(\mathcal{K})$ alakú formulahalmaz egyben $\text{Cons}(\Sigma)$ alakú is (a $\Sigma = \text{Th}(\mathcal{K})$ -ra, önmagára alkalmazva Cons -t) és persze mivel $\text{Cons} = \text{Th} \circ \text{Mod}$, így minden $\text{Cons}(\Sigma)$ alakú halmaz egyben $\text{Th}(\mathcal{K})$ alakú is. Az ilyen formulahalmazokat nevezzük **elméletnek**.

Vagyis:

Σ elmélet, ha $\Sigma = \text{Cons}(\Sigma)$, azaz ha Σ tartalmazza az összes következményét.

Ha Σ elmélet, $F \in \Sigma$ és $F \equiv G$, akkor $G \in \Sigma$.

Hiszen ekkor $\text{Mod}(\Sigma) \subseteq \text{Mod}(F) = \text{Mod}(G)$.

Ha Σ elmélet, akkor tartalmazza az összes tautológiát.

Hiszen $\Sigma \models \uparrow$ minden Σ -ra igaz, így minden elmélet tartalmazza a $\uparrow$ formulát, és az előző állítás miatt a vele ekvivalenseket: a tautológiákat is.

Elméletek

Mivel egy elmélet is egy formulahalmaz, az lehet kielégíthető vagy kielégíthetetlen.

Ha Σ elmélet, akkor a következők ekvivalensek:

- i) Σ kielégíthetetlen (ekkor **ellentmondásos**nak is nevezzük);
- ii) $\downarrow \in \Sigma$;
- iii) Σ az összes formula halmaza;
- iv) van olyan F formula, hogy $F, \neg F \in \Sigma$.

i) $\rightarrow$ ii) Ha Σ kielégíthetetlen, akkor $\Sigma \models \downarrow$. Ha Σ elmélet, akkor tartalmazza a következményeit, tehát ekkor $\downarrow \in \Sigma$.

ii) $\rightarrow$ iii) Mivel $\downarrow \rightarrow F$ tetszőleges F -re tautológia, így Σ -beli; mivel $\{\downarrow, \downarrow \rightarrow F\} \models F$, így ekkor $F \in \Sigma$.

iii) $\rightarrow$ iv) ha minden formula benne van, akkor persze hogy találni ilyen

iv) $\rightarrow$ i) nincs olyan $\mathcal{A}$, mely egyszerre modellje F -nek is és $\neg F$ -nek is

Galois-kapcsolatok máshol

Ha $A = B$ egy (véges dimenziós) **vektortér** elemei (mint pl. $A = B = \mathbb{R}^n$) és a relációnk a $\perp$ (két vektor relációjában van, ha merőlegesek, skaláris szorzatuk 0), akkor:

- az f és g függvények vektorok egy U halmazához az összes, U minden elemére merőleges vektor halmazát rendelik (aminek jele $U^\perp$);
- az $f \circ g$ lezárási operátor pedig az U vektorhalmazhoz az általa kifeszített alteret rendeli (jele $\langle U \rangle$)

Akkor a Galois-kapcsolatból pl. olyanokat kapunk, hogy

- $\langle \langle U \rangle \rangle = \langle U \rangle$
- tetszőleges U vektorhalmazra $U^\perp = \langle U \rangle^\perp$
- $X \subseteq Y^\perp$ pont akkor, ha $Y \subseteq X^\perp$
- ha $U \subseteq V$, akkor $V^\perp \subseteq U^\perp$
- stb. „ingyen”

Galois-kapcsolatok máshol

Ha pl. $A = B$ egy **részbenrendezett halmaz** és a relációnk a $\leq$ részbenrendezés, akkor

- $f(X) = \{y : \forall x \in X \ x \leq y\}$ az X felső korlátainak halmaza;
- $g(Y)$ az Y alsó korlátainak halmaza;
- a lezárási operátorok lezárják X -et „lefelé” ill. „felfelé” és (ha van) beveszik a szuprémumot /infimumot is – tehát $(g \circ f)(X) = \{y : y \leq \bigvee X\}$, ha a szuprémum létezik és hasonlóan $(f \circ g)(Y) = \{x : \bigwedge Y \leq x\}$

Erre az esetre a Galois-kapcsolatból azt kapjuk ingyen pl., hogy

- ha $X \subseteq Y$ és létezik a szuprémumuk/infimumuk, akkor $\bigvee X \leq \bigvee Y$ és $\bigwedge X \leq \bigwedge Y$;
- egy X halmaz felső korlátjainak infimuma az épp X szuprémuma (ha létezik);
- stb.

Lezárási operátorok

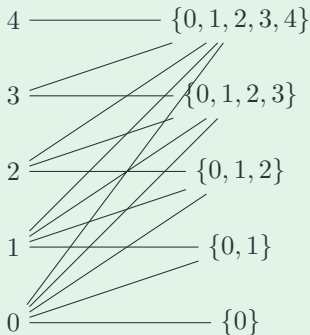
Nem csak az igaz, hogy minden (f, g) Galois-kapcsolat meghatároz egy $f \circ g$ (és egy $g \circ f$) lezárási operátort, de fordítva is:

Minden lezárási operátor előáll $X \mapsto g(f(X))$ alakban egy alkalmas R reláció által meghatározott (f, g) Galois-kapcsolatra.

- Legyen $h : P(A) \rightarrow P(A)$ egy lezárási operátor: monoton, bővítő, idempotens.
- Legyen $B = \{C \subseteq A : h(C) = C\}$ a **zárt** halmazok halmaza.
- Legyen a relációnk az $a \in C$: a lezárt halmazokat (mint B -beli elemeket) hozzuk relációba elemeikkel (mint A -beli elemekkel).
- Akkor $f(X) = \{C \subseteq A : C \text{ zárt, } X \subseteq C\}$ az összes olyan zárt halmaz halmaza, amiknek X része;
- Akkor $g(Y) = \{a \in A : a \in C \text{ minden } C \in Y\text{-ra}\}$. Azaz: $g(Y) = \bigcap_{C \in Y} C$ épp az Y -beli zárt halmazok metszete.

Lezárási operátorok

Ha pl. $h : P(\mathbb{N}_0) \rightarrow P(\mathbb{N}_0)$ a $h(X) := \{n : \exists m \in X \ n \leq m\}$ operátor:
...



- $f(\{0\})$: az összes, 0-t tartalmazó zárt halmaz halmaza
- $f(\{2\})$: az összes, 2-t tartalmazó zárt halmaz halmaza
- $f(\{0, 2\})$: az összes, 0-t és 2-t is tartalmazó zárt halmaz halmaza
- $g(\{\{0, 1, 2\}, \{0, 1, 2, 3\}\})$: a két zárt halmaz metszete, $\{0, 1, 2\}$

Bizonyítás, tovább

- Tehát $(g \circ f)(X)$ az összes, X -et tartalmazó zárt halmaz metszete.
- Zárt halmazok metszete is zárt: ha $h(X_i)$ zárt halmazok, akkor $h(\bigcap h(X_i)) \subseteq h(h(X_i)) = h(X_i)$, tehát $h(\bigcap h(X_i)) \subseteq \bigcap h(X_i)$, mivel pedig h bővítő, így $h(\bigcap h(X_i)) = \bigcap h(X_i)$.
- Mivel $X \subseteq h(X)$ és $h(X)$ zárt halmaz, így $(g \circ f)(X) \subseteq h(X)$ is igaz.
- Tehát $X \subseteq (g \circ f)(X) \subseteq h(X)$ és az utóbbi kettő zárt halmaz.
- Akkor a monotonitás miatt $h(X) \subseteq h(g \circ f)(X) = (g \circ f)(X)$ is igaz, tehát tényleg $h(X) = (g \circ f)(X)$.

Tehát: ha lezárási operátorral találkozunk, az alatt is mindig ott egy Galois-kapcsolat.

Műveletek egy családjára való lezárás

Legyen A egy alaphalmaz, O pedig $A^n \rightarrow A$ parciális függvények egy halmaza.

parciális: nem feltétlenül mindenhol definiált

Akkor ha $X \subseteq A$, úgy $O(X)$ jelöli az

$$X \cup \{f(x_1, \dots, x_n) : f/n \in O, x_1, \dots, x_n \in X\}$$

halmazt.

Tehát: X -be vegyük be az összes olyan $f(x_1, \dots, x_n)$ értéket is, amire f egy O -beli művelet, az $x_1, \dots, x_n$ elemek pedig X -beliek.

Ha pl. az A alaphalmazunk a klózik halmaza (az **összes** klóz halmaza), és **Res**-ben a rezolvensképzés szerepel műveletként, akkor ha Σ klózik egy halmaza, úgy $\mathbf{Res}(\Sigma) = \Sigma \cup \{R : R \text{ két } \Sigma\text{-beli klóz rezolvense}\}$.

Nyilván igaz, hogy $X \subseteq O(X)$ és ha $X \subseteq Y$, akkor $O(X) \subseteq O(Y)$.

Műveletek egy családjára való lezárás

Jelölje $O^*(X)$ az $\bigcup_{n \geq 0} O^n(X)$ halmazt.

azaz: az $X \cup O(X) \cup O(O(X)) \cup \dots$ halmazt.

Példa

$\text{Res}^*(\Sigma)$ -ban az összes, Σ -ból rezolúcióval levezethető klóz szerepel.

Véges változószámú parciális műveletek tetszőleges O családjára

O^* lezárási operátor.

- Ehhez azt elég megmutatnunk, hogy $O(O^*(X)) = O^*(X)$.
- $O(O^*(X)) = O^*(X) \cup \{f(x_1, \dots, x_n) : f \in O, x_i \in O^*(X)\}$.
- Ha $x_1 \in O^{k_1}(X)$, $x_2 \in O^{k_2}(X), \dots$, akkor mindegyik $x_i \in O^N(X)$ az $N = \max\{k_1, \dots, k_n\}$ -re.
- Akkor $f(x_1, \dots, x_n) \in O^{N+1}(X) \subseteq O^*(X)$.
- Tehát $O(O^*(X)) = O^*(X)$. Ezért $O(O(O^*(X))) = O^*(X)$ stb, így $O^*(O^*(X)) = O^*(X)$ és O^* tényleg lezárási operátor.

Műveletek egy családjára való lezárás

A Galois-kapcsolat miatt pedig:

- az $O^*(X)$ alakú halmazok épp azok, melyek zártak O összes műveletére;
- az $O^*(X)$ halmaz pedig az összes ilyen, X -et tartalmazó zárt halmaz metszete,
- ami szintén egy zárt halmaz,
- ezt tehát mondhatjuk így is: $O^*(X)$ a **legszűkebb olyan halmaz, mely tartalmazza X -et és mely zárt O összes műveletére.**

(Ilyet is láttunk már: így készítettük pl. a formulák halmazát is.)

Ha egy $h : P(A) \rightarrow P(A)$ lezárási operátort fel tudunk írni $h = O^*$ alakban úgy, hogy O -ban véges sok (kiszámítható) függvény szerepel, akkor tetszőleges $X \subseteq A$ -ra, ha X elemeinek generálására van algoritmus, úgy $O^*(X)$ elemeit is szisztematikusan tudjuk generálni.

Deduktív rendszerek

- Egy **deduktív rendszer** inputként kap egy Σ formulahalmazt és valamilyen algoritmikus módon kigenerálja Σ **összes következményét**. (Ez mindig egy végtelen halmaz, pl. az összes tautológia is benne van.)
- Azt tudjuk, hogy Cons egy lezárási operátor.
- A cél: műveleteknek egy olyan „kicsi” O halmazát megadni (ezeket fogja végezni a következtető rendszer), melyre $O^* = \text{Cons}$.
- Ekkor a deduktív rendszer csak O műveleteit kezdi el elvégezni Σ elemein valamilyen „fair” sorrendben (fair: előbb-utóbb minden műveletsor sorra kerül), és ezzel $\text{Cons}(\Sigma)$ minden elemét megkapja előbb-utóbb.
- A rezolúció önmagában nem ilyen rendszer: az nem igaz, hogy minden következményt le lehet vezetni rezolúcióval.
- Egy ilyen rendszer lesz viszont a Hilbert-rendszer. Azt viszont (szintén) csak **speciális** alakú formulákon definiáljuk.

Teljes rendszerek

Boole-függvények egy H rendszere **teljes** vagy **adekvát**, ha minden $n \geq 1$ -változós Boole-függvény előáll

- a projekciókból
- és H elemeiből
- alkalmas **kompozícióval**.

π_i jelölte az i -edik változó kiválasztását

Kompozíció

Ha f/n és $g_1/k, \dots, g_n/k$ Boole-függvények, akkor az $f \circ \langle g_1, \dots, g_n \rangle$ az a k -változós Boole-függvény, melyre

$$(f \circ \langle g_1, \dots, g_n \rangle)(x_1, \dots, x_k) = f(g_1(x_1, \dots, x_k), \dots, g_n(x_1, \dots, x_k)).$$

pl. $\wedge = \neg \circ \vee \circ \langle \neg \circ \pi_1, \neg \circ \pi_2 \rangle$, hiszen

$$\begin{aligned}(\neg \circ \vee \circ \langle \neg \circ \pi_1, \neg \circ \pi_2 \rangle)(x_1, x_2) &= \neg((\vee \circ \langle \neg \circ \pi_1, \neg \circ \pi_2 \rangle)(x_1, x_2)) \\&= \neg((\vee((\neg \circ \pi_1)(x_1, x_2), (\neg \circ \pi_2)(x_1, x_2)))) \\&= \neg(\vee(\neg x_1, \neg x_2))\end{aligned}$$

Teljes rendszerek

```
def inc: Double => Double = { x => x + 1 }  
def dup: Double => Double = { x => x * 2 }  
def combo = inc compose dup  
println(combo(5)) //prints 11
```

Persze ezt az előzőt érdemesebb lehet inkább úgy kiírni kézzel, hogy

$$x_1 \wedge x_2 = \neg(\neg x_1 \vee \neg x_2).$$

„ $\wedge$ függvény kifejezhető a $\{\neg, \vee\}$ függvényekkel”

Hogy H teljes, az pont azt jelenti, hogy ha vesszük azt az O operátort, melyben a projekciók és H elemei szerepelnek, akkor erre $O^*(\emptyset)$ tartalmazza az összes Boole-függvényt. Tehát megint egy műveletcsaláddal definiált lezárási operátorról van szó.

Recap: Shannon-expanzió

Azt már láttuk, hogy a $\{\neg, \vee, \wedge\}$ rendszer teljes.

Teljes rendszer még pl: $\{\neg, \wedge\}$, $\{\rightarrow, \downarrow\}$, $\{\text{nand}\}$, $\{\text{nor}\}$.

$$x \text{ nand } y := \neg(x \wedge y)$$

$$x \text{ nor } y := \neg(x \vee y)$$

(gyakorlaton ezeket megnézzük)

Note: hogy egy következtető algoritmus „teljes” és hogy Boole-függvények egy rendszere „teljes”, az egész mást jelent!

Teljes rendszerek

Most a célunk bebizonyítani a következőt:

Ha az $f/2$ Boole-függvény egyedül is teljes rendszert alkot, akkor $f = \text{nand}$ vagy $f = \text{nor}$.

Egy f/n Boole-függvény...

- **0-őrző**, ha $f(0, 0, \dots, 0) = 0$
- **1-őrző**, ha $f(1, 1, \dots, 1) = 1$
- **monoton**, ha $x_1 \leq y_1, \dots, x_n \leq y_n \Rightarrow f(x_1, \dots, x_n) \leq f(y_1, \dots, y_n)$
- **önduális**, ha $f(\neg x_1, \dots, \neg x_n) = \neg f(x_1, \dots, x_n)$
- **lineáris**, ha van olyan $I \subseteq \{1, \dots, n\}$ halmaz és $c \in \{0, 1\}$ konstans, melyre $f(x_1, \dots, x_n) = c + \sum_{i \in I} x_i$ (itt $+$ a xor művelet)

Boole-függvények tulajdonságai, példák

- Az $\wedge$ és $\vee$ függvények őrzik a 0-t és az 1-et is: $0 \wedge 0 = 0 \vee 0 = 0$,
 $1 \wedge 1 = 1 \vee 1 = 1$. $A \rightarrow$ őrzik az 1-et, de nem őrzik a 0-t. $A \neg$ nem őrzik egyiket se.
- $A \wedge$ és $\vee$ függvények monotonok: ha növeljük bármelyik argumentum értékét, az eredmény nem fog csökkenni. $A \neg$ nem monoton, mert $0 \leq 1$, de $\neg 0 \not\leq \neg 1$.
 $A \rightarrow$ sem monoton, mert $1 = 0 \rightarrow 0 \not\leq 1 \rightarrow 0 = 0$.
- $A \neg$ függvény önduális, hiszen $f(\neg x) = \neg(\neg x) = f(\neg x)$. $A \wedge$ nem önduális, hiszen $\neg(x \wedge y) \neq \neg x \wedge \neg y$ (pl. ha $x = 1$ és $y = 0$).
- $A \neg$ függvény lineáris: $\neg x = 1 + x$. $A \wedge$ nem lineáris ($x \wedge y$ nem egyenlő semmivel, ami előáll 1, x és y közül néhánynak a xorjaként)
- A projekciók őrzik a nullát és az egyet is, monotonak, lineárisak és önduálisak is.

Boole-függvények tulajdonságai

Ha H Boole-függvények olyan halmaza, melyben minden függvény őrzi a 0-t, akkor minden olyan függvény, mely belőlük és a projekciókból áll elő kompozícióval, szintén őrzi a 0-t.

- A projekciók őrzi a 0-t: $\pi_i(0, \dots, 0) = 0$.
- Ha $f = g \circ \langle h_1, \dots, h_k \rangle$ a $g/k, h_1/n, \dots, h_k/n$ 0-őrző függvényekre, akkor

$$\begin{aligned} f(0, 0, \dots, 0) &= g(h_1(0, \dots, 0), \dots, h_k(0, \dots, 0)) && \text{kompozíció def} \\ &= g(0, \dots, 0) && \text{mert minden } h_i \text{ őrzi a 0-t} \\ &= 0 && \text{mert } g \text{ őrzi a 0-t} \end{aligned}$$

Tehát ha H Boole-függvényeknek egy teljes rendszere, akkor **kell** legyen H -ban olyan függvény, mely **nem** őrzi a 0-t.

pl. $\{\neg, \vee\}$ -ben $\neg$, $\{\rightarrow, \downarrow\}$ -ban $\rightarrow$ nem őrzi a 0-t

Boole-függvények tulajdonságai

Teljesen hasonló állítások (ugyanígy bizonyítva) igazak a másik négy tulajdonságra is.

Ha H Boole-függvényeknek egy teljes rendszere, akkor kell legyen benne...

- olyan függvény, mely nem őrzi a 0-t;
- olyan függvény, mely nem őrzi az 1-et;
- olyan függvény, mely nem monoton;
- olyan függvény, mely nem önduális;
- és olyan függvény is, mely nem lineáris.

Pl. $\{\neg, \vee\}$ -ben $\neg$ nem őrzi a konstansokat és nem monoton, $\vee$ pedig nem önduális és nem is lineáris.

$\{\rightarrow, \downarrow\}$ -ban $\rightarrow$ nem őrzi a 0-t, nem monoton és nem lineáris; $\downarrow$ nem őrzi az 1-et (!) és nem önduális (mert $\neg \downarrow () \neq \downarrow ()$).

Ezek szerint ha $f/2$ egy olyan Boole-függvény, melyre $\{f\}$ teljes, akkor

- $f(0, 0) = 1$ kell legyen (különben őrizné a 0-t)
- $f(1, 1) = 0$ kell legyen (különben őrizné az 1-et)
- ha $f(0, 1) = 0$ és $f(1, 0) = 1$, akkor $f(x, y) = 1 + y$, lineáris lenne, ez nem lehet
- ha $f(0, 1) = 1$ és $f(1, 0) = 0$, akkor $f(x, y) = 1 + x$, lineáris lenne, ez sem lehet
- tehát vagy $f(0, 1) = 0$ és $f(1, 0) = 0$ (és akkor $f = \text{nor}$)
- vagy $f(0, 1) = 1$ és $f(1, 0) = 1$ (és akkor $f = \text{nand}$).

Post tétele

Boole-függvények egy H rendszere **pontosan akkor teljes**, ha mind az öt tulajdonságra létezik benne olyan függvény, mely **nem** rendelkezik az adott tulajdonsággal.

Hilbert rendszere egy **deduktív rendszer**: az input Σ **összes** következményét (és csak azokat) lehet vele levezetni.

Ebben a rendszerben csak a $\rightarrow$ konnektívát és a $\downarrow$ logikai konstanst használhatjuk az ítéletváltozókon kívül.

Minden formula ilyen alakra hozható, mert a $\{\rightarrow, \downarrow\}$ rendszer teljes.

A Hilbert rendszer **axiómái**

$$\text{Ax1 :} \quad (F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$$

$$\text{Ax2 :} \quad F \rightarrow (G \rightarrow F)$$

$$\text{Ax3 :} \quad ((F \rightarrow \downarrow) \rightarrow \downarrow) \rightarrow F$$

(Note: ezek a formulák tautológiák.)

$$(F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$$

Tautológia:

- Ha F hamis, akkor $1 \rightarrow (1 \rightarrow 1) = 1$
- Ha F igaz, akkor ez ekvivalens a $(G \rightarrow H) \rightarrow (G \rightarrow H)$ -val, ami szintén igaz minden értékadás mellett

Az axiómák

$$F \rightarrow (G \rightarrow F)$$

Tautológia:

ha F igaz, a jobb oldal igaz; ha F hamis, a bal oldal hamis

A sorrend fontos, a $\rightarrow$ **nem** asszociatív!

$(F \rightarrow G) \rightarrow F$ **nem** tautológia: ha F hamis, G igaz, ez a formula hamis

$A \rightarrow$ konvenciója: jobb-asszociatív, $F \rightarrow G \rightarrow H$ az $F \rightarrow (G \rightarrow H)$ -t jelenti, de a könnyebb olvashatóság érdekében ezt igyekszünk kerülni a példákban

Note: egyébként $F \vee (G \wedge H)$ sem ekvivalens $(F \vee G) \wedge H$ -val. Pl. $1 \vee (0 \wedge 0) = 1 \vee 0 = 1$, de $(1 \vee 0) \wedge 0 = 1 \wedge 0 = 0$. Azaz a „magukban” kommutatív, asszociatív műveletek **nem lesznek** automatikusan azok „egymás közt” is.

$$((F \rightarrow \downarrow) \rightarrow \downarrow) \rightarrow F$$

Tautológia:

Az $F \rightarrow \downarrow$ részformulát akár $\neg F$ -nek is írhatnánk.

Tehát ez $\neg \neg F \rightarrow F$, a kettős negálás eliminálása, tautológia

Az axiómák példányai

A fenti három axióma egy **példánya**: valamelyik axiómában szereplő F, G, H helyére **tetszőleges formulát írunk**.

(Ugyanazon betű helyére persze ugyanazt a formulát.)

Ennek a műveletnek jelet is adunk:

Ha F egy formula, melyben a $p_1, \dots, p_n$ változók szerepelnek, és $F_1, \dots, F_n$ formulák, akkor

$$F[p_1/F_1, \dots, p_n/F_n]$$

jelöli azt a formulát, melyet úgy kapunk F -ből, hogy benne minden p_i helyére az F_i formulát írjuk.

Példa

$$(F \rightarrow (G \rightarrow F))[F/p, G/(p \rightarrow p)] = (p \rightarrow ((p \rightarrow p) \rightarrow p)).$$

A modus ponens

A leválasztási következtetés, vagy modus ponens

$$\{F, F \rightarrow G\} \models G.$$

Ez egy helyes következtetési szabály.

Levezetés Hilbert rendszerében

Legyen Σ formulák egy halmaza, F pedig egy formula. Azt mondjuk, hogy F **levezethető Σ -ból Hilbert rendszerében**, jelben $\Sigma \vdash F$, ha van olyan $F_1, F_2, \dots, F_n$ formula-sorozat, melynek minden eleme

- Σ -beli vagy
- axiómapéldány vagy
- előáll két korábbiból modus ponenssel

és melyre $F_n = F$. (Ha Σ üres, akkor $\emptyset \vdash F$ helyett $\vdash F$ -et is írhatunk.)

$$\vdash p \rightarrow p.$$

$$\text{Ax1: } (F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$$

$$\text{Ax2: } F \rightarrow (G \rightarrow F)$$

$$1. p \rightarrow ((p \rightarrow p) \rightarrow p)$$

$$\text{Ax2}[F/p, G/(p \rightarrow p)]$$

$$2. (p \rightarrow ((p \rightarrow p) \rightarrow p)) \rightarrow ((p \rightarrow (p \rightarrow p)) \rightarrow (p \rightarrow p))$$

$$\text{Ax1}[F/p, G/(p \rightarrow p), H/p]$$

$$3. (p \rightarrow (p \rightarrow p)) \rightarrow (p \rightarrow p)$$

$$\text{MP}(1, 2)$$

$$4. p \rightarrow (p \rightarrow p)$$

$$\text{Ax2}[F/p, G/p]$$

$$5. p \rightarrow p$$

$$\text{MP}(3, 4)$$

(Note: mindig csak a külső $\rightarrow$ mentén választhatunk le!)

$$\{p \rightarrow q\} \vdash p \rightarrow (r \rightarrow q).$$

$$\text{Ax1: } (F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$$

$$\text{Ax2: } F \rightarrow (G \rightarrow F)$$

$$1. \left(p \rightarrow (q \rightarrow (r \rightarrow q)) \right) \rightarrow \left((p \rightarrow q) \rightarrow (p \rightarrow (r \rightarrow q)) \right)$$

$$\text{Ax1}[F/p, G/q, H/(r \rightarrow q)]$$

$$2. q \rightarrow (r \rightarrow q)$$

$$\text{Ax2}[F/q, G/r]$$

$$3. (q \rightarrow (r \rightarrow q)) \rightarrow (p \rightarrow (q \rightarrow (r \rightarrow q)))$$

$$\text{Ax2}[F/(q \rightarrow (r \rightarrow q)), G/p]$$

$$4. p \rightarrow (q \rightarrow (r \rightarrow q)) \quad \text{MP}(2, 3)$$

$$5. (p \rightarrow q) \rightarrow (p \rightarrow (r \rightarrow q)) \quad \text{MP}(1, 4)$$

$$6. p \rightarrow q \quad \in \Sigma$$

$$7. p \rightarrow (r \rightarrow q) \quad \text{MP}(5, 6)$$

$$\{p \rightarrow q, q \rightarrow r\} \vdash p \rightarrow r.$$

$$\text{Ax1: } (F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow G) \rightarrow (F \rightarrow H))$$

$$\text{Ax2: } F \rightarrow (G \rightarrow F)$$

$$1. (p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r))$$

Ax1

$$2. q \rightarrow r$$

$\in \Sigma$

$$3. (q \rightarrow r) \rightarrow (p \rightarrow (q \rightarrow r))$$

Ax2

$$4. p \rightarrow (q \rightarrow r)$$

MP(2, 3)

$$5. (p \rightarrow q) \rightarrow (p \rightarrow r)$$

MP(1, 4)

$$6. p \rightarrow q$$

$\in \Sigma$

$$7. p \rightarrow r$$

MP(5, 6)

$$\vdash \downarrow \rightarrow p.$$

(Gyakorlaton.)

- A fenti példákban p helyére tetszőleges F formulát írva minden levezetésben megkapjuk, hogy $\vdash (F \rightarrow F)$ és $\vdash (\downarrow \rightarrow F)$ minden F formulára igaz.
- Hasonlóan, ha $\Sigma \vdash F \rightarrow G$ és $\Sigma \vdash G \rightarrow H$, akkor $\Sigma \vdash F \rightarrow H$ a harmadik példa alapján.
- Ha $\Sigma \vdash F$ és $\Sigma \subseteq \Gamma$, akkor $\Gamma \vdash F$ (a nem Σ -beli elemeket nem használjuk fel)

A célunk bebizonyítani Hilbert rendszerének helyességi és teljességi tételét:

$$\Sigma \vdash F \quad \Leftrightarrow \quad \Sigma \models F$$

tetszőleges Σ formulahalmazra és F formulára.

persze itt a formulák csak a $\rightarrow$ és a $\downarrow$ konnektívákat tartalmazhatják

Kielégíthető formulából is kaphatunk kielégíthetetlen, vagy tautológiát:

- $F \wedge G$ kielégíthető, de $(F \wedge G)[F/p, G/\neg p] = p \wedge \neg p$ kielégíthetetlen
- $F \vee G$ nem tautológia, de $(F \vee G)[F/p, G/\neg p] = p \vee \neg p$ az

viszont

Tautológia példányai is tautológiák.

Tehát a Hilbert rendszer axióma-példányai tautológiák.

Az előző állítás az alábbiak a következménye:

Legyenek az F formulában szereplő változók $p_1, \dots, p_n$, és $F_1, \dots, F_n$ további formulák (melyekben más változók is előfordulhatnak).

Legyen $\mathcal{A}$ egy tetszőleges értékadás.

Definiáljuk a $\mathcal{B}$ értékadást a következőképpen:

$$\mathcal{B}(p_i) := \mathcal{A}(F_i).$$

(a p_i értéke $\mathcal{B}$ -ben legyen az az érték, ami F_i értéke $\mathcal{A}$ -ban.)

Ekkor

$$\mathcal{B}(F) = \mathcal{A}(F[p_1/F_1, \dots, p_n/F_n]).$$

Valóban következménye, hiszen ha F tautológia, akkor ez a fenti $\mathcal{B}(F)$ minden $\mathcal{A}$ eredeti értékadásra igaz lesz.

Bizonyítás

Az F formula felépítése szerinti indukciót használunk:

- Ha $F = p_i$, akkor $F[p_1/F_1, \dots, p_n/F_n] = F_i$.
Ekkor $\mathcal{B}(p_i) = \mathcal{A}(F_i)$ a $\mathcal{B}$ definíciója miatt.
- Ha $F = \neg G$, akkor

$$\begin{aligned}\mathcal{B}(F) &= \neg \mathcal{B}(G) && \neg \text{ szemantikája szerint} \\ &= \neg \mathcal{A}(G[p_1/F_1, \dots, p_n/F_n]) && \text{ indukciós feltevés szerint} \\ &= \mathcal{A}(\neg G[p_1/F_1, \dots, p_n/F_n]) && \neg \text{ szemantikája szerint} \\ &= \mathcal{A}(F[p_1/F_1, \dots, p_n/F_n]) && \text{ a példányosítás def szerint}\end{aligned}$$

- Ha $F = G \rightarrow H$, akkor

$$\begin{aligned}\mathcal{B}(G \rightarrow H) &= \mathcal{B}(G) \rightarrow \mathcal{B}(H) \\ &= \mathcal{A}(G[p_1/F_1, \dots, p_n/F_n]) \rightarrow \mathcal{A}(H[p_1/F_1, \dots, p_n/F_n]) \\ &= \mathcal{A}(G[p_1/F_1, \dots, p_n/F_n] \rightarrow H[p_1/F_1, \dots, p_n/F_n]) \\ &= \mathcal{A}(F[p_1/F_1, \dots, p_n/F_n]). \text{ (többi eset hasonló)}\end{aligned}$$

A jólmegalapozott indukció

Láttunk már indukciót

- természetes számokra: „0-ra igaz és ha n -re igaz, akkor $n + 1$ -re is”
- formulákra: „változókra igaz, és ha a formula közvetlen részformuláira is igaz, akkor a formulára is”
- függvények kompozícióira: „az x_i -re igaz és ha $f_1, \dots, f_k$ -ra és f -re igaz, akkor igaz $f \circ \langle f_1, \dots, f_k \rangle$ -ra is”

A módszer persze nem mindig működik, pl:

- „minden nemnegatív valós szám egész”
- „mert a 0 egész”
- „és az $r > 0$ valósra az $r/2$ kisebb”
 - „indukciós feltevés szerint $r/2$ tehát egész”
 - „tehát $2 \cdot (r/2) = r$ is egész”
- „tehát minden nemnegatív valós szám egész”

nyilvánvalóan hibás következtetés.

Valós számokra nem használhatunk indukciót. **Miért?**

A jólmegalapozott indukció

Általában egy indukciós bizonyítás a következőképp néz ki:

- van **objektumoknak** egy A halmaza (formulák, természetes számok,...)
- köztük egy **$\prec$ reláció**, $x \prec y$ kb. az „ x egyszerűbb/kisebb, mint y ” ($n \prec n + 1$, a formula részformulái kisebbek a formulánál,...)
- az alapeset: megmutatjuk azokra az x elemekre az állítást, akiknél nincs kisebb (0-ra, a változókra,...)
- az indukciós lépés: megmutatjuk, hogy ha az összes x -nél kisebb elemre igaz az állítás, akkor x -re is
- és ezzel megmutattuk, hogy minden A -beli objektumra igaz az állítás.

A fenti általános bizonyítási módszer akkor működik, ha $\prec$ -ben nincs végtelen leszálló lánc, vagyis

nincs ... $\prec x_3 \prec x_2 \prec x_1 \prec x_0$ sorozat.

A módszer neve: **jólmegalapozott indukció**.

ha nincs végtelen leszálló lánc $\prec$ -ben, akkor ő egy „well partial order”, WPO

A teljes és a strukturális indukció

A teljes indukció

Az objektumok a természetes számok: $\{0, 1, 2, \dots\}$, a reláció pedig

$$0 \prec 1 \prec 2 \prec 3 \prec \dots$$

Itt $\prec$ -ben nincs végtelen **leszálló** lánc (felszálló van, az nem baj)

A felépítés szerinti vagy strukturális indukció

Az objektumok a formulák és $F \prec G$, ha F a G -nek közvetlen részformulája.

Itt sincs $\prec$ -ben végtelen leszálló lánc.

A felépítés szerinti indukció minden ún. „algebrai adattípusra” működik $\approx$ alap komponensekből építünk fel véges objektumokat valamilyen képzési szabályok szerint

A valós számokra azért nem, mert pl. $\dots < 1/16 < 1/8 < 1/4 < 1/2 < 1$ egy végtelen leszálló lánc

A rezolúció helyességénél is ezt használtuk: ott $C \prec C'$ volt, ha C előbb került a listára, mint C'

Vissza a Hilbert-kalkulushoz: Helyesség

Ha $\Sigma \vdash F$, akkor $\Sigma \models F$

A helyesség iránya könnyű: ha $\Sigma \vdash F$, akkor $\Sigma \models F$, hiszen

- Legyen $F_1, \dots, F_n$ egy Σ fölötti levezetése F -nek. Teljes indukcióval megmutatjuk, hogy minden i -re $\Sigma \models F_i$.
- **Ha $F_i \in \Sigma$** , akkor $\Sigma \models F_i$ (ez minden Σ , F -re igaz, láttuk)
- **Ha F_i axiómapéldány**, akkor $\emptyset \models F_i$ (a tautológiák minden elméletben szerepelnek), így a monotonitás miatt $\Sigma \models F_i$ is igaz
- **Ha pedig $F_i = \text{MP}(F_j, F_k)$** a $j, k < i$ indexekre, akkor
 - Az **indukciós feltevés szerint** $\Sigma \models F_j$ és $\Sigma \models F_k$
 - Tehát $\Sigma \models \{F_j, F_k\}$
 - MP def miatt $F_k = F_j \rightarrow F_i$: $\Sigma \models \{F_j, F_j \rightarrow F_i\}$
 - A leválasztási következtetés: $\{F_j, F_j \rightarrow F_i\} \models F_i$
 - A tranzitivitás miatt tehát $\Sigma \models F_i$.

Ezzel beláttuk, hogy a Hilbert-rendszer helyes következtető rendszer.

Teljesség – a dedukciós tétel

A teljességhez először belátjuk a Hilbert-rendszer **dedukciós tételét**:

A dedukciós tétel

Tetszőleges Σ formulahalmazra és F, G formulákra

$$\Sigma \vdash (F \rightarrow G) \quad \Leftrightarrow \quad \Sigma \cup \{F\} \vdash G.$$

Ennek az állításnak megint az egyik irányát könnyebb, a másikat nehezebb belátni.

$$\Sigma \vdash (F \rightarrow G) \quad \Rightarrow \quad \Sigma \cup \{F\} \vdash G$$

Ha $\Sigma \vdash (F \rightarrow G)$, akkor

- $\Sigma \cup \{F\} \vdash (F \rightarrow G)$ (ugyanaz a levezetés jó)
- $\Sigma \cup \{F\} \vdash F$ (mert eleme)
- $\Sigma \cup \{F\} \vdash G$ (az előző kettőből MP)

A dedukciós tétel

$$\Sigma \cup \{F\} \vdash G \quad \Rightarrow \quad \Sigma \vdash (F \rightarrow G)$$

Legyen $F_1, \dots, F_n$ a G formula $\Sigma \cup \{F\}$ fölötti levezetése. Megmutatjuk **teljes indukcióval**, hogy minden i -re $\Sigma \vdash (F \rightarrow F_i)$.

- Ha $F_i \in \Sigma \cup \{F\}$, akkor két eset lehetséges:
 - Ha $F_i \in \Sigma$, akkor
 - $\Sigma \vdash F_i$ (mert eleme)
 - $\Sigma \vdash F_i \rightarrow (F \rightarrow F_i)$ (Ax2)
 - $\Sigma \vdash F \rightarrow F_i$ (MP)
 - Ha $F_i = F$, akkor $\Sigma \vdash F \rightarrow F$ (öt lépésben, ezt már láttuk)
- Ha F_i axiómapéldány, akkor ugyanúgy, mint mikor Σ -beli:
 - $\Sigma \vdash F_i$ (mert axiómapéldány)
 - $\Sigma \vdash F_i \rightarrow (F \rightarrow F_i)$ (Ax2)
 - $\Sigma \vdash F \rightarrow F_i$ (MP)

nincs még kész

(Note: a második axióma példányosításának sokszor ez a célja: megvan az F formulánk és egy $G \rightarrow F$ -re van szükségünk)

$$\Sigma \cup \{F\} \vdash G \quad \Rightarrow \quad \Sigma \vdash (F \rightarrow G)$$

- Ha $F_i = \text{MP}(F_j, F_k)$, ahol $j, k < i$, akkor:
 - $F_k = (F_j \rightarrow F_i)$ és az **indukciós feltevés szerint** $\Sigma \vdash F \rightarrow F_j$ és $\Sigma \vdash F \rightarrow (F_j \rightarrow F_i)$.
 - $(F \rightarrow (F_j \rightarrow F_i)) \rightarrow ((F \rightarrow F_j) \rightarrow (F \rightarrow F_i))$ Ax1
 - $(F \rightarrow F_j) \rightarrow (F \rightarrow F_i)$ MP
 - $F \rightarrow F_i$ MP

done

Ezzel bebizonyítottuk a dedukciós tételt.

Ezt felhasználva most bebizonyítjuk a Hilbert-rendszer teljességét is.

A Hilbert-rendszer teljessége: H-konzisztencia

de kell hozzá még egy def

H-konzisztens halmazok

Egy Σ formulahalmazt **H-konzisztensnek** nevezünk, ha **nem** igaz, hogy $\Sigma \vdash \downarrow$.

A következők ekvivalensek tetszőleges Σ formulahalmazra:

- i) Van olyan F formula, melyre $\Sigma \vdash F$ és $\Sigma \vdash (F \rightarrow \downarrow)$ is igaz.
- ii) Σ **nem** H-konzisztens.
- iii) $\Sigma \vdash F$ minden F formulára.

i) $\rightarrow$ ii) Ha $\Sigma \vdash \{F, F \rightarrow \downarrow\}$, akkor ezekből MP-vel $\Sigma \vdash \downarrow$.

ii) $\rightarrow$ iii) Ha $\Sigma \vdash \downarrow$, akkor mivel $\Sigma \vdash (\downarrow \rightarrow F)$ (ezt látjuk majd gyakorlaton), ezekből MP-vel $\Sigma \vdash F$ minden F formulára.

iii) $\rightarrow$ i) Ha minden F levezethető, akkor pl. $\downarrow$ és $\downarrow \rightarrow \downarrow$ is

A Hilbert rendszer teljessége: maximális H-konzisztencia

meg még egy def

A bizonyítás lényege:

- kiindulunk egy H-konzisztens Σ formulahalmazból
- kibővítjük egy (potenciálisan) sokkal nagyobb Σ' H-konzisztens halmazzá
- erről a Σ' -ről megmutatjuk, hogy kielégíthető úgy, hogy adunk hozzá egy kielégítő értékadást $\Rightarrow$ mivel $\Sigma \subseteq \Sigma'$, Σ is kielégíthető lesz

Maximális H-konzisztens halmazok

Egy Σ formulahalmazt **maximális H-konzisztens**nek nevezünk, ha

- Σ H-konzisztens és
- minden $F \notin \Sigma$ -ra $\Sigma \cup \{F\}$ már nem H-konzisztens.

belőle még nem vezethető le $\downarrow$, de ha bárkit hozzáveszünk, már az lesz

Minden Σ H-konzisztens halmazhoz van $\Sigma' \supseteq \Sigma$ **maximális** H-konzisztens halmaz.
minden ilyen Σ kibővíthető úgy h maximális legyen

A Hilbert-rendszer teljessége: maximális H-konzisztencia

Minden H-konzisztens Σ kibővíthető maximálissá: Bizonyítás

- Legyen $F_1, F_2, F_3, \dots$ az összes formula egy felsorolása. megszámlálható sok van
- Legyen $\Sigma_0 := \Sigma$.
- Ha $\Sigma_i \cup \{F_{i+1}\}$ H-konzisztens, akkor legyen $\Sigma_{i+1} := \Sigma_i \cup \{F_{i+1}\}$.
- Egyébként legyen $\Sigma_{i+1} := \Sigma_i$.
- Azt állítjuk, hogy $\Sigma' := \bigcup_{i \geq 0} \Sigma_i$ egy maximális H-konzisztens halmaz.
Note: $\Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \dots$, ezért ha $\Gamma \subseteq \Sigma'$ véges, akkor van olyan i , melyre $\Gamma \subseteq \Sigma_i$
- Σ' H-konzisztens, mert:
 - $\Sigma_0 = \Sigma$ H-konzisztens;
 - ha Σ_i H-konzisztens, akkor Σ_{i+1} a sorozat definíciója szerint az
 - tehát mindegyik Σ_i H-konzisztens
 - ha $\Sigma' \vdash \downarrow$, akkor (mivel Σ' -nek csak véges sok elemét használjuk egy levezetésben) $\Sigma_i \vdash \downarrow$ valamilyen i -re, ami nem lehet, hiszen mindegyik Σ_i H-konzisztens.
- Σ' maximális, mert ha $F_i \notin \Sigma'$, akkor $\Sigma_{i-1} \cup \{F_i\} \vdash \downarrow$, tehát $\Sigma' \cup \{F_i\} \vdash \downarrow$.

A Hilbert-rendszer teljessége: H-konzisztencia

Ha Σ maximális H-konzisztens halmaz, akkor tetszőleges F formulára vagy $F \in \Sigma$, vagy $(F \rightarrow \downarrow) \in \Sigma$, de nem mindkettő.

egy maximális H-konzisztens halmazban minden formulára vagy ő, vagy a negáltja szerepel a halmazban

Bizonyítás

- Ha $F, F \rightarrow \downarrow \in \Sigma$, akkor belőlük MP-vel $\Sigma \vdash \downarrow$, ami ellentmond a H-konzisztenciának. tehát „nem mindkettő” done
- Tegyük fel, hogy se F , se $F \rightarrow \downarrow$ nincs Σ -ban. indirekt
 - Mivel Σ **maximális**, ezért tehát $\Sigma \cup \{F\} \vdash \downarrow$ és $\Sigma \cup \{F \rightarrow \downarrow\} \vdash \downarrow$.
 - Akkor a **dedukciós tétel** szerint:
 - $\Sigma \vdash F \rightarrow \downarrow$
 - $\Sigma \vdash (F \rightarrow \downarrow) \rightarrow \downarrow$
 - $\Sigma \vdash \downarrow$ (MP-vel ebből a kettőből)

ami ellentmond annak, hogy Σ H-konzisztens.

A Hilbert-rendszer teljessége: H-konzisztencia

Tetszőleges Σ formulahalmaz pontosan akkor kielégíthető, ha H-konzisztens.

- Ha Σ kielégíthető, akkor $\Sigma \not\vdash$, tehát $\Sigma \not\vdash$ (a Hilbert-rendszer helyessége miatt). egyik irány kész is
- Legyen Σ egy H-konzisztens formulahalmaz. be akarjuk bizonyítani, hogy kielégíthető
- Akkor Σ kibővíthető egy $\Sigma' \supseteq \Sigma$ maximális H-konzisztens halmazzá. lehet, hogy többféleképp is, válasszunk egyet
- Ebben minden p változó (mint formula) vagy szerepel, vagy nem (akkor pedig $p \rightarrow \perp$ szerepel benne). ezért bővítettük ki maximálissá
- Legyen $\mathcal{A}$ a következő értéktáblázat:

$$\mathcal{A}(p) = 1 \quad \Leftrightarrow \quad p \in \Sigma'.$$

nincs még kész, most jön h ellenőrizzük: ez az értéktáblázat kielégíti Σ' -t

A Hilbert-rendszer teljessége

- Azt állítjuk, hogy $\mathcal{A} \models F$ pontosan akkor igaz F -re, ha $F \in \Sigma'$.
- Ezt F felépítése szerinti indukcióval látjuk be.
- Ha $F = \perp$, akkor $\mathcal{A} \not\models \perp$ és $\perp \notin \Sigma'$

első rész minden $\mathcal{A}$ -ra igaz, második mert Σ' H-konzisztens

- Ha $F = p$ változó, akkor az állítás $\mathcal{A}$ definíciója miatt igaz.
- Ha $F = (F_1 \rightarrow F_2) \in \Sigma'$, akkor:

- **indirekten** tegyük fel, hogy $\mathcal{A} \not\models (F_1 \rightarrow F_2)$.
- Ekkor $\mathcal{A} \models F_1$ és $\mathcal{A} \not\models F_2$.
- Az **indukciós feltevés szerint** $F_1 \in \Sigma'$ és $F_2 \notin \Sigma'$.
- Tehát mivel Σ' maximális, $F_2 \rightarrow \perp \in \Sigma'$.

$\rightarrow$ szemantikája miatt

- Ekkor $F_1, F_1 \rightarrow F_2, F_2, F_2 \rightarrow \perp, \perp$ egy Σ' fölötti levezetése $\perp$ -nak

ezért bővítettük ki Σ -t

$\in, \in, \text{MP}, \in, \text{MP}$ sorrendben

- Ami ellentmond annak, hogy Σ' H-konzisztens!

tehát ha $F_1 \rightarrow F_2 \in \Sigma'$, akkor $\mathcal{A} \models F_1 \rightarrow F_2$

A Hilbert-rendszer teljessége

- Ha $F = (F_1 \rightarrow F_2) \notin \Sigma'$, akkor:
 - Mivel Σ' **maximális**, ezért $(F_1 \rightarrow F_2) \rightarrow \downarrow \in \Sigma'$.
 - **Indirekten** tegyük fel, hogy $\mathcal{A} \models (F_1 \rightarrow F_2)$.
 - Tehát $\mathcal{A} \not\models F_1$ vagy $\mathcal{A} \models F_2$ (vagy mindkettő). → szemantikája
 - Ha $\mathcal{A} \models F_2$:
 - az indukciós feltevés szerint $F_2 \in \Sigma'$.
 - Akkor Σ' nem H-konzisztens, mert
 $F_2, F_2 \rightarrow (F_1 \rightarrow F_2), F_1 \rightarrow F_2, (F_1 \rightarrow F_2) \rightarrow \downarrow, \downarrow$ egy Σ' fölötti levezetése $\downarrow$ -nak.
 - Ha $\mathcal{A} \not\models F_1$:
 - Az indukciós feltevés szerint tehát $(F_1 \rightarrow \downarrow) \in \Sigma'$.
 - De $\Sigma' \vdash (\downarrow \rightarrow F_2)$ is igaz. mert $\downarrow \rightarrow F$ alakú
 - Tehát $\Sigma' \vdash F_1 \rightarrow \downarrow$ és $\Sigma' \vdash \downarrow \rightarrow F_2$, emiatt $\Sigma' \vdash F_1 \rightarrow F_2$ az egyik korábbi példa szerint
volt: $\Sigma \vdash F \rightarrow G, \Sigma \vdash G \rightarrow H \Rightarrow \Sigma \vdash F \rightarrow H$
 - Mivel $(F_1 \rightarrow F_2) \rightarrow \downarrow \in \Sigma'$, így $\Sigma' \vdash \downarrow$, ez az ág is ellentmond a H-konzisztenciának.

Tehát ha $(F_1 \rightarrow F_2) \notin \Sigma'$, akkor $\mathcal{A} \not\models F$ és ezzel az állítást igazoltuk.

A Hilbert-rendszer teljessége

Mondjuk ki az állítást még egyszer:

Tetszőleges Σ formulahalmaz pontosan akkor kielégíthető, ha H-konzisztens.

Ennek következménye:

A Hilbert-rendszer helyességi és teljességi tétele

Tetszőleges Σ formulahalmazra és F formulára

$$\Sigma \models F \quad \Leftrightarrow \quad \Sigma \vdash F.$$

$$\Sigma \models F \Leftrightarrow \Sigma \cup \{F \rightarrow \perp\} \models \perp$$

$$\Leftrightarrow \Sigma \cup \{F \rightarrow \perp\} \vdash \perp$$

$$\Leftrightarrow \Sigma \vdash (F \rightarrow \perp) \rightarrow \perp$$

$$\Leftrightarrow \Sigma \vdash F$$

következmény vs kielégíthetetlenség

előző állítás

dedukciós tétel

ezt mindjárt.

A Hilbert-rendszer teljessége

$$\Sigma \vdash (F \rightarrow \downarrow) \rightarrow \downarrow \Leftrightarrow \Sigma \vdash F$$

Elég belátnunk, hogy $\{(F \rightarrow \downarrow) \rightarrow \downarrow\} \vdash F$ és $\{F\} \vdash (F \rightarrow \downarrow) \rightarrow \downarrow$.

- $\{(F \rightarrow \downarrow) \rightarrow \downarrow\} \vdash F$:
 - dedukciós tétel szerint ez pontosan akkor igaz, ha $\vdash ((F \rightarrow \downarrow) \rightarrow \downarrow) \rightarrow F$
 - ami igaz (Ax3)
- $\{F\} \vdash (F \rightarrow \downarrow) \rightarrow \downarrow$:
 - dedukciós tétel szerint ez pontosan akkor igaz, ha $\{F, F \rightarrow \downarrow\} \vdash \downarrow$
 - ami igaz ($\in$, $\in$, MP)

Ezzel a Hilbert-rendszer teljességét beláttuk: tetszőleges Σ halmazból Hilbert rendszerében **pontosan** Σ következményei vezethetők le.

done, easy

A kompaktsági tétel

A bizonyításban most sehol nem használtuk, hogy Σ véges-e!

Egy Hilbert-rendszerbeli $\Sigma \models F$ bizonyításban viszont csak véges sok Σ -beli formulát használunk.

Tehát:

Az ítéletkalkulus kompaktsági tétele

Tetszőleges Σ formulahalmaznak pontosan akkor logikai következménye egy F formula, ha már egy **véges** részhalmazának is következménye.

A tételnek egy másik, ekvivalens alakja:

Az ítéletkalkulus kompaktsági tétele

Egy Σ formulahalmaz pontosan akkor kielégíthetetlen, ha van **véges** kielégíthetetlen részhalmaza.

A kompaktsági tétel

A két alak tényleg ekvivalens:

- A tétel első alakjából következik a második:

$$\begin{aligned}\Sigma \text{ kielégíthetetlen} &\Leftrightarrow \Sigma \models \perp && \text{(ezt tudjuk)} \\ &\Leftrightarrow \text{van véges } \Sigma_0 \subseteq \Sigma, \text{ melyre } \Sigma_0 \models \perp && \text{első alak} \\ &\Leftrightarrow \text{van véges kielégíthetetlen } \Sigma_0 \subseteq \Sigma.\end{aligned}$$

- A tétel második alakjából következik az első:

$$\begin{aligned}\Sigma \models F &\Leftrightarrow \Sigma \cup \{\neg F\} \models \perp && \text{(ezt tudjuk)} \\ &\Leftrightarrow \text{van véges kielégíthetetlen } \Sigma_0 \subseteq \Sigma \cup \{\neg F\} && \text{második alak} \\ &\Leftrightarrow \text{van véges } \Sigma_0 \subseteq \Sigma, \text{ melyre } \Sigma_0 \cup \{\neg F\} \models \perp && F\text{-et külön vesszük} \\ &\Leftrightarrow \text{van véges } \Sigma_0 \subseteq \Sigma, \text{ melyre } \Sigma_0 \models F && \text{indirekt következtetés}\end{aligned}$$

A kompaktsági tétel haszna

A kompaktsági tétel egy újabb alakja:

Az ítéletkalkulus kompaktsági tétele

Egy Σ formulahalmaz pontosan akkor kielégíthető, ha **minden véges** részhalmaza kielégíthető.

A tétel **haszna**

ha van

- egy végtelen Σ formulahalmazunk (nemsokára látunk ilyeneket), melynek az elemeit egy algoritmus **generálja**
- és egy olyan következtető algoritmusunk, ami input **véges** Σ_0 -ra és F -re el tudja dönteni véges idő alatt, hogy igaz-e $\Sigma_0 \models F$,

akkor van olyan algoritmusunk, ami inputként kap egy (akár végtelen) Σ -t (ha végtelen, akkor egy generátor algoritmust kap), és egy F formulát, és

- ha $\Sigma \models F$, akkor ezt véges időn belül mindenképp megmutatja
- ha $\Sigma \not\models F$, akkor végtelen ciklusba esik

A kompaktsági tétel haszna

- Később (számítud, bonyelm) az ilyen Σ halmazokat, melyre van az elemeit generáló algoritmus, **rekurzívan felsorolható** halmaznak fogjuk hívni
- Az olyan problémákat, melyekre van olyan A algoritmus, hogy tetszőleges I inputra
 - ha I -re a válasz „igen” kellene legyen, akkor A az I -n futtatva „igen”-t mond,
 - ha I -re a válasz „nem” kellene legyen, akkor A az I -n futtatva végtelen ciklusba esik,pedig **félíg eldönthető** problémáknak fogjuk hívni.

Tehát az előző dián szereplő állítás röviden:

Félíg eldönthető a következő probléma:

- Input: egy rekurzívan felsorolható Σ formulahalmaz és egy F formula
- Output: igaz-e $\Sigma \models F$?

Elsőrendű logika

Informálisan:

- Ítéletkalkulusban a változók a $\{0, 1\}$ halmazból vettek fel értékeket, ezeket a logikai konnektívákkal kapcsoltuk össze
- Elsőrendű logikában, avagy **predikátumkalkulusban** viszont
 - a változók objektumoknak egy **univerzumából** vagy alaphalmazából veszik fel az értékeiket; (pl. természetes számok, stringek, stb.)
 - az objektumokat **függvények** transzformálhatják más objektumokká; (pl. összeadás, stringek összefűzése stb.)
 - az objektumokból **predikátumok** képeznek igazságértéket (pl. páros-e a szám, alfanumerikus-e a string, stb.)
 - és a változókat **kvantálni** is lehet (minden számnál van nagyobb, van üres string, stb.)
- A **kiértékelés** sokkal komplexebbé válik (nem is mindig létezik rá algoritmus, ha az alaphalmaz végtelen)
- Ezért megint a **kielégíthetetlenség**re fogunk nézni (félígeldöntő) algoritmusokat

Szintaxis

- **Elsőrendű változók**, vagy individuum változók, mostantól röviden csak „változók”: $x, y, z, \dots, x_1, y_5, \dots$
- **Függvényjelek**: $f, g, \dots, f_1, g_5, \dots$
- **Predikátumjelek**: $p, q, r, \dots, p_1, q_5, \dots$
- **Konnektívák**: $\wedge, \vee, \rightarrow, \leftrightarrow, \neg$
- **Kvantorok**: $\forall, \exists$
- **Logikai konstansjelek**: $\uparrow, \downarrow$

Használunk még zárójeleket és vesszőket az egyértelmű olvashatóság kedvéért.

- Minden függvényjelnek és predikátumjelnek van egy **aritása**, avagy rangja, változószáma. Ha f egy n -változós jel, ezt f/n jelöli.
- A 0-aritású függvényjeleket **konstans**jelnek nevezzük.
- A 0-aritású predikátumjeleket **logikai változóknak**.

Minden jelből **megszámlálható** sok van: ez annyit jelent, hogy algoritmikusan generálni tudjuk az első, második, $\dots$ (n -változós) jelet.

Szintaxis: termek

Elsőrendű logikában két szintaktikus kategória van: a **termek** és a **formulák** halmaza. (Kiértékeléskor a termek vesznek fel objektumot értéként, a formulák pedig igazságértéket.)

A termek

- Minden változó term.
- Ha f/n függvényjel, $t_1, \dots, t_n$ pedig termek, akkor $f(t_1, \dots, t_n)$ is term.
- Más term nincs.

Például ha x, y változók, $c/0$, $f/1$ és $g/2$ függvényjelek, akkor $g(f(x), g(c(), y))$ term.

Ha $c/0$ konstansjel, akkor $c()$ helyett csak c -t írunk. A $p/0$ -nál is $p()$ helyett csak p -t.

Szintaxis: formulák

A formulák

- Ha p/n **predikátum**jel, $t_1, \dots, t_n$ pedig termék, akkor $p(t_1, \dots, t_n)$ egy (atomi) formula.
- Ha F formula, akkor $\neg F$ is az.
- Ha F és G formulák, akkor $F \vee G$, $F \wedge G$, $F \rightarrow G$, $F \leftrightarrow G$ is az.
- $\downarrow$ és $\uparrow$ is formulák.
- Ha F formula és x változó, akkor $\exists xF$ és $\forall xF$ is formulák.
- Más formula nincs.

(Az egyértelmű olvashatóság kedvéért itt is zárójelezünk.)

Például ha $p/1$ és $q/2$ predikátumjelek és $f/1$ függvényjel, akkor

$$\exists x(p(x) \wedge \forall yq(f(x), y))$$

egy formula.

Szemantika: struktúrák

Egy elsőrendű formulát egy **struktúrában** értékelünk ki.

Struktúra

Egy $\mathcal{A} = (A, I, \varphi)$ hármas, ahol

- A egy nemüres halmaz, az univerzum, vagy alaphalmaz az objektumoké
- φ a változóknak egy „default” értékadása, minden x változóhoz egy $\varphi(x) \in A$ objektumot rendel
- I az **interpretációs függvény**, ez rendel a függvény- és predikátumjelekhez szemantikát, „értelmet” az adott struktúrában:
 - ha f/n függvényjel, akkor $I(f)$ egy $A^n \rightarrow A$ függvény;
 - ha p/n predikátumjel, akkor $I(p)$ egy $A^n \rightarrow \{0, 1\}$ predikátum (vagy „reláció”).
nem pedig „interpolációs” ffs

Erre van egy megkötésünk:

Az $=$ bináris predikátumjelet minden struktúrában ténylegesen az egyenlőséggel kell interpretálnunk!

Ha t egy term, $\mathcal{A} = (A, I, \varphi)$ pedig egy struktúra, akkor t értéke $\mathcal{A}$ -ban egy A -beli objektum lesz, melyet $\mathcal{A}(t)$ -vel is jelölünk és felépítés szerinti indukcióval definiálunk:

A t term értéke az $\mathcal{A}$ struktúrában, $\mathcal{A}(t)$

- Ha $t = x$ változó, akkor $\mathcal{A}(t) = \varphi(x)$ (tehát: a változók értékét φ szabja meg)
- Ha $t = f(t_1, \dots, t_n)$, akkor $\mathcal{A}(t) = I(f)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n))$
(tehát: rekurzívan kiértékeljük a $t_1, \dots, t_n$ termeket a struktúrában, kapjuk az $a_1, \dots, a_n \in A$ objektumokat; ezeket behelyettesítjük az $I(f)$ függvénybe, amit ebben a struktúrában az f jel jelöl.)

Szemantika: termék kiértékelése

Például: ha $+ / 2$, $\times / 2$, $' / 1$ függvényjelek, $0 / 0$ és $1 / 0$ konstansjelek, x és y pedig változók, akkor egy struktúra lehet pl. $\mathcal{A} = (\mathbb{N}_0, I, \varphi)$, ahol

- $\mathbb{N}_0 = \{0, 1, 2, \dots\}$ a természetes számok halmaza;
- $\varphi(x) = 2$, $\varphi(y) = 3$, $\varphi(z) = 6, \dots$
- $I(+)$, $I(\times)$ és $I(')$ rendre az összeadás, szorzás és az $n \mapsto n + 1$ rákövetkezés függvények,
- $I(0) = 0$ és $I(1) = 1$.

Ha „ismert” bináris függvény- vagy predikátumjelekkel van dolgunk a példákban, azokat **infix** is írjuk a könnyebb olvashatóság kedvéért, pl. $+(x, y)$ helyett $x + y$ -t. Akkor pl.

- $\mathcal{A}((x + 1) \times y) = 9$, hiszen $\mathcal{A}(x) = 2$, $\mathcal{A}(1) = 1$, akkor $\mathcal{A}(x + 1) = I(+)(2, 1) = 2 + 1 = 3$, $\mathcal{A}(y) = 3$ és $\mathcal{A}((x + 1) \times y) = I(\times)(3, 3) = 9$.
- $\mathcal{A}((x \times y) + 0) = 6$,
- $\mathcal{A}((0 + 1) \times 1) = 1$.

Szemantika: termek kiértékelése

Általában is elég csak a ténylegesen használt változók értékét specifikálnunk φ -ben:

Ha t term, $\mathcal{A} = (A, I, \varphi)$ és $\mathcal{A}' = (A, I, \varphi')$ pedig olyan struktúrák, melyekre minden t -beli x -re $\varphi(x) = \varphi'(x)$, akkor $\mathcal{A}(t) = \mathcal{A}'(t)$.

Bizonyítás: t felépítése szerinti indukcióval

- Ha $t = x$: ekkor $\mathcal{A}(t) = \varphi(x) = \varphi'(x) = \mathcal{A}'(t)$
- Ha $t = f(t_1, \dots, t_n)$: ekkor minden i -re t_i összes x változójára $\varphi(x) = \varphi'(x)$ is igaz. Így az indukciós feltevés szerint $\mathcal{A}(t_i) = \mathcal{A}'(t_i)$ minden i -re, és ekkor

$$\begin{aligned}\mathcal{A}(t) &= I(f)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)) \\ &= I(f)(\mathcal{A}'(t_1), \dots, \mathcal{A}'(t_n)) \\ &= \mathcal{A}'(t).\end{aligned}$$

Ezért ha t -ben nincs változó, úgy a φ megadása is felesleges, ilyenkor csak $\mathcal{A} = (A, I)$ -nek írjuk a struktúrát.

Pl. az előbb $(0+1) \times 1$ egy változómentes term volt; ezeket a termeket **alaptermeknek** vagy ground termeknek nevezzük.

alapterm

Azokat a termeket, melyek nem tartalmaznak változót, **alaptermeknek** nevezzük.

Szemantika: formulák kiértékelése

A két kvantor kiértékelésének definíciójához szükségünk lesz a következő jelölésre:

Ha $\mathcal{A} = (A, I, \varphi)$ struktúra, x változó és $a \in A$ objektum, akkor $\mathcal{A}_{[x \mapsto a]}$ jelöli azt az (A, I, φ') struktúrát, melyben

$$\varphi'(y) := \begin{cases} a & \text{ha } y = x, \\ \varphi(y) & \text{különben.} \end{cases}$$

Vagyis: $\mathcal{A}_{[x \mapsto a]}$ -t úgy kapjuk $\mathcal{A}$ -ból, hogy benne az x változó default értékét a -ra változtatjuk, mást nem változtatunk meg.

Ha több változónak is beállítunk értéket, akkor $\mathcal{A}_{[x \mapsto a][y \mapsto b]}$ helyett $\mathcal{A}_{[x \mapsto a, y \mapsto b]}$ -t írunk, hogy olvasható maradjon.

A sorrend számít, ha a változók közt van átfedés! $\mathcal{A}_{[x \mapsto 2, x \mapsto 3]}$ -ban x default értéke 3 lesz.

Formula értéke struktúrában

Ha F formula, $\mathcal{A} = (A, I, \varphi)$ pedig struktúra, akkor az F értéke $\mathcal{A}$ -ban egy igazságérték, amit $\mathcal{A}(F)$ jelöl és az F felépítése szerinti indukcióval adunk meg:

- Logikai konstansok: $\mathcal{A}(\uparrow) = 1$, $\mathcal{A}(\downarrow) = 0$
- Konnektívák: $\mathcal{A}(F \wedge G) = \mathcal{A}(F) \wedge \mathcal{A}(G)$, $\mathcal{A}(\neg F) = \neg \mathcal{A}(F)$, ...
- Atomi formulák:

$$\mathcal{A}(p(t_1, \dots, t_n)) := I(p)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)).$$

Vagyis: $\mathcal{A}$ -ban először kiértékeljük a $t_1, \dots, t_n$ termeket, kapjuk az $a_1, \dots, a_n$ objektumokat és ezeket behelyettesítjük abba a predikátumba (és kapunk egy igazságértéket), amit ebben a struktúrában a p jelöl.

Formula értéke strukturában, tovább

- Kvantorok:

$$\mathcal{A}(\exists x F) := \begin{cases} 1 & \text{ha van olyan } a \in A, \text{ melyre } \mathcal{A}_{[x \mapsto a]}(F) = 1; \\ 0 & \text{különben.} \end{cases}$$

Tehát: akkor igaz, ha x értékét be tudjuk állítani úgy $\mathcal{A}$ -ban, hogy a megváltoztatott struktúra kielégítse F -et.

$$\mathcal{A}(\forall x F) := \begin{cases} 1 & \text{ha minden } a \in A\text{-ra igaz, hogy } \mathcal{A}_{[x \mapsto a]}(F) = 1; \\ 0 & \text{különben.} \end{cases}$$

Tehát: akkor igaz, ha x értékét bármire is állítjuk be $\mathcal{A}$ -ban, a megváltoztatott struktúra kielégíti F -et.

Formulák kiértékelése, példa

- Legyen $\mathcal{A} = (\mathbb{N}_0, I, \varphi)$ a természetes számok szokásos struktúrája: $I(0) = 0$, $I(1) = 1$, $I(+)$ összeadás, stb
- $I(<)$ a szokásos „kisebb” reláció
- $I(=)$ az egyenlőség (más nem is lehet)
- $\varphi(x) = 5$, $\varphi(y) = 6$

akkor pl. $\mathcal{A}$ -ban

- $\mathcal{A}(x < y) = 1$, hiszen $\mathcal{A}(x) = 5$, $\mathcal{A}(y) = 6$ és $I(<)(5, 6)$ igaz
- $\mathcal{A}(\exists z(y < z))$ igaz, hiszen pl. az $\mathcal{A}_{[z \mapsto 20]}(y < z)$ igaz lesz: $6 < 20$
- $\mathcal{A}(\exists x(x < x))$ hamis
- $\mathcal{A}(\forall x(y < x))$ hamis, hiszen pl. $\mathcal{A}_{[x \mapsto 5]}(x < y)$ hamis ($5 < 6$ nem igaz)

$\mathcal{A}$ -ban...

- $\exists x(1 < x \wedge x + x = z)$ pontosan akkor igaz, ha $\varphi(z)$ egy 2-nél nagyobb páros szám, jelölje ezt $\text{Even}(z)$
- $\exists x(x \times x = z)$ pontosan akkor igaz, ha $\varphi(z)$ négyzetszám
- $\exists x(x \times y = z)$ pontosan akkor igaz, ha $\varphi(y)$ osztja $\varphi(z)$ -t, jelölje ezt a formulát $y|z$
- $\neg\exists x(1 < x \wedge x < y \wedge x|y)$ pontosan akkor igaz, ha $\varphi(y)$ prímszám, jelölje ezt $\text{Prime}(y)$

$$\mathcal{A}\left(\forall x(\text{Even}(x) \rightarrow \exists z_1\exists z_2(\text{Prime}(z_1) \wedge \text{Prime}(z_2) \wedge z_1 + z_2 = x))\right) = ?$$

Formulák kiértékelése, példa

„minden 2-nél nagyobb páros szám felírható két prímszám összegeként”

Ez az ún. **Goldbach-sejtés**, nem ismert, hogy igaz-e.

- De miért nem értékeljük egyszerűen ki ezt a formulát, hogy megtudjuk?
- Mert **nem lehet** automatikusan: pl. a természetes számok szokásos struktúrájában (ha van összeadás, szorzás, mondjuk 0, 1, és mondjuk a $<$ rendezés) **nincs** olyan algoritmus, mely ki tudna értékelni tetszőleges input formulát.
- (Itt a „nincs” azt jelenti: matematikailag be van bizonyítva, hogy ilyen algoritmus nem létezik.)
- Fact: ha csak összeadást használunk, akkor arra van ilyen algoritmus. (Ezt hívják **Presburger aritmetikának**.)
- Fact: ha a természetes számok helyett a valós számokat vesszük az alaphalmaznak, akkor arra is van. (ez meg a **Tarski-Seidenberg tétel**).
- Open: ha a valós számokon még bevesszük függvénynek az $x \mapsto e^x$ függvényt is, nem tudjuk, hogy arra van-e kiértékelő algoritmus.

Prenex alakú formulák kiértékelése, játékelmélet

Legyen $F = Q_1x_1Q_2x_2 \dots Q_nx_nF'$ egy **prenex alakú** formula, $\mathcal{A}$ pedig egy struktúra.

Két egymással versenyző játékos, $\exists$ és $\forall$ játssza F -en és $\mathcal{A}$ -n a következő játékot:

- a játék n körig tart
- az i . körben x_i értékét állítjuk be
- ha $Q_i = \exists$, akkor $\exists$ mondja meg x_i értékét
- ha $Q_i = \forall$, akkor pedig $\forall$
- miután lement az n kör, kiértékeljük F' -t az így kialakult $\mathcal{A}'$ struktúrában
azaz $\mathcal{A}' = \mathcal{A}_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n]}$ -ben, ha x_i értéke a_i
- ha igaz lett, $\exists$ nyer, ha hamis, $\forall$ nyer.

A játékban valamelyik játékosnak van nyerő stratégiája.

$\mathcal{A} \models F$ pontosan akkor igaz, ha a fenti játékot F -en és $\mathcal{A}$ -n játszva $\exists$ -nek van nyerő stratégiája.

Ehrenfeucht-Fraïssé játék, példa

Legyen $\mathcal{N}$ a természetes számok standard struktúrája.

- $\mathcal{N} \models \forall x \exists y (x < y)$: ha először $\forall$ választja ki x -et, $\exists$ választhatja y -nak az eggyel nagyobb számot és így kihozza igazra a formula magját
- $\mathcal{N} \not\models \exists y \forall x (x < y)$: ha először $\exists$ választja ki y -t, akkor utána $\forall$ választhatja pl. az y -nál eggyel nagyobb számot x -nek és így kihozza hamisra a formula magját

Formulák kielégíthetlensége

Mint ítéletkalkulusban is, $\mathcal{A}(F) = 1$ -et úgy is írjuk, hogy $\mathcal{A} \models F$, vagy $\mathcal{A} \in \text{Mod}(F)$, szóban „ $\mathcal{A}$ modellje F -nek”.

Ismét, $\Sigma \models F$ a $\text{Mod}(\Sigma) \subseteq \text{Mod}(F)$ logikai következménynek a jele.

Tehát: már nagyon „egyszerű” $\mathcal{A}$ struktúrákra is reménytelenül nehéz lehet eldönteni, hogy $\mathcal{A} \models F$ igaz-e egy input F formulára vagy sem.

Ezért a következő stratégiát szoktuk követni:

- Leírunk néhány formulát egy Σ halmazba, melyre $\mathcal{A} \in \text{Mod}(\Sigma)$ (azaz Σ -ba felveszünk olyan formulákat, melyeket $\mathcal{A}$ kielégít)
- Megpróbáljuk bebizonyítani, hogy $\Sigma \models F$.
- Ha igen, akkor $\mathcal{A} \in \text{Mod}(\Sigma) \subseteq \text{Mod}(F)$ miatt $\mathcal{A} \models F$ is igaz.
- Ha nem... hát akkor vagy $\Sigma \models \neg F$ -et próbáljuk igazolni, vagy bővítjük még Σ -t, hátha van olyan tulajdonsága $\mathcal{A}$ -nak, melyet még nem vettünk figyelembe.

A Peano-axiómák

Pl. a természetes számok struktúrájában állítások bizonyítására a következő Σ formulahalmazból szoktunk kiindulni mint „axiómákból”:

- $\forall x(\neg(x' = 0))$
- $\forall x\forall y(x' = y' \rightarrow x = y)$
- $\forall x(x + 0 = x)$
- $\forall x\forall y(x + y' = (x + y)')$
- $\forall x(x \cdot 0 = 0)$
- $\forall x\forall y(x \cdot y' = x \cdot y + x)$
- $\forall x(x \leq 0 \rightarrow x = 0)$
- $\forall x\forall y(x \leq y' \rightarrow (x \leq y \vee x = y'))$
- $\forall x\forall y(x < y \vee x = y \vee y < x)$
- **Indukciós axióma séma:** $(F(0) \wedge \forall x(F(x) \rightarrow F(x')))) \rightarrow \forall xF(x)$, ahol $F(x)$ olyan formula, melyben legfeljebb az x változó fordul elő szabadon, és $F(0)$, $F(x')$ úgy állnak elő, hogy x helyébe 0-t ill. x' -t helyettesítünk (ld. később).

Az ítétekalkulusból ismert összefüggések a $\models$ -ra továbbra is igazak maradnak, így pl.:

- $\Sigma \models \perp$ pontosan akkor, ha Σ kielégíthetetlen;
- $\Sigma \models F$ pontosan akkor, ha $\Sigma \cup \{\neg F\}$ kielégíthetetlen;
- ha $\Sigma \models F$ és $\Sigma \subseteq \Gamma$, akkor $\Gamma \models F$;

ezért ha a $\Sigma \models F$ logikai következményt akarjuk bizonyítani, akkor azt megtehetjük úgy, hogy a $\Sigma \cup \{\neg F\}$ formulahalmaz (egy részhalmazáról) mutatjuk meg, hogy kielégíthetetlen.

Erre fogunk (félígeldöntő) algoritmusokat nézni.

Szabad és kötött változóelőfordulások

Szabad változók

Egy F formulában egy x változóelőfordulás szabad, ha nincs $\exists x$ kvantor hatáskörében; egyébként kötött, és a legbelső ilyen $\exists x$ kvantor köti.

$$\forall x \left(p(x, f(y)) \wedge p(x, z) \wedge \exists y \forall x p(x, y) \right)$$

A színes változóelőfordulások kötöttek, a fekete változóelőfordulások szabadok, a fekete nem szín.

Kötött változók default értékadása nem számít

Mint a termék kiértékelésénél is, a formulakénál sem feltétlen kell teljesen specifikálnunk a φ default értékadást: elég csak azokat a változókat megadni, melyek előfordulnak a formulában **szabadon**.

Ha $\mathcal{A} = (A, I, \varphi)$ és $\mathcal{A}' = (A, I, \varphi')$ struktúrák és F formula úgy, hogy minden F -ben **szabadon előforduló** x változóra $\varphi(x) = \varphi'(x)$, akkor $\mathcal{A}(F) = \mathcal{A}'(F)$.

F felépítése szerinti indukcióval

- Ha $F = \uparrow$, akkor mindkét oldal 1, ha $\downarrow$, mindkét oldal 0
- Ha $F = \neg G$, akkor F -ben és G -ben ugyanazok a változók fordulnak elő szabadon; az **indukciós feltevés** szerint $\mathcal{A}(F) = \neg \mathcal{A}(G) = \neg \mathcal{A}'(G) = \mathcal{A}'(\neg G)$.

Kötött változók default értékadása nem számít

Bizonyítás F felépítése szerinti indukcióval

- Ha pl. $F = G \vee H$, akkor a G -ben / H -ban szabadon szereplő változók szabadok F -ben is. Tehát ezekre a változókra φ és φ' megegyezik. Alkalmazva az **indukciós feltevést**:

$$\mathcal{A}(F) = \mathcal{A}(G) \vee \mathcal{A}(H) = \mathcal{A}'(G) \vee \mathcal{A}'(H) = \mathcal{A}'(F).$$

A többi bináris konnektívára is ugyanígy.

- Ha $F = p(t_1, \dots, t_n)$ atomi formula, akkor benne **minden változó szabad**, tehát φ és φ' megegyezik minden olyan változóra, aki bármelyik t_i -ben is szerepel. A termeknél már láttuk, hogy ekkor $\mathcal{A}(t_i) = \mathcal{A}'(t_i)$ minden i -re és így

$$\begin{aligned}\mathcal{A}(F) &= I(p)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)) \\ &= I(p)(\mathcal{A}'(t_1), \dots, \mathcal{A}'(t_n)) \\ &= \mathcal{A}'(F).\end{aligned}$$

Kötött változók default értékadása nem számít

Bizonyítás F felépítése szerinti indukcióval

- Ha $F = \exists xG$ vagy $F = \forall xG$, és F -ben a szabadon előforduló változók halmaza X , akkor G -ben vagy X , vagy $X \cup \{x\}$. (Attól függően, hogy G -ben van-e szabadon x vagy nincs.) Persze $x \notin X$.

Nézzük az $\mathcal{A}_{[x \mapsto a]}$ és az $\mathcal{A}'_{[x \mapsto a]}$ alakú struktúrákat! Ezek default értékadása megegyezik $X \cup \{x\}$ -en, hiszen

- X -en eleve megegyezik $\mathcal{A}$ -ban és $\mathcal{A}'$ -ben, az új struktúrákban meg az X -beli változók értéke ugyanaz marad;
- x -en mindkét struktúra default értéke a .

Tehát minden a -ra az **indukciós feltevés** szerint

$$\mathcal{A}_{[x \mapsto a]}(G) = \mathcal{A}'_{[x \mapsto a]}(G).$$

Amiből kapjuk, hogy $\mathcal{A}(F) = \mathcal{A}'(F)$.

Mondat

Egy formulát **mondatnak** vagy **zárt formulának** nevezünk, ha nem szerepel benne változó szabadon.

(Pl. a Peano axiómák mondatok voltak.)

Az előző állítás szerint ha egy F formula mondat, akkor hogy a $\mathcal{A} \models F$ fennáll-e, nem függ az $\mathcal{A} = (A, I, \varphi)$ struktúra φ -jétől.

Ezért ilyenkor a struktúrákat sokszor csak $\mathcal{A} = (A, I)$ formában írjuk.

Ezért beszéltünk az előbb „a” természetes számok struktúrájáról: lehetne a φ változtatásával sok különböző struktúrát kapni, de ha mondatokat értékelünk ki, a φ nem számít.

Az ítéletkalkulus az elsőrendű logikában

Nem véletlenül jelöljük a predikátumjeleket ugyanúgy p, q, r -rel, mint korábban az ítéletváltozókat, ugyanis

Az ítéletkalkulus az elsőrendű logikának az a speciális esete, amikor minden predikátumjel 0-változós.

Hiszen ekkor

- az atomi formulák $p()$, tehát egyszerűen p alakúak;
- a termeket nincs hova behelyettesíteni, így a függvényszimbólumok interpretációja nem lényeges, sőt az objektumok alaphalmaza sem az;
- így az individuumváltozók sem szerepelnek egy formulában sem;
- ha pedig x nem szerepel F -ben szabadon, úgy az előző állítás szerint $\forall xF$ és $\exists xF$ is ekvivalensek F -fel, tehát kvantorokat sem kell használnunk. **Itt fontos,**
hogy az univerzum mindig nemüres.

Tehát ebben az esetben pontosan azokat a formulákat kapjuk, mint ítéletkalkulusban; egy „értékadást” tekinthetünk mint egy struktúrát.

Mod, Th és Cons az elsőrendű logikában

- Ha Σ **mondatok** egy halmaza, akkor $\text{Mod}(\Sigma)$, Σ **modelljei** az olyan $\mathcal{A} = (A, I)$ alakú struktúrák osztálya, melyek Σ összes elemét kielégítik:

$$\text{Mod}(\Sigma) := \{ \mathcal{A} = (A, I) : \forall F \in \Sigma \mathcal{A} \models F \}$$

- Ha $\mathcal{K}$ az $\mathcal{A} = (A, I)$ alakú struktúrák egy osztálya, akkor $\text{Th}(\mathcal{K})$, $\mathcal{K}$ **elmélete** az összes olyan F **mondat** halmaza, mely igaz $\mathcal{K}$ minden elemében:

$$\text{Th}(\mathcal{K}) := \{ F : \forall \mathcal{A} \in \mathcal{K} \mathcal{A} \models F \}$$

- Ha Σ **mondatok** egy halmaza, akkor $\text{Cons}(\Sigma)$ a Σ összes **következményének** a halmaza (szintén mondatokat tartalmaz):

$$\text{Cons}(\Sigma) := \{ F : \Sigma \models F \}.$$

Minden, amit ítéletkalkulusra láttunk erre a három operátorra vonatkozóan, igaz marad elsőrendű logikára is (mert továbbra is a $\models$ által meghatározott Galois-kapcsolatról van szó).

- Általában arra vagyunk kíváncsiak, hogy egy F mondat igaz-e egy **konkrét** $\mathcal{A}$ struktúrában (mint pl. a természetes számokéban) vagy struktúrák egy **konkrét** $\mathcal{K}$ osztályában (mint pl. „igaz-e minden rendezett halmazra?”)
- Tehát, hogy $F \in \text{Th}(\mathcal{K})$ igaz-e, F benne van-e $\mathcal{K}$ elméletében.
 - csoportelmélet: $\mathcal{K}$ a csoportok osztálya
 - számelmélet vagy aritmetika: $\mathcal{A}$ a természetes számok struktúrája (összeadás, szorzás, rákövetkezés, rendezés)
 - halmazelmélet. . .
- Mivel a struktúrákban (pláne végtelen sokban) kiértékelni nehéz (nagyon sokszor algoritmikusan nem is lehetséges), ezért a kérdést nem így vizsgáljuk, hanem konstruálunk egy $\Sigma \subseteq \text{Th}(\mathcal{K})$ halmazt és a $\Sigma \models F$ kérdést próbáljuk eldönteni.
- Tehát azt, hogy $F \in \text{Cons}(\Sigma)$ igaz-e.

- Például a **számelmélet** kérdéseit sokszor úgy vizsgáljuk, hogy a kérdéses F mondat következik-e a Peano axiómákból.
- A legjobb eset az, amikor találunk egy olyan **véges** Σ mondathalmazt, melyre $\text{Mod}(\Sigma) = \mathcal{K}$, hiszen ekkor $\text{Cons}(\Sigma) = \text{Th}(\text{Mod}(\Sigma)) = \text{Th}(\mathcal{K})$, vagyis pontosan a Σ következményei adják a $\mathcal{K}$ elméletét.
- Ekkor azt mondjuk, hogy $\mathcal{K}$ **végesen axiomatizálható**, és Σ a $\mathcal{K}$ -nak egy véges **axiómarendszere**.
- Ha van olyan Σ , melyre $\text{Mod}(\Sigma) = \mathcal{K}$, de Σ nem feltétlenül véges, azt mondjuk, hogy $\mathcal{K}$ **gyengén axiomatizálható**. (Ez algoritmikus szempontból egy sokkal rosszabb helyzet.)
- Látni fogjuk, hogy pl. a természetes számok $\mathcal{N}$ struktúrája **még gyengén se axiomatizálható** (ez meg egy még rosszabb helyzet).

Egy példa

```
//@ requires n >= 0
//@ ensures \result == n*n
int wut( int n ) {
    int i = 0;
    int d = 1;
    int r = 0;
    //@ maintaining i*i == r &&
        i*2+1 == d
    while( i != n ){
        r = r + d;
        i = i + 1;
        d = d + 2;
    }
    return r;
}
```

Egy következtető rendszer

- levezeti a ciklusmagra:
 $\forall i \forall r \forall d (inv(i, r, d) \rightarrow inv(i + 1, r + d, d + 2))$, ahol
- $inv(i, r, d)$ az
 $i * i = r \wedge i * 2 + 1 = d$ formula
- $(i + 1) * (i + 1) = i * i + i * 2 + 1$,
 $(i + 1) * 2 + 1 = i * 2 + 1 + 2$,
Peano axiómákból kijön
- levezeti, hogy $inv(0, 0, 1)$ igaz a ciklusmagba belépéskor
- végül, a kilépéskor $i == n$,
tehát $r = n * n$ tényleg.

- Az elsőrendű logika esetére is szeretnénk **következtető rendszereket** építeni.
- Ehhez ismét **normálformákra** lesz szükségünk.
- A legegyszerűbb eset: ha a formulában nincs kvantor.

Ismét:

- **Literál:** atomi formula (ekkor pozitív) vagy negáltja (ekkor negatív), pl. $p(x, c)$, $\neg q(x, f(x), z)$
- **Klóz:** literálok véges diszjunkciója, pl. $p(x) \vee \neg q(y, c)$
- **CNF:** klózok konjunkciója, pl. $(p(x) \vee \neg q(y, c)) \wedge \neg p(x)$

Kvantormentes elsőrendű logikai formulát az ítéletkalkulusban megszokott módon hozhatunk CNF-re:

- Nyilak eliminálása
- Negálások bevitele deMorgan azonosságokkal $\Rightarrow$ NNF
- az NNF-ből CNF készítése disztributivitással

$$\left(p(x, f(x)) \vee q(y) \right) \leftrightarrow \left((p(x, y) \rightarrow q(x)) \wedge p(x, c) \right)$$

Nyilak eliminálása:

$$\begin{aligned} & \left(\neg \left(p(x, f(x)) \vee q(y) \right) \vee \left((\neg p(x, y) \vee q(x)) \wedge p(x, c) \right) \right) \\ & \wedge \left(\left(p(x, f(x)) \vee q(y) \right) \vee \neg \left((\neg p(x, y) \vee q(x)) \wedge p(x, c) \right) \right) \end{aligned}$$

CNF elsőrendben, példa

$$\left(\neg \left(p(x, f(x)) \vee q(y) \right) \vee \left((\neg p(x, y) \vee q(x)) \wedge p(x, c) \right) \right) \\ \wedge \left(\left(p(x, f(x)) \vee q(y) \right) \vee \neg \left((\neg p(x, y) \vee q(x)) \wedge p(x, c) \right) \right)$$

Negációk bevitele:

$$\left(\left(\neg p(x, f(x)) \wedge \neg q(y) \right) \vee \left((\neg p(x, y) \vee q(x)) \wedge p(x, c) \right) \right) \\ \wedge \left(\left(p(x, f(x)) \vee q(y) \right) \vee \left((p(x, y) \wedge \neg q(x)) \vee \neg p(x, c) \right) \right)$$

Egy atomi formula pontosan akkor lesz negatív, és egy $\wedge/\vee$ pontosan akkor vált a másikkra, ha **páratlan sok** negáció belsejében van.

$$\left(\left(\underline{\neg p(x, f(x))} \wedge \underline{\neg q(y)} \right) \vee \left(\underline{(\neg p(x, y) \vee q(x))} \wedge \underline{p(x, c)} \right) \right) \\ \wedge \left(\left(\underline{p(x, f(x))} \vee \underline{q(y)} \right) \vee \left(\underline{p(x, y)} \wedge \underline{\neg q(x)} \vee \underline{\neg p(x, c)} \right) \right)$$

(Klózok **pirossal**, CNF-ek **kékekkel** aláhúzva)

Disztributivitás:

$$\begin{aligned} & (\neg p(x, f(x)) \vee \neg p(x, y) \vee q(x)) \\ & \wedge (\neg p(x, f(x)) \vee p(x, c)) \\ & \wedge (\neg q(y) \vee \neg p(x, y) \vee q(x)) \\ & \wedge (\neg q(y) \vee p(x, c)) \\ & \wedge (p(x, f(x)) \vee q(y) \vee p(x, y) \vee \neg p(x, c)) \\ & \wedge (p(x, f(x)) \vee q(y) \vee \neg q(x) \vee \neg p(x, c)) \end{aligned}$$

Kielégíthetlenség def szerint

Elsőrendű formulák kielégíthetlenségének bizonyítására egy módszer lehet a **szemantika szerinti kifejtés**:

$$F = \forall x p(x) \wedge \exists y (p(y) \rightarrow q(f(y))) \wedge \forall z \neg q(z)$$

egy kielégíthetetlen formula, mert:

- Tegyük fel **indirekten**, hogy F kielégíthető.
- Akkor van modellje: legyen $\mathcal{A} = (A, I, \varphi)$ egy struktúra, melyre $\mathcal{A}(F) = 1$.
- Mivel F három formula konjunkciója, így
 - $\mathcal{A}(\forall x p(x)) = 1$
 - $\mathcal{A}(\exists y (p(y) \rightarrow q(f(y)))) = 1$
 - és $\mathcal{A}(\forall z \neg q(z)) = 1$.

nincs kész még

Kielégíthetetlenség def szerint

- $\mathcal{A}(\exists y(p(y) \rightarrow q(f(y)))) = 1$ szerint van olyan $a \in A$, melyre

$$\mathcal{A}_{[y \mapsto a]}(p(y) \rightarrow q(f(y))) = 1.$$

- Tehát erre az $a \in A$ -ra
 - vagy $\mathcal{A}_{[y \mapsto a]}(p(y)) = 0$, azaz $I(p)(a) = 0$,
 - vagy $\mathcal{A}_{[y \mapsto a]}(q(f(y))) = 1$, azaz $I(q)(I(f)(a)) = 1$

teljesül.

- Ugyanakkor, $\mathcal{A}(\forall x p(x)) = 1$ szerint minden $b \in A$ objektumra $\mathcal{A}_{[x \mapsto b]}(p(x)) = 1$, tehát $I(p)(b) = 1$.
- Speciálisan ha minden b -re, akkor $b := a$ -ra is $I(p)(a) = 1$, így $I(p)(a) \neq 0$.
- Emiatt $I(q)(I(f)(a)) = 1$ kell legyen.

nincs kész még

Kielégíthetetlenység def szerint

- Továbbá, $\mathcal{A}(\forall z \neg q(z)) = 1$ szerint minden $c \in A$ objektumra $\mathcal{A}_{[z \mapsto c]}(\neg q(z)) = 1$, tehát $\mathcal{A}_{[z \mapsto c]}(q(z)) = 0$, vagyis $I(q)(c) = 0$.
- Ha minden c -re, akkor $c := I(f)(a)$ -ra is, emiatt pedig $I(q)(I(f)(a)) = 0$, ami ellentmondás.

Tehát a formula kielégíthetetlen.

done, easy

Helyettesítés: értékadás szintaktikai szinten

- Az előző módszerben egy kényelmetlen részlet az **értékadás**.
- A következő algoritmusainkban **tisztán** formulákkal szeretnénk dolgozni, objektum-értékek és interpretációs függvények nélkül.
- Ezért definiáljuk a **helyettesítést**: ha x változó és t term, akkor
 - ha u term, akkor $u[x/t]$ egy term lesz, melyre

$$\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(u) = \mathcal{A}(u[x/t])$$

- ha pedig F formula, akkor $F[x/t]$ egy formula lesz, melyre

$$\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) = \mathcal{A}(F[x/t]).$$

Vagyis: az új term/formula értéke az eredeti struktúrában ugyanaz legyen, mintha a struktúrában elvégeztünk volna egy értékadást, és a megváltozott struktúrában értékelnénk ki az eredeti termet/formulát.

- Értékadás helyett helyettesítéssel dolgozva nem kell majd az eredeti struktúrához nyúlnunk.

Helyettesítés termekben

Hasonlóan a korábbiakhoz, a termék esete az egyszerűbb.

az $u[x/t]$ term

- Ha u és t termék, x pedig változó, akkor az $u[x/t]$ termet az u term felépítése szerinti **indukcióval** definiáljuk:
- Ha u az x változó, akkor $u[x/t] := t$.
- Ha $u = y \neq x$ egy másik változó, akkor $u[x/t] := u$.
- Ha $u = f(t_1, \dots, t_n)$, akkor

$$u[x/t] := f(t_1[x/t], \dots, t_n[x/t]).$$

(Vagyis: végezzük el a helyettesítést rekurzívan az összes résztermjében.)

Például $f(x, g(x, y)) [x/g(c, y)] = f(g(c, y), g(g(c, y), y))$.

Praktice úgy kapjuk $u[x/t]$ -t, hogy u -ban minden x -et t -re cserélünk.

Helyettesítés termekben

Tetszőleges $\mathcal{A}$ struktúrára, x változóra és u, t termekre

$$\mathcal{A}(u[x/t]) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(u).$$

u felépítése szerinti indukciót alkalmazunk:

- Ha $u = x$, akkor $u[x/t] = t$ és $\mathcal{A}(t) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(x)$ igaz.
- Ha $u = y \neq x$ változó, akkor $u[x/t] = u$ és $\mathcal{A}(y) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(y)$ igaz.
- Ha $u = f(t_1, \dots, t_n)$, akkor pedig

$$\begin{aligned}\mathcal{A}(u[x/t]) &= \mathcal{A}(f(t_1[x/t], \dots, t_n[x/t])) \\ &= I(f)(\mathcal{A}(t_1[x/t]), \dots, \mathcal{A}(t_n[x/t])) && \text{term kiértékelés} \\ &= I(f)(\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(t_1), \dots, \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(t_n)) && \text{indukciós feltevés} \\ &= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(f(t_1, \dots, t_n)) && \text{term kiértékelés}\end{aligned}$$

Helyettesítés formulákban

A formulák esetében (a kvantorok miatt) a helyettesítési algoritmus ennél bonyolultabb.

- „cseréljük az összes x -et t -re” nem felel meg az értékadásnak: pl. ha a formulánk $\forall x(1 \leq x)$ – ami a természetes számok struktúrájában **hamis**, és az $[x/(y+1)]$ helyettesítést szeretnénk végrehajtani így:
- $\forall(y+1)(1 \leq (y+1))$ (common ZH és vizsga „megoldás”) wut? ez még csak nem is formula, kvantor csak változót köthet le
- $\forall x(1 \leq (y+1))$ – ez viszont **igaz** lesz a természetes számok struktúrájában, akkor is, ha az x -nek az $\varphi(y) + 1$ értékét adjuk default értéknek. Tehát ez se jó így.

Helyettesítés formulákban

Az előző problémát az okozta, hogy az $\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}$ struktúrákban ha kiértékelünk egy formulát, abban csak a **szabad** x -ek értéke kellene $\mathcal{A}(t)$ -re változzon.

- „cseréljük az összes **szabad** x -et t -re”
- Jól hangzik, de nézzük a természetes számok struktúráját a $\varphi(x) = 1$, $\varphi(y) = 42$ default értékekkel, és az $[x/y + y + 1]$ helyettesítést mondjuk a következő formulában:

$$\forall y(x \geq y + 1)$$

- A **szemantikai** szinten: $\mathcal{A}_{[x \mapsto \mathcal{A}(y+y+1)]}$ -ben x új default értéke 85. A formula pedig **hamis**, hiszen 85 nem nagyobb minden y -ra, mint $y + 1$ (pl. $y = 314$ -re kisebb).
- Ha viszont cseréljük a szabad x -et $y + y + 1$ -re:

$$\forall y(y + y + 1 \geq y + 1)$$

Ez a formula **igaz**.

- Tehát ez se jó értékadás kiváltására.

Helyettesítés formulákban

Az előző problémát pedig az okozta, hogy x meg kellene kapja az $\mathcal{A}(t)$ értékét, de t -ben szerepel egy y , aminek t kiértékelésekor az y default értékét kellene kapnia.

- Viszont az előbb belekerült a lecserélt t egy $\forall y$ hatáskörébe, így nem a default y -t kapta értékül.
- A megoldás: ha egy t -beli változó „lekötődik”, akkor **nevezzük át a kötő kvantor változóját valami egészen újra.**

Tehát ha F formula, x változó és t term, akkor $F[x/t]$ -t **F felépítése szerinti indukcióval** a következőképp definiáljuk:

- Ha $F = \uparrow$ vagy $F = \downarrow$, akkor $F[x/t] := F$. (Nothing to do here.)
- Ha $F = \neg G$, akkor $F[x/t] := \neg(G[x/t])$. (Rekurzívan elvégezzük a helyettesítést.)
- Ha $F = G \vee H$, akkor $F[x/t] := G[x/t] \vee H[x/t]$. (Rekurzívan elvégezzük.) A többi bináris konnektívára ugyanígy.

Helyettesítés formulákban

$F[x/t]$

- Ha $F = p(t_1, \dots, t_n)$, akkor

$$F[x/t] := p(t_1[x/t], \dots, t_n[x/t]).$$

(Elvégezzük a termekben a helyettesítést.)

- Ha $F = \forall xG$ vagy $F = \exists xG$ (tehát ha ugyanaz az x a kötött változó, mint akit most helyettesítünk), akkor $F[x/t] := F$.

(Csak a szabad x -eket akarjuk cserélni, ebben a formulában nincs x szabadon, így marad.)

- Ha $F = \forall yG$ vagy $F = \exists yG$, $y \neq x$ és y nem szerepel t -ben, akkor

$$F[x/t] := \forall y(G[x/t])$$

(vagy $\exists y(G[x/t])$, értelemszerűen).

$F[x/t]$

- Végül, ha $F = \forall y G$ vagy $F = \exists y G$, $y \neq x$ és y szerepel t -ben, akkor:
 - Legyen z egy olyan változó, ami nem szerepel se F -ben, se t -ben és nem is x .

$$F[x/t] := \forall z (G[y/z][x/t]).$$

(értelemszerűen, $\exists$ esetben $\exists$ kvantorral kezdünk.)

Vagyis: először keressünk egy új z változót, nevezzük át a kötött y -t erre a z -re F -ben, majd így már elvégezhetjük a helyettesítést.

(Ha egy determinisztikus algoritmust szeretnénk mindenképpen, akkor mondhatjuk, hogy „legyen z az első ilyen változó”.)

Helyettesítés formulákban

Ez a bonyolultabb algoritmus már megvalósítja az értékadást a szintaktikai szinten:

Ha $\mathcal{A}$ struktúra, F formula, x változó és t term, akkor

$$\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) = \mathcal{A}(F[x/t]).$$

Bizonyítás F felépítése szerinti indukcióval

- Ha $F = \uparrow (\downarrow)$, mindkét oldal 1 (0), OK
- Ha $F = (\neg G)$, akkor szokás szerint

$$\mathcal{A}(F[x/t]) = \mathcal{A}(\neg(G[x/t]))$$

$$= \neg \mathcal{A}(G[x/t])$$

$$= \neg \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(G)$$

$$= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(\neg G) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F)$$

$[x/t]$ def negálásra

$\neg$ szemantika

indukciós feltevés

$\neg$ szemantika

Bizonyítás F felépítése szerinti indukcióval

- Ha pl. $F = G \vee H$, szintén ugyanígy

$$\begin{aligned}\mathcal{A}(F[x/t]) &= \mathcal{A}(G[x/t] \vee H[x/t]) && [x/t] \text{ def vagyra} \\ &= \mathcal{A}(G[x/t]) \vee \mathcal{A}(H[x/t]) && \vee \text{ szemantika} \\ &= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(G) \vee \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(H) && \text{indukciós feltevés} \\ &= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(G \vee H) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) && \vee \text{ szemantika}\end{aligned}$$

- A többi bináris konnektívára is ugyanígy
- Ha $F = p(t_1, \dots, t_n)$, akkor

$$\begin{aligned}\mathcal{A}(F[x/t]) &= \mathcal{A}(p(t_1[x/t], \dots, t_n[x/t])) && [x/t] \text{ def atomira} \\ &= I(p)(\mathcal{A}(t_1[x/t]), \dots, \mathcal{A}(t_n[x/t])) && \text{atomi szemantika} \\ &= I(p)(\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(t_1), \dots, \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(t_n)) && \text{termekre ezt tudjuk} \\ &= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(p(t_1, \dots, t_n)) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) && \text{atomi szemantika}\end{aligned}$$

Bizonyítás F felépítése szerinti indukcióval

- Ha $F = \forall xG$ vagy $F = \exists xG$, akkor x nem szabad F -ben. Ekkor tudjuk, hogy $\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) = \mathcal{A}(F)$, mert a két struktúra olyan változó default értékén tér csak el, mely nem szabad F -ben. Tehát mivel ekkor $F = F[x/t]$, így $\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) = \mathcal{A}(F[x/t])$.
- Ha $F = \forall yG$ vagy $F = \exists yG$, $x \neq y$ és y nem szerepel t -ben, akkor tetszőleges $a \in A$ -ra $\mathcal{A}_{[x \mapsto \mathcal{A}(t), y \mapsto a]} = \mathcal{A}_{[y \mapsto a, x \mapsto \mathcal{A}(t)]}$, hiszen $\mathcal{A}(t)$ értéke nem függ a benne szereplő y értékétől, a értéke meg nem függ x -étől, tehát az értékadások sorrendje nem számít. Tehát ekkor

$$\begin{aligned}\mathcal{A}_{[y \mapsto a]}(G[x/t]) &= \mathcal{A}_{[y \mapsto a, x \mapsto \mathcal{A}(t)]}(G) && \text{indukciós feltevés} \\ &= \mathcal{A}_{[x \mapsto \mathcal{A}(t), y \mapsto a]}(G) && \text{mert } y \text{ nincs } t\text{-ben} \\ &= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]_{[y \mapsto a]}}(G) && \text{külön vettük,}\end{aligned}$$

és ha ez minden a -ra igaz, akkor $\mathcal{A}(F[x/t]) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F)$ is fennáll.

Bizonyítás F felépítése szerinti indukcióval

- Végül, ha $F = \forall yG$ vagy $F = \exists yG$, és y szerepel t -ben, akkor ha $z \neq x$ olyan változó, mely sem F -ben, sem t -ben nem szerepel, akkor tetszőleges $a \in A$ -ra:

$$\mathcal{A}_{[z \mapsto a]}(G[y/z][x/t])$$

$$= \mathcal{A}_{[z \mapsto a, x \mapsto \mathcal{A}(t)]}(G[y/z])$$

indukciós feltevés

$$= \mathcal{A}_{[x \mapsto \mathcal{A}(t), z \mapsto a]}(G[y/z])$$

z, x felcserélhetők

$$= \mathcal{A}_{[x \mapsto \mathcal{A}(t), z \mapsto a, y \mapsto a]}(G)$$

indukciós feltevés

$$= \mathcal{A}_{[x \mapsto \mathcal{A}(t), y \mapsto a]}(G)$$

z -től nem függ már semmi

$$= \mathcal{A}_{[x \mapsto \mathcal{A}(t)]_{[y \mapsto a]}}(G)$$

külön vettük,

és ha ez minden a -ra igaz, akkor $\mathcal{A}(F[x/t]) = \mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F)$ is fennáll.

Ez volt az utolsó eset.

- Tehát: van egy (összetett) algoritmusunk helyettesítésre, ami kiváltja az értékadást.
- (Legalábbis akkor, ha az x -nek adni kívánt értéket jelöli valamilyen term a struktúrában, de ez nekünk épp elég lesz.)
- Ebből kijön az is (amit eddig is „érezni lehetett”), hogy változót átnevezni ekvivalens átalakítás:

Ha $F = \forall xG$ (vagy $\exists xG$) formula és y nem fordul elő szabadon G -ben, akkor $F \equiv \forall y(G[x/y])$. ($\exists$ esetben $\exists$, persze.)

Azt elég belátnunk, hogy tetszőleges $\mathcal{A}$ struktúrára és $a \in A$ -ra $\mathcal{A}_{[x \mapsto a]}(G) = \mathcal{A}_{[y \mapsto a]}(G[x/y])$. De persze

$$\begin{aligned}\mathcal{A}_{[y \mapsto a]}(G[x/y]) &= \mathcal{A}_{[y \mapsto a, x \mapsto a]}(G) \\ &= \mathcal{A}_{[x \mapsto a]}(G)\end{aligned}$$

a helyettesítési lemma szerint
mert y nincs G -ben szabadon.

A $\forall xG \equiv \forall y(G[x/y])$ képletben (ha y nincs G -ben szabadon) persze oda kell figyelni a részletekre, pl. ha

$$G = p(x, z) \wedge \forall x p(x, z) \wedge \exists y p(x, y),$$

akkor az átnevezés eredménye:

$$\forall xG \equiv \forall y(p(y, z) \wedge \forall x(p(\textcolor{red}{x}, z)) \wedge \exists z p(\textcolor{red}{y}, z))$$

lesz.

Hogy egy formulával dolgozni tudjunk, ún. **zárt Skolem** alakra fogjuk hozni. Ennek lépései:

- Nyilak eliminálása – a szokásos módon
- Kiigazítás – hogy ne legyen változónév-ütközés
- Prenex alakra hozás – ekkor az összes kvantor előre kerül
- Skolem alakra hozás – ekkor az összes kvantor elöl lesz és mind $\forall$
- Lezárás – ne maradjon szabad változó-előfordulás

Kiigazítás

Egy formula **kiigazított**, ha

- Különböző helyen lévő kvantorok különböző változókat kötnek és
- Nincs olyan változó, mely szabadon is és kötötten is előfordul.

$$\exists x p(x, y) \wedge \forall x q(f(x), x) \wedge \exists y p(y, f(y))$$

nem kiigazított, mert pl.

- az x változót két különböző helyen is köti kvantor,
- az y pedig előfordul kötötten is és szabadon is.

(A fenti kettő közül az egyik is elég lenne.)

$$\exists x_1 p(x_1, y) \wedge \forall x_2 q(f(x_2), x_2) \wedge \exists y_1 p(y_1, f(y_1))$$

egy ekvivalens kiigazított formula.

Általában is igaz:

Tetszőleges F formulát ekvivalens kiigazított alakra hozhatunk a **kötött** változók új változókra való átnevezésével.

A gyakorlatban pl. ezt indexeléssel érjük el.

Fontos, hogy a szabad változókat nem nevezhetjük át! Akkor nem lesz ekvivalens a formula az eredetivel.

Egy formula **prenex** alakú, ha $Q_1x_1Q_2x_2\ldots Q_nx_nF$ alakú, ahol F kvantormentes és mindegyik Q_i kvantor (vagy $\forall$, vagy $\exists$).

Tehát prenex, ha minden kvantor elől van.

Note: a

$$\forall x p(x) \wedge q(y)$$

formula **nincs** prenex alakban, mert a $q(y)$ már nincs a $\forall x$ hatáskörében!

Prenex alakra hozás

Állítás

Minden formula ekvivalens prenex alakra hozható.

A prenex alakra hozás első lépéseként ki kell igazítsuk a formulát. Ez pl. a következők miatt fontos:

- **Nem igaz**, hogy $\exists xF \wedge \exists xG \equiv \exists x(F \wedge G)$. Például, ha $F = (x < 0)$ és $G = (x > 0)$, az egész számok szokásos struktúrájában kiértékelve
 - a bal oldal: „van negatív szám és van pozitív szám”, ez igaz,
 - a jobb oldal: „van olyan szám, ami negatív és pozitív”, ez hamis.

Tehát ha különböző kvantorok különböző változókat kötnek, nem feltétlen lehet azokat „összeolvasztani”.

- Egyébként $\exists xF \vee \exists xG \equiv \exists x(F \vee G)$ igaz. Az univerzális kvantor esetében is hasonló a helyzet: $\forall xF \wedge \forall xG \equiv \forall x(F \wedge G)$ igaz, de $\forall xF \vee \forall xG \equiv \forall x(F \vee G)$ **nem mindig igaz**.

- Az **sem mindig igaz**, hogy $\exists x F \wedge G \equiv \exists x (F \wedge G)$.

Például, ha $F = (x < 0)$ és $G = (x > 0)$ (tehát G -ben x szabadon fordul elő!), akkor az egész számok szokásos struktúrájában a $\varphi(x) = 42$ default értékadással

- A bal oldal: „van negatív szám, és $42 > 0$ ”, ez igaz,
- A jobb oldal: „van olyan szám, ami negatív és pozitív”, ez hamis.
- Itt pedig az okozza a problémát, hogy az eredetileg G -ben szabad x előfordulást ha leköti egy kvantor, akkor már nem a struktúra φ komponense adja neki az értéket, tehát a szemantika (jó eséllyel) egész más lesz.

Ez a két ok az, amiért a **kiigazításakor** épp azt szeretnénk elérni, hogy

- különböző kvantor-előfordulások különböző változókat kössenek le
- és ne legyen olyan változó, mely szabadon is és kötötten is előfordul.

Prenex alakra hozás

A következők viszont igazak:

$$\neg \exists x F \equiv \forall x \neg F$$

$$\neg \forall x F \equiv \exists x \neg F$$

$$\exists x F \vee G \equiv \exists x (F \vee G)$$

ha x nem szerepel G -ben szabadon

$$\exists x F \wedge G \equiv \exists x (F \wedge G)$$

ha x nem szerepel G -ben szabadon

$$\forall x F \vee G \equiv \forall x (F \vee G)$$

ha x nem szerepel G -ben szabadon

$$\forall x F \wedge G \equiv \forall x (F \wedge G)$$

ha x nem szerepel G -ben szabadon

$$F \vee \exists x G \equiv \exists x (F \vee G)$$

ha x nem szerepel F -ben szabadon

$$F \wedge \exists x G \equiv \exists x (F \wedge G)$$

ha x nem szerepel F -ben szabadon

$$F \vee \forall x G \equiv \forall x (F \vee G)$$

ha x nem szerepel F -ben szabadon

$$F \wedge \forall x G \equiv \forall x (F \wedge G)$$

ha x nem szerepel F -ben szabadon

Az előző fólia ekvivalenciái közül csak néhányat nézünk meg.

$$\neg \exists x F \equiv \forall x \neg F$$

Tetszőleges $\mathcal{A}$ struktúrára

$$\mathcal{A}(\neg \exists x F) = 1 \Leftrightarrow \mathcal{A}(\exists x F) = 0$$

$\neg$ szemantikája

$$\Leftrightarrow \text{minden } a \in A\text{-ra } \mathcal{A}_{[x \mapsto a]}(F) = 0$$

$\exists$ szemantikája

$$\Leftrightarrow \text{minden } a \in A\text{-ra } \mathcal{A}_{[x \mapsto a]}(\neg F) = 1$$

$\neg$ szemantikája

$$\Leftrightarrow \mathcal{A}(\forall x \neg F) = 1$$

$\forall$ szemantikája

Prenex alakra hozás

$\exists x F \wedge G \equiv \exists x (F \wedge G)$, ha x nem szerepel G -ben szabadon

$$\mathcal{A}(\exists x F \wedge G) = 1$$

$$\Leftrightarrow \mathcal{A}(\exists x F) = 1 \text{ és } \mathcal{A}(G) = 1$$

$$\Leftrightarrow \text{valamilyen } a \in A\text{-ra } \mathcal{A}_{[x \mapsto a]}(F) = 1 \text{ és } \mathcal{A}(G) = 1$$

$$\Leftrightarrow \text{valamilyen } a \in A\text{-ra } (\mathcal{A}_{[x \mapsto a]}(F) = 1 \text{ és } \mathcal{A}_{[x \mapsto a]}(G) = 1)$$

$$\Leftrightarrow \mathcal{A}(\exists x (F \wedge G)) = 1,$$

ahol a **piros** lépésnél használtuk, hogy mivel G -ben nincs x szabadon, így
 $\mathcal{A}(G) = \mathcal{A}_{[x \mapsto a]}(G)$ tetszőleges a -ra igaz.

Megjegyzés: itt megint fontos, hogy az univerzum nem lehet üres! Ha megengednénk üres univerzumot, akkor pl. $\exists x F \vee G$ mindig G -vel lenne ekvivalens üres univerzum esetén (mert a $\exists x F$ alakú mondatok ott hamisak lennének), a $\exists x (F \vee G)$ pedig hamis lenne. (ugyanezért).

Prenex alakra hozás

Az előző ekvivalenciák adnak egy prenex alakra hozó algoritmust:

- Imináljuk el a formulából a $\rightarrow$ és $\leftrightarrow$ konnektívákat a szokott módon
- Az eredményt igazítsuk ki
- Az eredményen addig alkalmazzuk (balról jobbra) az ekvivalenciákat (amiket lehet a kiigazítottság miatt), amíg prenex alakú formulát nem kapunk.

A sorrend fontos:

- A „kvantorokat kihozó” ekvivalenciák csak kiigazított formulán alkalmazhatók
- A nyilak eliminálása elronthatja a kiigazítottságot (a $\leftrightarrow$ -on belüli kvantorok duplikálódnak)

Példa prenex alakra hozás

$$\neg \left(\left((\forall x \neg \forall y (p(x, y) \vee \exists z q(x, z))) \wedge \exists w p(x, w) \right) \vee \neg \exists v q(v, v) \right)$$

$$\neg \left(\exists w \left((\forall x \exists y \neg \exists z (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \forall v \neg q(v, v) \right)$$

$$\neg \left(\exists w \left((\forall x \exists y \neg \exists z (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \forall v \neg q(v, v) \right)$$

$$\neg \exists w \left(\left((\forall x \exists y \forall z \neg (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \forall v \neg q(v, v) \right)$$

Példa prenex alakra hozás

$$\neg \exists w \left(\left((\forall x \exists y \forall z \neg (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \forall v \neg q(v, v) \right)$$

$$\forall w \neg \forall v \left(\forall x \exists y \forall z \left((\neg (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \neg q(v, v) \right)$$

$$\forall w \neg \forall v \forall x \exists y \forall z \left(\left((\neg (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \neg q(v, v) \right)$$

$$\forall w \exists v \exists x \forall y \exists z \neg \left(\left((\neg (p(x, y) \vee q(x, z))) \wedge p(x, w) \right) \vee \neg q(v, v) \right)$$

(Kaphattunk volna másik ekvivalens prenex alakot is, pl. v -t kihozhattuk volna utoljára is, stb.)

Skolem alak

Egy F formula **Skolem alakú**, ha

$$F = \forall x_1 \forall x_2 \dots \forall x_n F^*,$$

ahol F^* -ben (a formula **magjában**) már nincs kvantor.

(Tehát: prenex alakú, és csak univerzális kvantor szerepel benne.)

(Thoralf Skolem **norvég** matematikus, 1887–1963 után, tehát „sz”-szel ejtjük)

A Skolem alakú formulákkal azért jó dolgozni, mert

$$\forall x_1 \dots \forall x_n F^* \models F^*[x_1/t_1, \dots, x_n/t_n]$$

tetszőleges t_i termekre. Ez formálisan kijön abból, hogy

- ha $\mathcal{A}(\forall x F) = 1$, akkor tetszőleges $a \in A$ -ra $\mathcal{A}_{[x \mapsto a]}(F) = 1$
- tehát tetszőleges t termre $\mathcal{A}_{[x \mapsto \mathcal{A}(t)]}(F) = 1$ (hiszen $\mathcal{A}(t) \in A$)
- a helyettesítési lemma szerint pedig ez épp $\mathcal{A}(F[x/t])$.

Skolem alak

Az (sajnos) **nem igaz**, hogy minden formulához megadható lenne **ekvivalens** Skolem alak.

A következő viszont igaz:

Minden F formulához konstruálható egy olyan F' Skolem alakú formula, ami pontosan akkor kielégíthető, ha F is az.

Tehát: tudunk adni olyan algoritmust, mely

- egy F formulából elkészít egy F' Skolem alakot
- úgy, hogy ha F kielégíthető, akkor F' is, és ha F kielégíthetetlen, akkor F' is.

Ezt úgy mondjuk, hogy F és F' **s -ekvivalensek**, jelben $F \equiv_s F'$.

(s is for „satisfiability” here)

Ez általában elég lesz, hiszen eleve a kielégíthetetlenségre keresünk algoritmust.

A Skolem alakra hozás lépései:

- Prenex alakra hozzuk a formulát (duh)
- Egyesével elimináljuk belőle a $\exists x$ kvantorokat **valahogy**

Persze ha csak úgy törölnénk a $\exists x$ kvantorokat, az nem lenne helyes algoritmus még s -ekvivalencia szempontjából sem: pl.

$$\forall x \exists y (p(x, y) \wedge \neg p(y, x))$$

kielégíthető: legyen a struktúránk a természetes számok halmaza, és p -t interpretáljuk mondjuk a „kisebb” relációval. Ekkor a mondat: „minden számnál van olyan, ami nála nagyobb és nem kisebb”, ez igaz.

Viszont ha töröljük a $\exists y$ kvantort:

$$\forall x (p(x, y) \wedge \neg p(y, x))$$

kielégíthetetlen (vegyük észre, hogy itt y szabaddá vált), hiszen tetszőleges $\mathcal{A}$ struktúrában az $a := \varphi(y)$ elemre

$$\mathcal{A}_{[x \mapsto \varphi(y)]}(p(x, y) \wedge \neg p(y, x)) = I(p)(\varphi(y), \varphi(y)) \wedge \neg I(p)(\varphi(y), \varphi(y)),$$

ami mindenképp hamis, bármi is $I(p)(\varphi(y), \varphi(y))$ értéke.

Skolem alak

Az sem megoldás, ha a $\exists x$ -eket $\forall x$ -ekre cseréljük, az előző példából így kapott

$$\forall x \forall y (p(x, y) \wedge \neg p(y, x))$$

szintén kielégíthetetlen (ez is biztosan hamissá válik akkor, ha x -nek és y -nak ugyanazt az értéket adjuk).

Az előző ötletekkel a következő a probléma intuitíve:

- Egy $\forall x \exists y F$ alakú formula kiértékelésekor előbb megválasztjuk x értékét, majd **ettől függően** választjuk meg y értékét.
- Általában is egy $\exists$ kvantor értékének a megválasztásakor (prenex alakú formula esetében) a „jó” érték függhet **az összes előtte álló $\forall$ -kvantált változó értékétől**.
- Pl. $\forall x \forall y \exists z ((x < z) \wedge (y < z))$ a természetes számok struktúrájában: „minden x -re és y -ra létezik z , mely mindkettőnél nagyobb”, itt is z értéke függhet x -étől és y -étől is.

A Skolem alakra hozás ötlete:

Ha az $\exists y$ -nal kötött változó „jó” értéke függhet az $x_1, \dots, x_n$ változók értékétől, akkor vezessünk be egy f függvényt, ami „megmondja”, hogy hogyan függ!

Azaz az elképzelés az, hogy $I(f)(a_1, \dots, a_n) := b$ egy olyan b -re, amit akkor kapna y értékül, ha x_1 értéke a_1 , x_2 értéke a_2 , stb.

Ezeket az újonnan bevezetett függvényeket **Skolem-függvényeknek** nevezzük.

Ez működni is fog:

Ha az f/n függvényjel nem szerepel az $F = \forall x_1 \forall x_2 \dots \forall x_n \exists y F^*$ kiigazított formulában, akkor

$$F \equiv_s \forall x_1 \forall x_2 \dots \forall x_n F^*[y/f(x_1, \dots, x_n)] =: F'$$

Bizonyítás

Ennél többet mutatunk meg: azt, hogy

- ha $\mathcal{A}(F) = 1$ az $\mathcal{A} = (A, I, \varphi)$ struktúrára, akkor van olyan $\mathcal{A}' = (A, I', \varphi)$ struktúra az f -fel kibővített nyelvre, melyre minden eredeti s (függvény- vagy predikátum)szimbólumra $I(s) = I'(s)$.

Tehát: ha F -nek van modellje, akkor ahhoz hozzá tudunk adni egy „jó” f interpretációt úgy, hogy az az F' -nek (is) modellje legyen.

Bizonyítás folytatása

- Továbbá, azt is meg fogjuk mutatni, hogy ha $\mathcal{A}'(F') = 1$, akkor $\mathcal{A}(F) = 1$ arra az $\mathcal{A}$ struktúrára, amit úgy kapunk $\mathcal{A}'$ -ből, hogy f interpretációját „elfelejtjük”.
- Ez az irány könnyű: ha $\mathcal{A}'(F') = 1$, akkor minden $a_1, a_2, \dots, a_n \in A$ -ra

$$\mathcal{A}'_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n]}(F^*[y/f(x_1, \dots, x_n)]) = 1,$$

ami a helyettesítési lemma szerint épp azt jelenti, hogy

$$\mathcal{A}'_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n, y \mapsto I'(f)(a_1, \dots, a_n)]}(F^*) = 1.$$

Ez pedig azt jelenti, hogy a $b = I'(f)(a_1, \dots, a_n)$ értékre

$$\mathcal{A}'_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n, y \mapsto b]}(F^*) = 1, \text{ tehát}$$

$$\mathcal{A}'_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n]}(\exists y F^*) = 1,$$

tehát $\mathcal{A}'(F) = 1$ (és mivel F -ben nincs f , így $\mathcal{A}(F) = 1$).

Bizonyítás folytatása

A másik irány:

- Tegyük fel, hogy $\mathcal{A}(F) = 1$. Akkor tetszőleges $a_1, \dots, a_n \in A$ esetén

$$\mathcal{A}_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n]}(\exists y F^*) = 1.$$

- Tehát tetszőleges $a_1, \dots, a_n \in A$ esetén van olyan b , melyre

$$\mathcal{A}_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n, y \mapsto b]}(F^*) = 1.$$

- Definiáljuk az $\mathcal{A}'$ struktúrában $I'(f)$ -et úgy, hogy az $I'(f)(a_1, \dots, a_n)$ értéke legyen egy ilyen „alkalmas” b .
- Akkor tetszőleges $a_1, \dots, a_n \in A$ esetén

$$\mathcal{A}'_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n, y \mapsto I'(f)(a_1, \dots, a_n)]}(F^*) = 1.$$

- Ez a helyettesítési lemma szerint épp azt jelenti, hogy

$$\mathcal{A}'_{[x_1 \mapsto a_1, \dots, x_n \mapsto a_n]}(F^*[y/f(x_1, \dots, x_n)]) = 1, \text{ tehát } \mathcal{A}'(F') = 1.$$

Tehát vegyük a következő algoritmust: az F input formulát

- először hozzuk prenex alakra (nyilak eliminálása, kiigazítás, kvantorok kiemelése)
- majd minden $\exists y$ -lekötött változót a formula magjában **cseréljük le** egy $f(x_1, \dots, x_n)$ termre, ahol
 - f egy **teljesen új** függvénszimbólum (tehát ha formulák egy egész Σ halmazával dolgozunk, akkor Σ semelyik elemében nem lehet f ! és minden $\exists$ kvantor esetében újabb és újabb Skolem-függvényeket gyártunk!)
 - $x_1, \dots, x_n$ pedig az y **előtt** szereplő $\forall$ -kötött változók.

A kapott formula s -ekvivalens lesz az eredetivel.

(Sőt, ha ezt egy input Σ formulahalmaz minden formulája elvégezzük, akkor a kapott Σ' formulahalmaz is s -ekvivalens lesz az eredetivel.)

Skolem alak, példa

$$\exists x \forall y \forall z \exists v \exists w (p(x, y, f(z)) \wedge \neg q(x, f(v), w) \wedge p(c, v))$$

- x egy egzisztenciálisan kötött változó, előtte nem szerepel univerzálisan kötött, tehát helyére egy nullaváltozós Skolem-függvényt, azaz **Skolem konstanst** vezetünk be. Ez nem lehet c , mert az már szerepel a formulában, legyen mondjuk d .
- $\exists v$ előtt $\forall$ -kötött változó y és z , tehát egy új bináris Skolem függvényjelet generálunk, legyen ez g (mert f nem lehet, az foglalt), v -t mindenhol $g(y, z)$ -re cseréljük.
- w -t pedig mindenhol $h(y, z)$ -re (a neki generált Skolem-függvény már nem lehet sem f , sem g).

Az eredmény:

$$\forall y \forall z (p(d, y, f(z)) \wedge \neg q(d, f(g(y, z)), h(y, z)) \wedge p(c, g(y, z))).$$

Zárt Skolem alak

Az ún. **alap (ground) rezolúciós algoritmus** olyan Σ formulahalmazon tud majd dolgozni, melynek

- minden eleme Skolem alakú,
- zárt (tehát nincs bennük szabad változó),
- és minden formula magja CNF-ben van.

Ha tehát egy tetszőleges Σ formulahalmazt kapunk inputként, ennek minden elemét Skolem-alakra tudjuk már hozni, a magjukat pedig CNF-re.

Hogy **zárt** legyen minden formulánk, azt úgy érjük el, hogy

- minden x szabad előfordulás helyett egy **új** c_x konstansjelet vezetünk be
- ezt úgy, hogy minden formulában az összes szabad x helyére **ugyanazt** a c_x -et írjuk!

Ezzel az átalakítással továbbra is s -ekvivalensek maradunk (mert: legyen $I(c_x) := \varphi(x)$ jó lesz).

Levezetés a Peano-axiómákból

Nézzük most meg, hogy kb. hogyan jöhet ki a Peano-axiómákból, hogy

$$\forall x \forall y \forall z \left((y + z) \times x = y \times x + z \times x \right).$$

Jelölje PA a Peano-axiomákat. Köztük vannak az **indukciós axiómaséma** példányai:

$$\left(F(0) \wedge \forall x (F(x) \rightarrow F(x')) \right) \rightarrow \forall x F(x).$$

Ha $F(x) = \forall y \forall z \left((y + z) \times x = y \times x + z \times x \right)$, akkor elég levezetni ezt a kettőt:

- $PA \models \forall y \forall z \left((y + z) \times 0 = y \times 0 + z \times 0 \right)$

(ez az $F(0)$)

- $PA \models$

$$\forall x \left(\forall y \forall z \left((y + z) \times x = y \times x + z \times x \right) \rightarrow \forall y \forall z \left((y + z) \times x' = y \times x' + z \times x' \right) \right)$$

(ez a $\forall x (F(x) \rightarrow F(x'))$)

$$\text{PA} \models \forall y \forall z \left((y + z) \times 0 = y \times 0 + z \times 0 \right)$$

Két eleme PA-nak: $\forall x(x \times 0 = 0)$ és $\forall x(x + 0 = x)$. Ezekkel:

$\forall y \forall z ((y + z) \times 0 = 0)$ „ha minden x -re, akkor $y + z$ -re is”

$\forall y \forall z (y \times 0 = 0)$ „ha minden x -re, akkor y -ra is”

$\forall y \forall z (z \times 0 = 0)$ „ha minden x -re, akkor z -ra is”

$\forall y \forall z (y \times 0 + z \times 0 = 0 + 0)$ egyenlő dolgok összege egyenlő

$\forall y \forall z (y \times 0 + z \times 0 = 0)$ $0 + 0 = 0$

$\forall y \forall z ((y + z) \times 0 = y \times 0 + z \times 0)$ mindkettő 0-val egyenlő

Ezzel az indukciós axiómaséma példányának az $F(0)$ része tényleg következménye PA-nak. (Note: itt figyeljük meg, hogy az egyenlőségnek használjuk pár tulajdonságát!)

$$\forall x \left(\forall y \forall z \left((y + z) \times x = y \times x + z \times x \right) \rightarrow \forall y \forall z \left((y + z) \times x' = y \times x' + z \times x' \right) \right)$$

A $\forall x \forall y (x \times y' = x \times y + x)$ Peano-axiómából:

$$\forall x \forall y \forall z \left((y + z) \times x' = (y + z) \times x + (y + z) \right)$$

Egy egyenlőséges implikációt („ha $t_1 = t_2$, akkor $u[x/t_1] = u[x/t_2]$ ”) alkalmazva:

$$\forall x \left(\forall y \forall z \left((y + z) \times x = y \times x + z \times x \right) \rightarrow \forall y \forall z \left((y + z) \times x' = (y \times x + z \times x) + (y + z) \right) \right)$$

A $\forall x \forall y (x + y = y + x)$ és $\forall x \forall y \forall z ((x + y) + z = x + (y + z))$ -ből:

$$\forall x \forall y \forall z \left((y \times x + z \times x) + (y + z) = (y \times x + y) + (z \times x + z) \right)$$

A $\forall x \forall y (x \times y' = x \times y + x)$ -ből ez tovább:

$$\forall x \forall y \forall z \left((y \times x + z \times x) + (y + z) = (y \times x') + (z \times x') \right)$$

és megvan:

$$\forall x \left(\forall y \forall z \left((y + z) \times x = y \times x + z \times x \right) \rightarrow \forall y \forall z \left((y + z) \times x' = y \times x' + z \times x' \right) \right) \quad 228$$

- A helyettesítési algoritmusnak látszólag van egy gyenge pontja.
- Csak akkor tudjuk elvégezni a $\mathcal{A}_{[x \mapsto a]}(F)$ értékadást a szintaktikai szinten $\mathcal{A}(F[x/t])$ -ként, ha **van olyan t** , amire $\mathcal{A}(t) = a$.
- Nem minden struktúrában van ilyen t ! Pl. a **valós** vagy a **racióális** számok struktúrájában ha csak a $+$ és $\times$ jeleket tudjuk használni, mondjuk a 0 és 1 konstansokkal, akkor azokkal is csak az **egész** számokat tudjuk eljelölni (ezeket tudjuk felépíteni ground termként).
- Azt **szeretnénk**, ha egy $\forall x F$ alakú formula (vagy általában: zárt Skolem alakok halmazának) kielégíthetőségének vizsgálatakor elég lenne ellenőriznünk az „összes” $F[x/t]$ alakú formulából álló halmazt.
- Ez pl. a racionális számok struktúrájában nem működik: a $\forall x(\text{egész}(x))$ formula ott **hamis**, de ha bármilyen $[x/t]$ helyettesítést végzünk a magon, a kapott $\text{egész}(t)$ formula mindig **igaz** lesz (ha t alapterm).

- Ezt a problémát oldják meg az ún. **Herbrand struktúrák**, amikre a következő lesz igaz:
- Herbrand struktúrában minden elem eljelölhető egy alaptermmel.
- Ha egy Σ formulahalmaz kielégíthető, akkor van Herbrand modellje (azaz: Σ -t kielégítő Herbrand struktúra) is.
- Ez a két tulajdonság fogja azt biztosítani, hogy **kielégíthetetlenség bizonyításakor** a szemantikus értékadás művelet tényleg kiváltható lesz a szintaktikus helyettesítés művelettel.

Herbrand struktúrák

De mi is egy Herbrand struktúra?

- (Jacques Herbrand, 1908–1931)
- Az **alaptermek** vagy **ground termek** a változómentes termek.
- Tehát: melyeket felépíthetünk csak a függvényjelek alkalmazásával.
- Pl. ha $f/1$, $g/2$ és $c/0$ függvényjelek, akkor alaptermek pl.

$$c, f(c), f(f(c)), g(c, c), g(c, f(c)), g(f(c), f(c)), \dots$$

- Az összes alapterm halmazát T_0 jelöli. (A 0 itt a „0 változó”-t jelenti, általában T_n -be azok a termek tartoznak, melyekben csak az $\{x_1, \dots, x_n\}$ változók szerepelnek.)
- A következő részben fontos, hogy T_0 ne legyen üres. Ezért **ha a nyelvünkben nincs konstansjel, akkor generálunk egyet.**

Herbrand struktúrák

- Egy $\mathcal{A} = (A, I, \varphi)$ struktúrát **Herbrand struktúrának** nevezünk, ha
- **univerzuma** $A = T_0$ az alaptermek halmaza és
- tetszőleges f/n függvényjelre $I(f)(t_1, \dots, t_n) = f(t_1, \dots, t_n)$.

Tehát az objektumok maguk az alaptermek, és a függvényjelek interpretációja „értelem-szerű”.

Például ha $f/1, g/2, c/0$:

- $c, f(c), f(f(c)), g(c, c), g(f(c), f(c))$ mind **objektumok**
- persze $c, f(c), f(f(c)), \dots$ továbbra is (alap)**termek** is egyben
- és pl. a $\mathcal{A}(g(c, f(c)))$ értéke egy $\mathcal{A}$ Herbrand struktúrában:
 - $I(c)() = c$ a Herbrand struktúra definíciója szerint, tehát $\mathcal{A}(c) = c$.
 - mivel $\mathcal{A}(c) = c$, ezért $\mathcal{A}(f(c)) = I(f)(\mathcal{A}(c)) = I(f)(c) = f(c)$.
 - Tehát $g(c, f(c))$ értéke

$$\mathcal{A}(g(c, f(c))) = I(g)(\mathcal{A}(c), \mathcal{A}(f(c))) = I(g)(c, f(c)) = g(c, f(c)).$$

Herbrand struktúrák

Amit az előbb láttunk, mindig igaz: Herbrand struktúrában egy alapterm értéke mindig önmaga.

Állítás

Tetszőleges $\mathcal{A}$ Herbrand struktúrára és t alaptermre $\mathcal{A}(t) = t$.

Bizonyítás

A t term **felépítése szerinti indukciót** alkalmazunk:

- $t = x$, x változó: ez az eset **nem lehet**, mert alaptermben nincs változó.
- $t = f(t_1, \dots, t_n)$: Ha t alapterm, akkor $t_1, \dots, t_n$ is mind alaptermek. Tehát az **indukciós feltevés szerint** $\mathcal{A}(t_i) = t_i$ minden i -re és ekkor

$$\begin{aligned}\mathcal{A}(f(t_1, \dots, t_n)) &= I(f)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)) && \text{term kiértékelés def} \\ &= I(f)(t_1, \dots, t_n) && \text{indukciós feltevés} \\ &= f(t_1, \dots, t_n) = t. && \text{Herbrand } I(f) \text{ def}\end{aligned}$$

Példa

Ha csak a 0 konstansjelünk és az s unáris függvényjelünk van, akkor egy Herbrand struktúrában:

- az univerzum az alaptermek halmaza: $\{0, s(0), s(s(0)), s(s(s(0))), \dots\}$
- azzal, hogy $I(0) = 0$ és $I(s)(s^n(0)) = s^{n+1}(0)$

tehát a természetes számok struktúrája az s (successor, rákövetkezés) művelettel egy Herbrand struktúra.

Ha ebben a struktúrában az összeadást mint **predikátumot** definiáljuk (a szándék: $I(\text{add})(x, y, z) = 1$, ha $x + y = z$), ezt formalizálhatjuk:

- $\forall x \forall y \forall z_1 \forall z_2 (\text{add}(x, y, z_1) \wedge \text{add}(x, y, z_2) \rightarrow z_1 = z_2)$
- $\forall x \forall y (\text{add}(x, 0, x))$
- $\forall x \forall y \forall z (\text{add}(x, y, z) \rightarrow \text{add}(x, s(y), s(z)))$

A szorzást is hasonlóan.

Egyenlőség egyelőre ne legyen

Az egyenlőség

Note: Herbrand struktúrákban az **egyenlőség** kezelése problémássá válik.

- Pl. ha a természetes számok struktúrájában használhatjuk a $+/2$ függvényjelet is, akkor a 0 és a $0 + 0$ alaptermek **nem** egyenlőek, de szeretnénk őket egyenlőnek **kezelni**.
- erre még visszatérünk
- egyelőre a nyelvünkben ne legyen egyenlőség

Herbrand struktúrák

- Az így nyilván igaz, hogy egy Herbrand struktúrában minden objektumot el tudunk jelölni egy alaptermmel: a t alaptermet mind objektumot maga a t alapterm jelöli.
- Ami emiatt **algoritmikus szempontból** fontos: ha $\mathcal{A}$ egy Herbrand struktúra és van egy $\forall x F$ alakú formulánk, akkor arra

$$\mathcal{A} \models \forall x F \Leftrightarrow \mathcal{A} \models F[x/t] \text{ minden } t \text{ alaptermre.}$$

(Mert így a helyettesítési lemma szerint $\mathcal{A}_{[x \mapsto a]} \models F$ minden $a \in A$ -ra.)

- Azaz, $\forall x F$ és $\{F[x/t] : t \in T_0\}$ Herbrand-modelljei ugyanazok!

Ha $\forall x_1 \forall x_2 \dots \forall x_n F$ egy zárt Skolem alak, ahol F kvantormentes, akkor **Herbrand kiterjesztése** az

$$E(\forall x_1 \forall x_2 \dots \forall x_n F) := \left\{ F[x_1/t_1, x_2/t_2, \dots, x_n/t_n] : t_i \in T_0 \right\}$$

formulahalmaz. (E mint „extension”)

Példa

Ha $F = \forall x \forall y (p(x, y) \wedge \neg p(f(x), g(x, y)))$, és $c/0$, $f/1$, $g/2$ a függvényjelek, akkor $E(F)$ -ben vannak pl.

- $p(c, c) \wedge \neg p(f(c), g(c, c))$
- $p(c, f(c)) \wedge \neg p(f(c), f(c, f(c)))$
- $p(f(c), g(c, c)) \wedge \neg p(f(f(c)), g(f(c), g(c, c)))$
- ...

Ez a formula kielégíthetetlen, hiszen $E(F)$ -nek $\neg p(f(c), g(c, c))$ is és $p(f(c), g(c, c))$ is következménye, ami nem lehetséges.

Herbrand struktúrák

Amiért a kielégíthetlenség vizsgálatához elég csak Herbrand struktúrákban gondolkodnunk, az a következő:

Tétel

Zárt Skolem normálformák tetszőleges halmaza pontosan akkor kielégíthető, ha van Herbrand modellje.

(A tétel ebben a formában csak olyan mondatokra vonatkozik, melyekben **nincs egyenlőség.**)

A tétel haszna számunkra

- ha az input Σ formulahalmazról kell belátnunk, hogy kielégíthetetlen,
- akkor először is zárt Skolem alakra hozzuk, legyen ez mondjuk Σ' ,
- majd megmutatjuk, hogy nincs olyan Herbrand-struktúra, ami modellje Σ' -nek.

A Herbrand tételből és abból, hogy a zárt Skolem alakra hozás s -ekvivalens művelet, következik, hogy Σ pontosan ekkor kielégíthetetlen.

Herbrand tétel – bizonyítás

- Ha Σ -nak van Herbrand modellje, akkor nyilván kielégíthető.
- Mivel Σ mondatokból áll, a φ értékadást elhagyhatjuk.
- Tehát csak azt kell belátnunk, hogy ha van valamilyen $\mathcal{A} = (A, I)$ modellje Σ -nak, akkor van egy $\mathcal{A}' = (T_0, I')$ Herbrand modellje is.
- Mivel $\mathcal{A}'$ Herbrand-modell kell legyen, az alaphalmaz (T_0) és a függvényjelek interpretációja $I'(f)(t_1, \dots, t_n) = f(t_1, \dots, t_n)$ adott.
- Tehát csak a predikátumjeleknek kell szemantikát adnunk.
- Ha p/n predikátumjel és $t_1, \dots, t_n$ alaptermek, akkor legyen

$$I'(p)(t_1, \dots, t_n) := I(p)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)).$$

Vagyis: ha p -t akarjuk kiértékelni a Herbrand struktúrában a $t_1, \dots, t_n$ alaptermeken (mint objektumokon), akkor értékeljük ki a t_i alaptermeket (mint termeket) $\mathcal{A}$ -ban, és az így kapott (A -beli) objektumokon p értékét ($\mathcal{A}$ -ban) adjuk vissza $\mathcal{A}'$ -ben.

Bizonyítás folytatása

- Azt állítjuk, hogy ez az $\mathcal{A}'$ Herbrand struktúra modellje Σ -nak.
- Ehhez legyen $\forall x_1 \dots \forall x_n F \in \Sigma$. Megmutatjuk, hogy $\mathcal{A}'_{[x_1/t_1, \dots, x_n/t_n]}(F) = 1$ minden $t_1, \dots, t_n$ alaptermre.
- A helyettesítési lemma szerint

$$\mathcal{A}'_{[x_1/t_1, \dots, x_n/t_n]}(F) = \mathcal{A}'(F[x_1/t_1, \dots, x_n/t_n]),$$

hiszen $\mathcal{A}'(t_i) = t_i$ minden i -re, mert $\mathcal{A}'$ Herbrand struktúra.

- Azt tudjuk, hogy $\mathcal{A}(\forall x_1 \dots \forall x_n F) = 1$, tehát tetszőleges $t_1, \dots, t_n$ alaptermekre

$$\mathcal{A}_{[x_1/\mathcal{A}(t_1), \dots, x_n/\mathcal{A}(t_n)]}(F) = 1,$$

ami a helyettesítési lemma szerint azt jelenti, hogy

$$\mathcal{A}(F[x_1/t_1, \dots, x_n/t_n]) = 1.$$

Bizonyítás folytatása

- Elég tehát belátnunk, hogy

$$\mathcal{A}'(F^*) = \mathcal{A}(F^*)$$

tetszőleges F^* kvantormentes és változómentes (alap)formulára.

- Ezt F^* **felépítése szerinti indukcióval** tesszük:
- Ha $F^* = p(t_1, \dots, t_n)$ atomi formula, akkor $t_1, \dots, t_n$ alaptermek (mert F^* -ban nincs változó). Ekkor

$$\begin{aligned} & \mathcal{A}'(p(t_1, \dots, t_n)) \\ &= I'(p)(\mathcal{A}'(t_1), \dots, \mathcal{A}'(t_n)) \\ &= I'(p)(t_1, \dots, t_n) \\ &= I(p)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)) \\ &= \mathcal{A}(p(t_1, \dots, t_n)) \end{aligned}$$

mert $\mathcal{A}'$ Herbrand struktúra

$I'(p)$ def szerint

szemantika def szerint

Bizonyítás folytatása

- Ha $F^* = \neg G$, akkor G is kvantormentes alapformula, tehát

$$\begin{aligned}\mathcal{A}'(F^*) &= \neg \mathcal{A}'(G) && \text{szemantika def} \\ &= \neg \mathcal{A}(G) && \text{indukciós feltevés} \\ &= \mathcal{A}(\neg G) = \mathcal{A}(F) && \text{szemantika def}\end{aligned}$$

- és ha pl. $F^* = G \vee H$, akkor szintén a szokásos módon

$$\begin{aligned}\mathcal{A}'(G \vee H) &= \mathcal{A}'(G) \vee \mathcal{A}'(H) && \text{szemantika} \\ &= \mathcal{A}(G) \vee \mathcal{A}(H) && \text{indukció} \\ &= \mathcal{A}(G \vee H) = \mathcal{A}(F^*) && \text{szemantika,}\end{aligned}$$

a többi bináris konnektívára is ugyanígy.

- Mivel pedig kvantorok nem lehetnek F^* -ban, így kész vagyunk.

A Herbrand tétel és az egyenlőség kezelése

- Az $I'(p)(t_1, \dots, t_n) := I(p)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n))$ definíció **nem valid**, ha p az **egyenlőség** predikátum.
- Hiszen pl. ekkor a természetes számok struktúrájában, ahol van $+/2$ függvényjel, ez pl.

$$I'(=)(0, 0 + 0) := I(=)(\mathcal{A}(0), \mathcal{A}(0 + 0)) = I(=)(0, 0)$$

lenne, azaz **igaz** kéne legyen.

- Viszont az egyenlőségjelet a Herbrand struktúrában sem definiálhatjuk felül!
- (felül kéne, mert 0 és $+(0, 0)$ nem ugyanaz az alapterm.)
- A megoldás: **= helyett** a formulákban felveszünk egy új (mondjuk equals) bináris predikátumjelet, erről kimondjuk, hogy **kongruencia** és ezt használjuk = helyett mindenhol.

A Herbrand tétel és az egyenlőség kezelése

A **kongruencia** azt jelenti, hogy

- ekvivalencia-reláció:
 - $\forall x \text{ equals}(x, x)$ – reflexív
 - $\forall x \forall y \left(\text{equals}(x, y) \rightarrow \text{equals}(y, x) \right)$ – szimmetrikus
 - $\forall x \forall y \forall z \left(\text{equals}(x, y) \wedge \text{equals}(y, z) \rightarrow \text{equals}(x, z) \right)$ – tranzitív
- ...és hogy „egyenlő” értékeken alkalmazott függvények/predikátumok értéke is „egyenlő” (**kompatibilis** velük): minden f/n -re és p/n -re

$$\begin{aligned} & \forall x_1 \forall y_1 \forall x_2 \forall y_2 \dots \forall x_n \forall y_n \left(\text{equals}(x_1, y_1) \wedge \dots \wedge \text{equals}(x_n, y_n) \right. \\ & \quad \rightarrow \text{equals}(f(x_1, \dots, x_n), f(y_1, \dots, y_n)) \Big) \text{függvényjelekre} \\ & \quad \rightarrow p(x_1, \dots, x_n) \leftrightarrow p(y_1, \dots, y_n) \Big) \text{predikátumjelekre} \end{aligned}$$

Ha így alakítjuk át a formuláinkat (minden egyenlőséget equalsra cserélünk és felírjuk a fenti formulákat, miszerint equals egy kongruencia), akkor az így átalakított Σ -ra már igaz a Herbrand tétel.

Példa: a Peano axiómák

Ha például vesszük a Peano axiómákat:

- a függvényjelek ott 0 , s , $+$ és $\times$ voltak
- a predikátumjelek $\leq$ és használtuk az $=$ jelet is
- az $=$ helyett bevesszük az `equals` predikátumot
- Az átalakított axiómák:

$$\forall x (\neg(s(x) = 0)) \Rightarrow \forall x (\neg \text{equals}(s(x), 0))$$

$$\forall x \forall y (s(x) = s(y) \rightarrow x = y) \Rightarrow \forall x \forall y (\text{equals}(s(x), s(y)) \rightarrow \text{equals}(x, y))$$

$$\forall x (x + 0 = x) \Rightarrow \forall x (\text{equals}(x + 0, x))$$

$$\forall x (x \leq 0 \rightarrow x = 0) \Rightarrow \forall x (x \leq 0 \rightarrow \text{equals}(x, 0))$$

...

Példa: a Peano axiómák

...és felvesszük a következőket:

$$\forall x \text{ equals}(x, x)$$

$$\forall x \forall y (\text{equals}(x, y) \rightarrow \text{equals}(y, x))$$

$$\forall x \forall y \forall z (\text{equals}(x, y) \wedge \text{equals}(y, z) \rightarrow \text{equals}(x, z))$$

$$\forall x_1 \forall y_1 (\text{equals}(x_1, y_1) \rightarrow \text{equals}(s(x_1), s(y_1)))$$

$$\forall x_1 \forall y_1 \forall x_2 \forall y_2 (\text{equals}(x_1, y_1) \wedge \text{equals}(x_2, y_2) \rightarrow \text{equals}(x_1 + y_1, x_2 + y_2))$$

$$\forall x_1 \forall y_1 \forall x_2 \forall y_2 (\text{equals}(x_1, y_1) \wedge \text{equals}(x_2, y_2) \rightarrow \text{equals}(x_1 \times y_1, x_2 \times y_2))$$

$$\forall x_1 \forall y_1 \forall x_2 \forall y_2 (\text{equals}(x_1, y_1) \wedge \text{equals}(x_2, y_2) \rightarrow (x_1 \leq y_1 \leftrightarrow x_2 \leq y_2))$$

akkor ennek a rendszernek egy Herbrand modelljében

- az alaptermek pl. 0 , $s(0)$, $s(0) + s(s(0))$, $s(0) + 0$ stb.
- a függvényinterpretációk: $I(+)(s(0), s(s(0))) = s(0) + s(s(0))$ stb.
- az equals-ra pl. $\text{equals}(s(s(0)), s(0) + s(0))$ igaz (de nem **egyenlőek**)
- a $\leq$ -re pl. $s(s(0)) \leq s(0) + s(s(0)) \times s(s(0))$ igaz

Herbrand tétel

Következmény

Formulák egy Σ halmaza pontosan akkor kielégíthető, ha létezik **megszámlálható** modellje.

Vagyis: ha Σ kielégíthető, akkor van olyan modellje is, melynek univerzuma „nem túl nagy”: véges vagy megszámlálhatóan végtelen.

Bizonyítás

- Tegyük fel, hogy Σ kielégíthető.
- Hozzuk zárt Skolem alakra Σ -t, legyen a kapott formulahalmaz Σ' .
- Tudjuk, hogy ekkor Σ' is kielégíthető, és így van Herbrand modellje.
- Ez a Herbrand modell megszámlálható (mert fel tudjuk sorolni az összes alaptermet, mert a függvényjelek halmaza is megszámlálható volt)
- Ez a Herbrand modell az eredeti Σ -nak is modellje (azzal, hogy „elfelejtjük” a bevezetett Skolem-függvényeket).

Ez például azt is jelenti, hogy $\mathbb{R}$ nem axiomatizálható még gyengén sem: **nincs olyan Σ formulahalmaz**, melyre $\text{Mod}(\Sigma) = \{\mathbb{R}\}$, tehát melynek (izomorfia erejéig) pontosan csak a valós számok struktúrája az egyetlen modellje.

- Hiszen ha Σ -nak $\mathbb{R}$ egy modellje, akkor kielégíthető.
- Ha pedig kielégíthető, akkor van megszámlálható modellje.
- De $\mathbb{R}$ nem megszámlálható (a valós számokat nem lehet $r_1, r_2, \dots$ felsorolni úgy, hogy ne maradjon ki valaki).

Detour: $\mathbb{R}$ nem megszámlálható

- Egy valós szám a $[0, 1]$ intervallumban: egy $0.b_1b_2b_3b_4\dots$ alakú végtelen string, $b_i \in \{0, 1\}$ (kettes számrendszerben végtelen „tizedes”törtként)
- írjunk fel valós számokat egy $r_1, r_2, r_3, \dots$ sorozatban
- akkor az egész sorozatot egy (végtelen) táblázatba rendezhetjük:

	b_1	b_2	b_3	b_4	b_5	$\dots$
$r_1 =$	0	0	1	0	0	$\dots$
$r_2 =$	0	1	0	1	1	$\dots$
$r_3 =$	1	1	0	0	1	$\dots$
$r_4 =$	0	1	0	0	0	$\dots$
$\dots$						

- és a pirossal jelölt átlóban szereplő $0.0100\dots$ elemet ha minden koordinátán megváltoztatjuk, a kapott $0.1011\dots$ elem egyik sorban sem szerepelhet (hiszen az r_i -től a b_i -ben különbözik)
- tehát pl. ez a valós szám nem jelenhet meg a táblázatban $\Rightarrow$ nem lehet felsorolni a valós számokat (ez a „**diagonális módszer**”)

Herbrand tétel

Ha Σ zárt Skolem normálformák halmaza, akkor legyen

$$E(\Sigma) := \bigcup_{F \in \Sigma} E(F),$$

tehát Σ Herbrand kiterjesztését úgy kapjuk, hogy az összes Σ -beli F formula Herbrand kiterjesztésének vesszük az unióját.

Példa

Ha $\Sigma = \{ \forall x p(x), \forall y \neg p(f(y)) \}$, akkor $E(\Sigma)$ -ban van pl.

- $p(c)$ (az első formula Herbrand kiterjesztéséből, c az új konstansjel)
- $p(f(c))$ (szintén az elsőből)
- $\neg p(f(c))$ (a másodikból)

és $E(\Sigma)$ kielégíthetetlen (hiszen szerepel benne $p(f(c))$ és $\neg p(f(c))$ is).

Következmény

Zárt Skolem normálformák Σ halmaza pontosan akkor kielégíthető, ha $E(\Sigma)$ kielégíthető.

Bizonyítás

Σ kielégíthető

$\Leftrightarrow \Sigma$ -nak van Herbrand modellje

$\Leftrightarrow \mathcal{A} \models \Sigma$ valamilyen $\mathcal{A}$ Herbrand struktúrára

$\Leftrightarrow$ van olyan $\mathcal{A}$ Herbrand struktúra, melyre $\mathcal{A} \models F$ minden $F \in E(\Sigma)$ -ra

$\Leftrightarrow E(\Sigma)$ -nak van Herbrand modellje

$\Leftrightarrow E(\Sigma)$ kielégíthető.

Az elsőrendű logika kompaktsági tétele

- Egy $E(\Sigma)$ Herbrand kiterjesztés atomi formulái $p(t_1, \dots, t_n)$ alakú alapformulák (t_i -k mind alaptermek).
- Egy Herbrand struktúrát eddig úgy tekintettünk, mint amelynek az I interpretációs függvénye egy p/n predikátumjelhez egy $I(p) : T_0^n \rightarrow \{0, 1\}$ predikátumot rendel, majd $p(t_1, \dots, t_n)$ értékét úgy kapjuk, hogy $I(p)$ -be behelyettesítjük $(t_1, \dots, t_n)$ -t.
- Ehelyett úgy is tekinthetjük az interpretációs függvényt, mint ami minden egyes $p(t_1, \dots, t_n)$ alap formulához **közvetlenül** rendel egy $\{0, 1\}$ értéket.
- Tehát egy $E(\Sigma)$ Herbrand kiterjesztésnek egy Herbrand modelljét keresni ugyanaz, mint
 - a $p(t_1, \dots, t_n)$ atomi alapformulák mindegyikét egy **ítélekalkulusi változónak** tekinteni
 - és ennek az itélekalkulusi formulahalmaznak keresni egy modelljét.

Ha vesszük a $\Sigma = \{ \text{even}(0), \forall x (\text{even}(x) \leftrightarrow \neg \text{even}(s(x))) \}$ formulahalmazt, ennek $E(\Sigma)$ Herbrand kiterjesztése:

- $\text{even}(0)$
- $\text{even}(0) \leftrightarrow \neg \text{even}(s(0))$
- $\text{even}(s(0)) \leftrightarrow \neg \text{even}(s(s(0)))$
- ...

Az alaptermek: $0, s(0), s(s(0)), \dots$

Egy Herbrand struktúra tehát ekkor értéket kell adjon az $\text{even}(0)$, $\text{even}(s(0))$, $\text{even}(s(s(0)))$, ... **ítéletkalkulusi** változóknak.

Pl. modell: $\text{even}(s^n(0)) := 1$, ha n páros.

Az elsőrendű logika kompaktsági tétele

Tétel

Elsőrendű logikai formulák tetszőleges Σ halmazára és F formulára $\Sigma \models F$ pontosan akkor igaz, ha már egy **véges** $\Sigma_0 \subseteq \Sigma$ halmazra is $\Sigma_0 \models F$ igaz.

Bizonyítás

- $\Sigma \models F$ pontosan akkor, ha $\Sigma \cup \{\neg F\}$ kielégíthetetlen.
- Ez pontosan akkor, ha $E(\Sigma \cup \{\neg F\})$ kielégíthetetlen.
- Ez pontosan akkor, ha nincs Herbrand modellje.
- Ez pontosan akkor, ha az $E(\Sigma \cup \{\neg F\})$ mint **ítéletkalkulusbeli** formulahalmaz kielégíthetetlen.
- Az ítéletkalkulus kompaktsági tétele szerint ez pontosan akkor, ha van egy véges $\Sigma'_0 \subseteq E(\Sigma \cup \{\neg F\})$ kielégíthetetlen részhalmaza.
- Ez a véges részhalmaz nyilván $\Sigma \cup \{\neg F\}$ -nek egy **véges** Σ_0 részhalmazának Herbrand kiterjesztésében benne van.
- Tehát van olyan véges $\Sigma_0 \subseteq \Sigma$, melyre $\Sigma_0 \models F$.

Az elsőrendű logika kompaktsági tétele

A kompaktsági tétel szerint. . .

- van **félíg eldöntő** kielégíthetlenségi algoritmus elsőrendű logikára, ha Σ rekurzívan felsorolható:
 - az i . iterációban az input Σ első i formulájának vesszük a Herbrand-kiterjesztésének első i elemét (mondjuk)
 - erre a véges halmazra futtatunk egy ítéletkalkulusbeli algoritmust (pl. rezolúciót)
 - ha kielégíthetetlen $\Rightarrow \Sigma$ is az
 - ha nem $\Rightarrow$ következő iteráció
- viszont az is kijön belőle, hogy vannak tulajdonságok, melyeket **nem lehet** elsőrendű logikában kifejezni

Az elsőrendű logika kompaktsági tétele

Pl:

Állítás

Nem lehet kifejezni elsőrendű logikában, hogy „az univerzum véges”.

Bizonyítás

- Tegyük fel, hogy a Σ formulahalmaznak minden $\mathcal{A}$ struktúra modellje, ha univerzuma véges.
- Legyen $F_n = \exists x_1 \exists x_2 \dots \exists x_n (\bigwedge_{i \neq j} x_i \neq x_j)$.
- F_n -t egy struktúra akkor elégíti ki, ha legalább n -elemű.
- Nézzük a $\Delta := \Sigma \cup \{F_n : n \geq 0\}$ formulahalmazt.
- Nyilván Δ -t nem elégíti ki egyetlen véges struktúra sem, hiszen ha $\mathcal{A}$ n -elemű, akkor $\mathcal{A} \not\models F_{n+1}$.

Bizonyítás folytatása

- De Δ minden **véges** részhalmazát kielégíti egy véges struktúra, hiszen egy véges részhalmazban csak véges sok F_i szerepel, és ha n a legnagyobb ezek közül, akkor egy $n + 1$ -elemű struktúra kielégíti az adott véges részhalmazt.
- A kompaktsági tétel szerint ekkor tehát Δ is kielégíthető, tehát van modellje
- Ez a modell végtelen kell legyen
- Mivel pedig $\Sigma \subseteq \Delta$, ez a végtelen struktúra modellje Σ -nak is.

Hasonlóan kijön pl, hogy nem lehet

- gyengén axiomatizálni a természetes számokat: ha $\mathbf{Ar} = \text{Th}(\mathcal{N})$ a számelmélet („aritmetika”), akkor $\mathbf{Ar}$ -nak van olyan modellje, mely nem izomorf $\mathcal{N}$ -nel
- tranzitívan lezárni: ha $F(x, y)$ egy formula, nem feltétlenül tudunk olyan $F^+(x, y)$ formulát felírni, mely pont akkor igaz egy (a, b) párra, ha van olyan $a = x_0, x_2, \dots, x_n = b$ sorozat, melyre $F(x_i, x_{i+1})$ igaz minden i -re
- SQL-ben egy „főnök-közvetlen beosztott” táblából egy queryvel „főnök-beosztott” táblát készíteni (mert ez maga a tranzitív lezárás lenne, az SQL-beli queryk pedig mind kifejezhetők elsőrendű logikában)

Alap rezolúció

(aka „ground rezolúció”)

Az **alap rezolúciós algoritmus**:

- Input: elsőrendű formulák egy Σ halmaza
- Ha Σ kielégíthetetlen, az algoritmus ezt véges sok lépésben levezeti
- Ha kielégíthető, akkor vagy ezt vezeti le, vagy végtelen ciklusba esik
- Módszer:
 - Σ elemeit zárt Skolem alakra hozzuk, a kapott formulák magját CNF-re. Jelölje Σ' a kapott klózalmazt.
 - Ekkor $E(\Sigma')$ a klózok **alap példányainak** halmaza
 - Az $E(\Sigma')$ halmazon futtatjuk az ítéletkalkulus-beli rezolúciós algoritmust.
- Mivel $E(\Sigma')$ általában **végtelen**, így az algoritmus (mondjuk)
 - egy lépésben legenerálja és felveszi $E(\Sigma')$ egy elemét
 - az eddigi klózokkal rezolvenst képez, amíg csak lehet
 - ha közben megkapjuk az üres klózt, Σ kielégíthetetlen
 - különben generáljuk a következő elemet

$$\forall x(p(x) \wedge \neg p(f(x)))$$

Az input már zárt Skolem alakban van, a mag CNF-ben. Klózok:

$$A = \{p(x)\}, \quad B = \{\neg p(f(x))\}$$

Néhány alapterm (fel kell vegyünk egy konstansjelet, mondjuk c -t, mert nincs a nyelvben):

$$T_0 = \{c, f(c), f(f(c)), \dots\}$$

Alap rezolúciós levezetés:

1. $\{p(f(c))\} \quad A[x/f(c)]$
2. $\{\neg p(f(c))\} \quad B[x/c]$
3. $\square \quad \text{Res}(1, 2)$

Alap rezolúció

$$\forall x \forall y \left((\neg p(x) \vee \neg p(f(a)) \vee q(y)) \wedge p(y) \wedge (\neg p(g(b, x)) \vee \neg q(b)) \right)$$

Az input megint zárt Skolem alakban van. Klózok:

$$A = \{\neg p(x), \neg p(f(a)), q(y)\}, \quad B = \{p(y)\}, \quad C = \{\neg p(g(b, x)), \neg q(b)\}$$

Néhány alapterm:

$$T_0 = \{a, b, f(a), f(b), g(a, a), g(a, b), g(f(a), f(b)), \dots\}$$

Levezetés:

- | | | |
|----|-------------------------------------|--------------------|
| 1. | $\{\neg p(f(a)), q(b)\}$ | $A[x/f(a), y/b]$ |
| 2. | $\{\neg p(g(b, a)), \neg q(b)\}$ | $C[x/a]$ |
| 3. | $\{\neg p(f(a)), \neg p(g(b, a))\}$ | $\text{Res}(1, 2)$ |
| 4. | $\{p(f(a))\}$ | $B[y/f(a)]$ |
| 5. | $\{\neg p(g(b, a))\}$ | $\text{Res}(3, 4)$ |
| 6. | $\{p(g(b, a))\}$ | $B[y/g(b, a)]$ |
| 7. | $\square$ | $\text{Res}(5, 6)$ |

Alap rezolúció

Tétel

Az alap rezolúciós algoritmus helyes és teljes.

Tehát pontosan akkor igaz $\Box \in \text{Res}^*(E(\Sigma'))$, ha $\Sigma \models \downarrow$.

Ez igaz, hiszen

- a zárt Skolem alakra hozás s -ekvivalens átalakítás, tehát Σ pontosan akkor kielégíthetetlen, ha Σ' az;
- a Herbrand-tétel következménye szerint Σ' pontosan akkor kielégíthetetlen, ha $E(\Sigma')$ az;
- az ítéletkalkulus kompaktsági tétele szerint $E(\Sigma')$ pontosan akkor kielégíthetetlen, ha van egy véges Σ_0 kielégíthetetlen részhalmaza;
- a rezolúciós algoritmus teljessége szerint ha a Σ_0 véges klózalmaz kielégíthetetlen, akkor az algoritmus ezt
- tehát ha Σ kielégíthetetlen, akkor az algoritmus leáll ezzel a válasszal akkor, amikor egy ilyen Σ_0 halmaznak már legenerálta az összes elemét (és rezolvenseiket, köztük $\Box$ -t)

Az alap rezolúció alkalmazásakor **nagy a keresési tér**.

(Minden változót helyettesítenünk egy-egy alaptermmel.)

Az **elsőrendű rezolúció**ban

- a keresési tér kisebb lesz
- ezzel párhuzamosan, a rezolvens képzési algoritmus bonyolultabbá válik

A módszerhez először meg kell ismerjünk az **egyesítési algoritmust**.

Egyesítés

- Azt már láttuk, hogy ha F egy formula, x egy változó, t pedig egy term, akkor mit jelent $F[x/t]$.
- Ha $x_1, \dots, x_n$ változók és $t_1, \dots, t_n$ termek, akkor az $[x_1/t_1][x_2/t_2] \dots [x_n/t_n]$ **helyettesítést** úgy végezzük el, hogy először az $[x_1/t_1]$ -et, majd az eredményen az $[x_2/t_2]$ -t, $\dots$, végül az $[x_n/t_n]$ -t.

Általában az ilyen helyettesítési sorozatokat s -sel fogjuk jelölni.

Formálisan n szerinti **indukcióval**: ha F formula, $s = [x_1/t_1] \dots [x_n/t_n]$ helyettesítés, akkor $F \cdot s$:

- ha $n = 0$ (nincs is helyettesítés), akkor $F \cdot s := F$;
- ha $n > 0$, akkor $F \cdot s := (F \cdot [x_1/t_1] \dots [x_{n-1}/t_{n-1}]) \cdot [x_n/t_n]$
(vagyis: előbb elvégezzük az első $n - 1$ helyettesítést, majd ez után az eredményen az utolsót)

Az $n = 0$ esetet (az „üres” helyettesítést) $[]$ -vel is jelöljük (nem összekeverni a $\square$ üres klózzal. $\dots$)

Egyesítés

A sorrend számíthat, ha valamelyik t_i -ben szerepel valamelyik x_j :

$$f(x, g(y))[x/g(y)][y/a] = f(g(a), g(a))$$

$$f(x, g(y))[y/a][x/g(y)] = f(g(y), g(a))$$

Formulák **halmazaira** (pl klózokra) is értelmezzük a helyettesítést: ha C formulák egy halmaza és s egy helyettesítés, akkor legyen

$$C \cdot s := \{F \cdot s : F \in C\}$$

azaz $C \cdot s$ -et úgy kapjuk, hogy C minden elemén elvégezzük s -t és az eredményeket egy halmazba rakjuk

Ha $C = \{\ell_1, \dots, \ell_n\}$ literálok egy halmaza (azaz egy klóz), akkor s a C **egyesítője**, ha $\ell_1 s = \dots = \ell_n s$.

A C klózra azt mondjuk, hogy **egyesíthető**, ha van egyesítője.

$$C = \left\{ p(g(x), y), p(y, g(a)) \right\}$$

Ez a klóz egyesíthető, egy egyesítője

$$s = [x/a][y/g(a)]$$

hiszen

$$p(g(x), y) \cdot [x/a][y/g(a)] = p(g(a), g(a))$$

$$p(y, g(a)) \cdot [x/a][y/g(a)] = p(g(a), g(a))$$

$$C = \left\{ p(x, f(y)), p(g(y), z) \right\}$$

egyesíthető, egy egyesítője pl.

$$s = [x/g(a)][y/a][z/f(a)]$$

mely melletti képe $C \cdot s = \{p(g(a), f(a))\}$.

Egy másik egyesítő:

$$s_0 = [x/g(y)][z/f(y)]$$

mely melletti képe $C \cdot s_0 = \{p(g(y), f(y))\}$.

Észrevehetjük: $s_0 \cdot [y/a] = s$ és emiatt valamilyen értelemben s_0 „tűnik” a „jobb” egyesítőnek.

Azt mondjuk, hogy az s helyettesítés **általánosabb** az s' helyettesítésnél, ha van olyan s'' helyettesítés, melyre $s \cdot s'' = s'$.

Pl. az előző dián látott s_0 általánosabb volt az s -nél.

Az **egyesítési algoritmus**

- inputja egy C klóz,
- outputja:
 - ha C egyesíthető, akkor egy **legáltalánosabb egyesítőjét** adja vissza;
 - különben azzal tér vissza, hogy nem egyesíthető.

Mikor nem egyesíthető egy klóz?

$$\left\{ p(f(x), y), p(g(x), y) \right\}$$

nem egyesíthető: bármit is helyettesítünk a **változók** helyébe, az első term az első literálban f -fel, a másodikban g -vel fog kezdődni.

$$\left\{ p(x), p(f(x)) \right\}$$

nem egyesíthető: bármit is helyettesítünk x helyébe, mondjuk t -t, az első literál $p(t)$ lesz, a második $p(f(t))$ és $t \neq f(t)$.

Az egyesítési algoritmus tehát, ha inputja egy C klóz:

- $s := []$
- Ha $|C| \leq 1$, adjuk vissza s -t.
- Vegyünk két literált, $\ell_1 \neq \ell_2$ -t C -ből és keressük meg az első eltérő pozíciójukat.
- Ha itt
 - az egyik literálban egy x változó áll,
 - a másikban egy t term, melyben **nincs** x ,akkor legyen $C := C \cdot [x/t]$, $s := s \cdot [x/t]$ és ugorjunk a kettes pontra.
- Különben adjuk vissza, hogy C nem egyesíthető.

$$C = \left\{ \neg p(f(z, g(a, y)), h(z)), \neg p(f(f(u, v), w), h(f(a, b))) \right\}$$

- $s = []$
- Az első eltérő pozíció: z vs. $f(u, v)$, OK, z olyan változó, mely nem szerepel $f(u, v)$ -ben
- akkor $s = [z/f(u, v)]$, elvégezzük C -n:

$$\left\{ \neg p(f(f(u, v), g(a, y)), h(f(u, v))), \neg p(f(f(u, v), w), h(f(a, b))) \right\}$$

$$\left\{ \neg p(f(f(u, v), g(a, y)), h(f(u, v))), \neg p(f(f(u, v), w), h(f(a, b))) \right\}$$

- w változó, $g(a, y)$ -ben nem szerepel $\Rightarrow$ még mindig OK
- $s = [z/f(u, v)][w/g(a, y)]$
- elvégezzük $[w/g(a, y)]$ -t az aktuális C -n:

$$\left\{ \neg p(f(f(u, v), g(a, y)), h(f(u, v))), \neg p(f(f(u, v), g(a, y)), h(f(a, b))) \right\}$$

A végeredmény $[z/f(u, v)][w/g(a, y)][u/a][v/b]$ lesz.

Az elsőrendű rezolúciós algoritmusban

- a Σ' -beli klózokat **közvetlenül** felvehetjük a listára
- viszont használjuk az **egyesítési algoritmust** is a rezolvensképzésnél (nem csak akkor rezolválhatunk két literál mentén, ha betű szerint egymás komplementerei)

Két **elsőrendű logikai** klóz, C_1 és C_2 **elsőrendű rezolvensét** a következőképp kapjuk:

- **Átnevezzük** a klózekban a változókat úgy (legyenek a változóátnevezések s_1 és s_2), hogy a kapott $C_1 \cdot s_1$ és $C_2 \cdot s_2$ klózek ne tartalmazzanak közös változót.
- **Kiválasztunk** $C_1 \cdot s_1$ -ből $\ell_1, \dots, \ell_m$ és $C_2 \cdot s_2$ -ből $\ell'_1, \dots, \ell'_n$ literálokat, mindkettőből legalább egyet-egyet.
- Futtatjuk az **egyesítési** algoritmust a

$$C = \{ \ell_1, \dots, \ell_m, \overline{\ell'_1}, \dots, \overline{\ell'_n} \}$$

klózon. (Tehát a C_1 -ből jövő kiválasztott literálokon és a C_2 -ből jövőök komplementerein.)

- Ha C egyesíthető az s legáltalánosabb egyesítővel, akkor s -et **végrehajtjuk** a nem kiválasztott literálok halmazán:

$$R := \left((C_1 \cdot s_1 - \{\ell_1, \dots, \ell_m\}) \cup (C_2 \cdot s_2 - \{\ell'_1, \dots, \ell'_n\}) \right) \cdot s$$

A kapott R klóz a C_1 és C_2 egy **elsőrendű rezolvense**.

Elsőrendű rezolúció

A literálok kiválasztásánál érdemes észrevenni a következőt:

- Csak akkor tudunk a kiválasztott literálok mentén egyesíteni, ha a

$$C = \{ \ell_1, \dots, \ell_m, \overline{\ell'_1}, \dots, \overline{\ell'_n} \}$$

klóz egyesíthető.

- Ehhez az mindenképp kell, hogy az összes kiválasztott literálban ugyanaz legyen a **predikátumjel**. (Hiszen ha már azon is eltérnek, akkor C nem lesz egyesíthető.)
- Az is kell, hogy a C -be kerülő literálok **előjele** megegyezzen.
- Tehát a kiválasztási fázisban (mivel C_1 és C_2 szerepe szimmetrikus) elég:
 - választanunk egy p predikátumjelet
 - $C_1 \cdot s_1$ -ből **pozitív** p -s literálokat választani, legalább egyet,
 - $C_2 \cdot s_2$ -ből **negatív** p -s literálokat választani, legalább egyet,
 - és ezeknek az előjel nélküli változatát megpróbálni egyesíteni.

- Ebben a formájában ha összehasonlítjuk az ítéletkalkulus-beli rezolvensképzéssel, ott:
- Nem kell átneveznünk változókat, hiszen nincsenek a klózokban elsőrendű változók. $s_1 = s_2 = []$
- Kiválasztva egy p predikátumjelet mindkét klózban csak a p és a $\neg p$ szerepelhet p -s literálként;
- Így csak úgy tudunk rezolvenst képezni, ha $p \in C_1$ és $\neg p \in C_2$, ekkor $C = \{p\}$ lesz, ami egyesíthető az $s = []$ helyettesítéssel;
- Ekkor R az ítéletkalkulus-beli rezolvens lesz.
- Tehát ítéletkalkulusban a két rezolvensképző módszer megegyezik; az elsőrendű rezolvensképzés **kiterjeszti** az ítéletkalkulus-belit.

Elsőrendű rezolúció

$$C_1 = \{p(f(x)), \neg q(z), p(z)\}$$

$$C_2 = \{\neg p(x), r(g(x), a)\}$$

- Átnevezés: (mondjuk) a C_2 -beli x -et y -ra nevezzük át

$$C_1 s_1 = \{p(f(x)), \neg q(z), p(z)\}$$

$$C_2 s_2 = \{\neg p(y), r(g(y), a)\}$$

- Kiválasztás: válasszuk mondjuk $C_1 s_1$ -ből $p(f(x))$ -et és $p(z)$ -t, $C_2 s_2$ -ből pedig $\neg p(y)$ -t
- Egyesítés: a

$$\{p(f(x)), p(z), p(y)\}$$

klóz legáltalánosabb egyesítője $s = [z/f(x)][y/f(x)]$

- Végrehajtás: a nem-kiválasztott literálokon végrehajtjuk s -t:

$$\{\neg q(z), r(g(y), a)\} \cdot [z/f(x)][y/f(x)] = \{\neg q(f(x)), r(g(f(x)), a)\}$$

Ez a két klóz (egyik) rezolvense.

Elsőrendű rezolúció

Az elsőrendű rezolúciós algoritmus tehát:

- Input: (elsőrendű) klózok egy Σ halmaza. Úgy tekintjük, mintha a Σ -beli klózok változói univerzálisan lennének kvantálva.
- Output:
 - ha $\Sigma \models \downarrow$, akkor „kielégíthetetlen”
 - különben „kielégíthető” vagy végtelen ciklus
- Módszer: listát vezetünk klózokról. Egy klózt felvehetünk, ha:
 - Σ -beli vagy
 - két, már a listán szereplő klóz rezolvense.
- Ha $\square$ rákerül a listára, akkor Σ kielégíthetetlen.
- Különben, ha már nem tudunk több klózt levezetni, Σ kielégíthető.

Mint korábban is, $\text{Res}(\Sigma)$ jelöli azt a halmazt, mely tartalmazza Σ elemeit és a belőlük egy rezolvensképzéssel levezethető klózokat; $\text{Res}^*(\Sigma)$ pedig a Σ -ból rezolúcióval levezethető összes klóz halmazát.

$$\begin{aligned}
 & \forall x \forall y \forall z \left((\neg p(x) \vee q(x) \vee r(x, f(x))) \right. \\
 & \quad \wedge (\neg p(x) \vee q(x) \vee s(f(x))) \\
 & \quad \wedge p(a) \wedge t(a) \wedge (\neg r(a, z) \vee t(z)) \\
 & \quad \left. \wedge (\neg t(x) \vee \neg q(x)) \wedge (\neg t(y) \vee \neg s(y)) \right)
 \end{aligned}$$

- | | | | | | |
|----|-----------------------------------|--------------|-----|-------------------------------------|-------------|
| 1. | $\{\neg p(x), q(x), r(x, f(x))\}$ | $\in \Sigma$ | 8. | $\{q(a), r(a, f(a))\}$ | Res(1, 3) |
| 2. | $\{\neg p(x), q(x), s(f(x))\}$ | $\in \Sigma$ | 9. | $\{\neg q(a)\}$ | Res(4, 6) |
| 3. | $\{p(a)\}$ | $\in \Sigma$ | 10. | $\{r(a, f(a))\}$ | Res(8, 9) |
| 4. | $\{t(a)\}$ | $\in \Sigma$ | 11. | $\{\neg p(x), q(x), \neg t(f(x))\}$ | Res(2, 7) |
| 5. | $\{\neg r(a, z), t(z)\}$ | $\in \Sigma$ | 12. | $\{q(a), \neg t(f(a))\}$ | Res(3, 11) |
| 6. | $\{\neg t(x), \neg q(x)\}$ | $\in \Sigma$ | 13. | $\{\neg t(f(a))\}$ | Res(9, 12) |
| 7. | $\{\neg t(y), \neg s(y)\}$ | $\in \Sigma$ | 14. | $\{\neg r(a, f(a))\}$ | Res(5, 13) |
| | | | 15. | $\square$ | Res(10, 14) |

Az elsőrendű rezolúciónak is van helyességi és teljességi tétele:

Elsőrendű klózok Σ halmaza pontosan akkor kielégíthetetlen, ha $\square \in \text{Res}^*(\Sigma)$.

Elsőrendű rezolúció: helyesség

- A helyesség (ha kijöhet az üres klóz, akkor Σ kielégíthetetlen) ismét a rezolvensképzés helyességéből következik.
- Mivel a klózik univerzálisan kvantáltak (a Skolem alakból), így tetszőleges C klózra és s helyettesítésre $C \models C \cdot s$
- Tehát a rezolvensképzésnél felírt C_1 -nek $C_1 \cdot s_1 \cdot s$, C_2 -nek pedig $C_2 \cdot s_2 \cdot s$ egy-egy logikai következménye
- Tehát $\{C_1, C_2\} \models \{C_1 s_1 s, C_2 s_2 s\}$
- Ennek a két klóznak pedig a rezolvens következménye (az „eredeti” rezolúciós következtetés szerint).

Elsőrendű rezolúció: teljesség

- A teljességi irányhoz felhasználjuk az alap rezolúciós algoritmus teljességét.
- Tehát: ha Σ kielégíthetetlen, akkor az üres klóznak van egy $C'_1, C'_2, \dots, C'_n = \square$ **alap**rezolúciós levezetése.
- Ebből az alaprezolúciós levezetésből fogunk készíteni egy $C_1, C_2, \dots, C_n$ **elsőrendű** rezolúciós levezetést.
- A klózokat úgy fogjuk elkészíteni **indukcióval** n szerint, hogy minden i -re a C_i -nek a C'_i egy (alap) példánya lesz.
- Mivel a $C'_n = \square$ üres klóz csak önmagának példánya, így $C_n = \square$ kell legyen.

Elsőrendű rezolúció: teljesség

Ha pl. Σ -ban:

$$\{p(x, f(y))\} \quad \{\neg p(f(x), z), r(x, g(z))\} \quad \{\neg r(f(y), g(y)), \neg p(y, y)\}$$

Akkor:

C'_i		C_i
1. $\{p(fffc, fc)\}$	$\in E(\Sigma)$	$\{p(x, f(y))\}$
2. $\{\neg p(fffc, fc), r(ffc, gfc)\}$	$\in E(\Sigma)$	$\{\neg p(f(x), z), r(x, g(z))\}$
3. $\{r(ffc, gfc)\}$	Res(1, 2)	$\{r(x, g(f(y)))\}$
4. $\{\neg r(ffc, gfc), \neg p(fc, fc)\}$	$\in E(\Sigma)$	$\{\neg r(f(y), g(y)), \neg p(y, y)\}$
5. $\{\neg p(fc, fc)\}$	Res(3, 4)	$\{\neg p(f(z), f(z))\}$
6. $\{p(fc, fc)\}$	$\in E(\Sigma)$	$\{p(x, f(y))\}$
7. $\square$	Res(5, 6)	$\square$

Elsőrendű rezolúció: teljesség

Tehát az állítás:

Ha $C'_1, C'_2, \dots, C'_n$ egy Σ fölötti alap rezolúciós levezetés, akkor van olyan $C_1, C_2, \dots, C_n$ szintén Σ fölötti elsőrendű rezolúciós levezetés, melyben minden i -re a C'_i a C_i -nek egy alap példánya.

- Ha $C'_i \in E(\Sigma)$, azaz C'_i egy Σ -beli C klóz (alap) példánya, akkor legyen $C_i := C$.
- A másik lehetőség, hogy C'_i a C'_j és C'_k klózok, $j, k < i$, egy rezolvense.
- Ennek az esetnek a belátásához felírjuk az ún. **lift lemmát**.

A lift lemma

A lift lemma

Ha C_1 -nek C'_1 , C_2 -nek pedig C'_2 alap példányai, melyeknek R' rezolvense, akkor van C_1 -nek és C_2 -nek olyan **elsőrendű** R rezolvense, melynek R' alap példánya.

Ha ezt belátjuk, akkor kész vagyunk: ha ugyanis C'_i a C'_j és C'_k klózik alap rezolvense és C_j -nek C'_j , C_k -nak pedig C'_k alap példánya, akkor a lift lemma szerint C_j -nek és C_k -nak van egy C elsőrendű rezolvense, melynek C'_i alap példánya; legyen ez a C_i .

A lift lemma

Bizonyítás

- Legyen $\ell \in C'_1$, $\bar{\ell} \in C'_2$ és $R' = (C'_1 - \{\ell\}) \cup (C'_2 - \{\bar{\ell}\})$.
- Tudjuk, hogy C_1 -nek C'_1 , C_2 -nek pedig C'_2 egy alap példánya.
- Ha átnevezzük a változókat a klózokban, úgy a kapott C_1s_1 -nek ill. C_2s_2 -nek is alap példányai lesznek C'_1 és C'_2 .
- Vegyünk tehát egy s_1 és egy s_2 változó-átnevezést, melyre C_1s_1 -ben és C_2s_2 -ben nincs közös változó.
- Akkor van egy olyan **közös** s alaphelyettesítés, hogy $C_1s_1s = C'_1$ és $C_2s_2s = C'_2$.
- Tehát mivel $\ell \in C'_1$, ezért vannak olyan $\ell_1, \dots, \ell_n \in C_1s_1$, $n \geq 1$ literálok, melyekre $\{\ell_1, \dots, \ell_n\}s = \ell$.
- Ugyanígy, vannak olyan $\ell'_1, \dots, \ell'_m \in C_2s_2$, $m \geq 1$ literálok, melyekre $\{\ell'_1, \dots, \ell'_m\}s = \bar{\ell}$.
- Vegyük be az összes ilyen, tehát a többi literál s melletti képe legyen egy ℓ -től eltérő.

Bizonyítás folytatása

- Akkor az $\{\ell_1, \dots, \ell_n, \overline{\ell'_1}, \dots, \overline{\ell'_m}\}$ literálhalmaz egyesíthető, legyen mondjuk az s_0 legáltalánosabb egyesítője.
- Mivel az s_0 a legáltalánosabb egyesítő és s egy egyesítő, így van olyan s' helyettesítés, melyre $s_0 s' = s$.
- Ekkor az R' rezolvens az

$$R = \left((C_1 s_1 - \{\ell_1, \dots, \ell_n\}) \cup (C_2 s_2 - \{\ell'_1, \dots, \ell'_m\}) \right) s_0$$

elsőrendű rezolvensnek s' melletti alap példánya.

A lift lemma – példa

- $C_1 = \{p(x), p(f(y)), q(z)\}$
- $C_2 = \{\neg p(y), r(x)\}$
- $C'_1 = \{p(f(c)), q(g(c))\}$
- $C'_2 = \{\neg p(f(c)), r(c)\}$
- $R' = \{q(g(c)), r(c)\}$
- Ekkor:
- $C_1 s_1 := \{p(x), p(f(y)), q(z)\} \vee$
- $C_2 s_2 := \{\neg p(w), r(u)\}$
- $s := [x/f(c)][y/c][z/g(c)][w/f(c)][u/c]$
- Az s helyettesítés egyesíti a $C = \{p(x), p(f(y)), p(w)\}$ literálokat
- Ezekre $s_0 = [x/f(y)][w/f(y)]$ a legáltalánosabb
- A rezolvens $R = \{q(z), r(u)\}$
- Ennek R' tényleg alap példánya.

A logikai programozás alapfeladata:

- Input: **elsőrendű**, univerzálisan kvantált Horn-klózik egy véges Σ halmaza és egy $R = \exists(R_1 \wedge \dots \wedge R_n)$ alakú formula, ahol az R_i -k atomi formulák.
- Output: Igaz-e, hogy $\Sigma \models R$?

(Itt $\exists F$ az F formula **egzisztenciális lezártját** jelöli: az F -beli összes szabadon előforduló változót lekötjük egy-egy $\exists$ kvantorral.)

actual Prolog syntax

- Legyen `|` egy bináris függvényjel, `0` és `[]` konstans, `s` unáris.
- A már megszokott Peano reprezentációban a természetes számok: pl. `ss0` a 2, `ssss0` a 4.
- a `|` függvényjellel **listákat** építünk, pl. `1|(4|(2|(8|[])))` az `[1, 4, 2, 8]` lista.
- A listákat a fenti módon jobbra igazítjuk és nem teszünk ki zárójeleket.
- Syntax sugar: `[1, 4, 2|[8, 5, 7]]` az `[1, 4, 2, 8, 5, 7]` lista.

A Prolog – példa

Lista összegzés Peano számokkal

Legyenek a következő formuláink (univerzálisan kvantált Horn-klózek) Σ -ban:

- $\text{sum}([], 0)$ üres lista összege 0
- $\text{sum}(x|y, z) \rightarrow \text{sum}(s(x)|y, s(z))$
ha az ' $x|y$ ' lista összege ' z ', akkor az ' $[x+1 | y]$ ' lista összege ' $z+1$ '
- $\text{sum}(y, z) \rightarrow \text{sum}(0|y, z)$
ha az ' y ' lista összege ' z ', akkor a ' $[0|y]$ ' lista összege is ' z '

Ez nálunk:

$$\{\text{sum}([], 0)\}, \{\neg \text{sum}(x|y, z), \text{sum}(s(x)|y, s(z))\}, \{\neg \text{sum}(y, z), \text{sum}(0|y, z)\}$$

Prologban:

visszafele van a „nyíl”, jobb oldalon éselés, vesszőkkel elválasztva

- $\text{sum}([], 0).$
- $\text{sum}([s(X) | Y], s(Z)) :- \text{sum}([X | Y], Z).$
- $\text{sum}([0 | Y], Z) :- \text{sum}(Y, Z).$ prologban a változók nagybetűvel kezdődnek

A Prolog – példa

Query

- **megkérdezzük**, hogy ebből a Σ -ból következik-e, hogy

$$\exists x(\text{sum}(1|4|2, x))$$

(azaz: van-e olyan x , amire x az $1|4|2$ lista összege):

- $?\text{-sum}([s0|[ssss0|[ss0|[]]]], X)$

persze az összegzést direkt egy funkcionális predikátumnak készítettük, persze, hogy van; a Prolog vissza is ad egyet amire igaz, ha le tudja vezetni h létezik

Erre a Prolog „illeszt”:

- az első szabály **feje** nem illik $\text{sum}([], 0)$
- a másodiké illik: az aktuális állapotunk $\text{sum}([s(x)|y], s(z)) \text{ :- } \text{sum}([x|y], z)$

$$\text{sum}([0|[ssss0|[ss0|[]]]], x')$$

lesz

eltároljuk az $[x/s(x')]$ egyesítőt is, visszatérésnél használjuk majd

aktuális állapot

- $\{\neg \text{sum}(s(0)|s(s(s(s(0))))|s(s(0))|[], x)\}$
- $\{\text{sum}(s(x)|y, s(z)), \neg \text{sum}(x|y, z)\}$
- $\{\text{sum}(s(x')|y, s(z)), \neg \text{sum}(x'|y, z)\}$
- egyesítő: $[x'/0][y/ssss0|ss0|[]][x/s(z)]$
- rezolvens: $\{\neg \text{sum}(0|ssss0|ss0|[], z)\}$

átnevezés a programklózbán

erre pedig „illik a $\text{sum}([0|y], z) :- \text{sum}(y, z)$. szabály”

aktuális állapot

- $\{\neg \text{sum}(0|ssss0|ss0|[], z)\}$
- $\{\text{sum}(0|y, z), \neg \text{sum}(y, z)\}$
- $\{\text{sum}(0|y, z'), \neg \text{sum}(y, z')\}$
- egyesítő most: $[y/ssss0|ss0|[]][z'/z]$
- rezolvens: $\{\neg \text{sum}(ssss0|ss0|[], z)\}$
- egyesítő **eddig**: $[x'/0][y/ssss0|ss0|[]][x/s(z)][y/ssss0|ss0|[]][z'/z]$
- egyesítőnek az eredeti klózra ható része: $[x/s(z)]$

átnevezés a programklózbán

aktuális állapot

- $\{\neg \text{sum}(ssss0|ss0|[], z)\}$
- $\{\text{sum}(s(x)|y, s(z)), \neg \text{sum}(x|y, z)\}$
- $\{\text{sum}(s(x)|y, s(z')), \neg \text{sum}(x|y, z')\}$
- egyesítő: $[x/sss0][y/ss0|[]][z/s(z')]$
- rezolvens: $\{\neg \text{sum}(sss0|ss0|[], z')\}$
- eddigi egyesítőnek az eredeti klózra ható része:
 $[x/s(z)] \cdot [x/sss0][y/ss0|[]][z/s(z')] = [x/ss(z')]$

átnevezés a programklózbán

aktuális állapot

- $\{\neg \text{sum}(sss0|ss0|[], z')\}$
- $\{\text{sum}(s(x)|y, s(z)), \neg \text{sum}(x|y, z)\}$
- egyesítő: $[x/ss0][y/ss0|[]][z'/s(z)]$
- rezolvens: $\{\neg \text{sum}(ss0|ss0|[], z)\}$
- eddigi egyesítőnek az eredeti klózra ható része:
 $[x/ss(z')] \cdot [x/ss0][y/ss0|[]][z'/s(z)] = [x/sss(z)]$

A Prolog – példa

idhővel...

- $\{\neg \text{sum}(0|\square, z)\}$
- $\{\text{sum}(0|y, z), \neg \text{sum}(y, z))\}$
- $\{\text{sum}(0|y, z'), \neg \text{sum}(y, z'))\}$
- egyesítő: $[y/\square][z/z']$
- rezolvens: $\{\neg \text{sum}(\square, z')\}$
- eddigi egyesítőnek az eredeti klózra ható része:
 $[x/sssssss(z)] \cdot [y/\square][z/z'] = [x/sssssss(z')]$

- $\{\neg \text{sum}(\square, z')\}$
- $\{\neg \text{sum}(\square, 0)\}$
- egyesítő: $[z'/0]$
- rezolvens: $\square$
- eddigi egyesítőnek az eredeti klózra ható része:
 $[x/sssssss(z')]\cdot [z'/0] = [x/sssssss0]$

Valójában a Prolog egy SLD rezolúciót végez:

- A **kérdés** negáltját bevesszük Σ -ba – kapunk egy univerzálisan kvantált negatív klózt
- Ebből a negatív klózból indítunk egy lineáris rezolúciós levezetést
- Mindig jegyezzük a helyettesítést, hogy **eddig** mit hajtottunk végre a kérdés klózon
- Ha kijön az üres klóz, alkalmazzuk a kérdés klózon a helyettesítést, ez a válasz

note: ha átneveznénk a munkaklózunk változóit rezolváláskor, azt is bele kell írni a logba

A logikai programozás

Tehát a logikai programozásban

- Adott **program klózoknak** vagy **definit klózoknak** egy Σ véges halmaza (ezek nem negatív Horn klózok)
- Adott még egy R **kérdés klóz** (ez pedig egy negatív Horn klóz)
- SLD rezolúciót végzünk úgy, hogy **megjegyezzük a helyettesítést** is
- Tehát egy **konfiguráció** egy (C, s) pár, ahol C negatív klóz, s pedig helyettesítés
- A kiindulási konfiguráció: $(R, [])$
- Egy **elfogadó** konfiguráció: $(\square, s)$ valamilyen s -re
- Egy **átmenet** vagy **lépés**: $(C, s) \vdash (C', s')$, ha van olyan D program klóz, mellyel C -nek rezolvense C' és a rezolvens képzésekor kapott s'' legáltalánosabb egyesítőre $ss'' = s'$
- Ha $(R, []) \vdash^* (\square, s)$, akkor az **eredmény** Rs .

Az SLD rezolúció Horn-klózkodra vonatkozó teljességéből kapjuk:

- Ha $\Sigma \models \exists R$, akkor (és csak akkor) van $(R, []) \vdash \dots \vdash (\Box, s)$ alakú (azaz **sikeres**) **kiszámítás**.
- Ekkor $\Sigma \models Rs$ is igaz (tehát még az egzisztenciális kvantorok által kötött változóknak is megkapjuk egy-egy „jó” helyettesítését).
- Továbbá, ha $\Sigma \models Rs$, akkor van olyan sikeres $(R, []) \vdash^* (\Box, s')$ kiszámítás, ahol s' legalább olyan általános helyettesítés, mint s .

Megjegyzés: a Prolog az oldal klózkod kipróbálásának egy **sorrendjét** is rögzíti (a fejeket deklarációs sorrendben próbálja végig). Ily módon végtelen ciklusba is eshet akkor is, ha matematikailag lenne sikeres kiszámítás.

```
f(L, R) :- f(L, [], R).
```

```
f([], R, R).
```

```
f([H|T], R1, R) :- f(T, [H|R1], R).
```

```
?- f([1,4,2], X).
```

```
f([1,4,2], [], X) %masodik rule, H/1, T/[4,2], R1/[], R/X
```

```
f([4,2], [1], X) %masodik rule, H/4, T/[2], R1/[1], R/X
```

```
f([2], [4,1], X) %masodik rule, H/2, T/[], R1/[4,1], R/X
```

```
f([], [2,4,1], X) %első rule, R/[2,4,1], X/[2,4,1]
```

```
%ures kloz
```

```
--> X = [2,4,1]
```

Prolog

```
e(1,2). e(2,3). e(3,1). e(2,4).  
e(4,5). e(5,4).
```

```
path(X,X).  
path(X,Y) :- e(X,Z), path(Z,Y).
```

math: OK, de SLD miatt végtelen ciklus

1. $\{\neg path(1, 4)\}$
2. $\{\neg e(1, z), \neg path(z, 4)\}$ $\leftarrow \{\neg e(x, z), \neg path(z, y), path(x, y)\}$
3. $\{\neg path(2, 4)\}$ $\leftarrow \{e(1, 2)\}$
4. $\{\neg e(2, z), \neg path(z, 4)\}$ $\{\leftarrow \{\neg e(x, z), \neg path(z, y), path(x, y)\}$
5. $\{\neg path(4, 4)\}$ $\leftarrow \{e(2, 4)\}!!!$
6. $\square$ $\leftarrow \{\neg path(x, x)\}$

Prolog

```
e(1,2). e(2,3). e(3,1). e(2,4). e(4,5). e(5,4). %input graf
```

```
path(X,X).
```

```
path(X,Y) :- e(X,Z), path(Z,Y).
```

math: OK, de SLD miatt végtelen ciklus

1. $\{\neg path(1,4)\}$
2. $\{\neg e(1,z), \neg path(z,4)\}$ $\leftarrow \{\neg e(x,z), \neg path(z,y), path(x,y)\}$
3. $\{\neg path(2,4)\}$ $\leftarrow \{e(1,2)\}$
4. $\{\neg e(2,z), \neg path(z,4)\}$ $\leftarrow \{\neg e(x,z), \neg path(z,y), path(x,y)\}$
5. $\{\neg path(3,4)\}$ $\leftarrow \{e(2,3)\}$
6. $\{\neg e(3,z), \neg path(z,4)\}$ $\leftarrow \{\neg e(x,z), \neg path(z,y), path(x,y)\}$
7. $\{\neg path(1,4)\}$ $\leftarrow \{e(3,1)\}$ loop!

`split([], [], []).`

`split([X], [X], []).`

`split([A,B|T], [A|L], [B|R]) :- split(T,L,R).`

1. $\{\neg split([1, 4, 2, 8, 5, 7], x, y)\}$

2. $\{\neg split([2, 8, 5, 7], y', z)\}$ $\leftarrow \{\neg split(x, y, z), split([y_1, z_1|x], [y_1|y], [z_1|z])\}$

- átnevezés a program klózban: $[x/x'] [y/y']$
- egyesítő: $[y_1/1] [z_1/4] [x'/[2, 8, 5, 7]] [x/[1|y']] [y/[4|z]]$
- hatása az eredetin eddig: $[x/[1|y']] [y/[4|z]]$

3. $\{\neg split([5, 7], y, z')\}$ $\leftarrow \{\neg split(x, y, z), split([y_1, z_1|x], [y_1|y], [z_1|z])\}$

- átnevezés a program klózban: $[z/z']$
- egyesítő: $[y_1/2] [z_1/8] [x/[5, 7]] [y'/[2|y]] [z/[8|z']]$
- hatása az eredetin eddig:
 $[x/[1|y']] [y/[4|z]] [y'/[2|y]] [z/[8|z']] = [x/[1, 2|y]] [y/[4, 8|z']]$

`split([], [], []).`

`split([X], [X], []).`

`split([A,B|T], [A|L], [B|R]) :- split(T,L,R).`

1. $\{\neg split([], y', z)\}$ $\leftarrow \{\neg split(x, y, z), split([y_1, z_1|x], [y_1|y], [z_1|z])\}$

- átnevezés a program klózban: $[y/y']$
- egyesítő: $[y_1/5][z_1/7][x/[]][y/[5|y']][z'/[7|z]]$
- hatása az eredetin eddig

$$[x/[1, 2|y]][y/[4, 8|z']][y/[5|y']][z'/[7|z]] = [x/[1, 2, 5|y']][y/[4, 8, 7|z]]$$

2. $\square$ $\leftarrow \{split([], [], [])\}$

- egyesítő: $[y'/[]][z/[]]$
- hatása az eredetin: $[x/[1, 2, 5]][y/[4, 8, 7]]$

$$\Rightarrow X = [1, 2, 5], Y = [4, 8, 7]$$

Prolog

Ha több fej is illik az aktuális klózunkra, backtrack is lehet

```
parent(joe, john). parent(joe, mary)
male(joe). male(john). female(mary).
mother(X, Y) :- parent(X, Y), female(Y).
?- mother(joe, X).
```

- $\{\neg mother(joe, X)\}$
- $\{\neg parent(joe, Y), \neg female(Y)\}$ $\leftarrow \{\neg parent(X, Y), \neg female(Y), mother(X, Y)\}$
- $\{\neg female(john)\}$ $\leftarrow \{parent(joe, john)\}$
stuck, visszalépés, következő lehetséges fej illesztése
- $\{\neg female(mary)\}$ $\leftarrow \{parent(joe, mary)\}$
- $\square$ $\leftarrow \{female(mary)\}$ OK

$X = mary$

Amiben a Prolog többet tud, mint amit eddig láttunk:

- van built-in term kiértékelés a standard struktúrában: 'X is Y + 1'
`sum_list([],0).`
`sum_list([H|T], S) :- sum_list(T,X), S is X + H.`
- egyenlőség tesztelés: `=\=` a két term értéke nem egyenlő; `\==` a két term nem egyenlő; `\=` a két term nem egyesíthető
- `==` a két term értéke egyenlő; `==` a két term ugyanaz; `=` a két term egyesíthető
- (read some trash at this site)

Amiben a Prolog többet tud, mint amit eddig láttunk:

- van pár built-in mellékható predikátum, pl. 'write/1', ha ilyen literált rezolválsz, kiírja (azonnal) a konzolra a predikátumban lévő termet

```
print_list([]).  
print_list([A]) :- write(A).  
print_list([H|T]) :- write(H), write(","), print_list(T).
```

- 'fail/0' predikátum: nem kiüthető, visszalépést kényszerít ki (mellékható törzsnél lehet érdekes)

```
e(1,2). e(2,3). e(3,1). e(2,4). e(4,5). e(5,4).  
print_neighbors(X) :- e(X,Y), print(Y), fail.  
print_neighbors(X). %különben false lenne
```

Amiben a Prolog többet tud, mint amit eddig láttunk:

- A Horn-klózek nem mindenre elegek – $\neg$ egyfajta negálás
- „not provable at this point”
- akkor megy tovább, ha a predikátumra a motor *false*t ad

```
in_list(X, [X|L]).
```

```
in_list(X, [_|L]) :- in_list(X, L).
```

```
not_in_list(X, L) :- \+ in_list(X, L).
```

- syntax sugar: ha a törzsben egy változó nem szerepel, `_` wildcard:

```
in_list(X, [X|_]).
```

```
in_list(X, [_|L]) :- in_list(X, L).
```

Amiben a Prolog többet tud, mint amit eddig láttunk:

- Vágás: ! a kiértékelés gyorsítására
- „ha idáig eljutsz, ezt a literált ne backtrackeld többször”

```
merge([],L,L).
merge(L,[],L).
merge([H1|T1],[H2|T2],[H2|L]) :- H1 >= H2, !, merge([H1|T1],T2 L).
merge([H1|T1],[H2|T2],[H1|L]) :- merge(T1,[H2|T2],L).

sort([],[]).
sort([A],[A]).
sort(L,S) :- split(L,L1,L2), sort(L1,S1), sort(L2,S2), merge(S1,S2,S).
```

keresés gráfban példa, ez már nem esik végtelen ciklusba

```
% input gráf
e(1,2). e(2,3). e(3,1). e(2,4). e(4,5). e(5,4).

in_list(X,[X|_]).
in_list(X,[_|L]) :- in_list(X,L).

path(X,X,[X]) :- !.
path(X,Y,[X|P]) :- search(X,Y,[X],P).

search(X,X,_,[]) :- !.
search(X,Y,Z,[W|P]) :- e(X,W), \+ in_list(W,Z), search(W,Y,[W|Z],P).
```

Másodrendű logika

Az elsőrendű logika bővítése **relációváltozókkal** (predikátumváltozókkal).

Formulák

Az elsőrendű logika formulaképzési szabályai plusz:

- Ha R n rangú predikátumváltozó és $t_1, \dots, t_n$ termek, akkor $R(t_1, \dots, t_n)$ is atomi formula.
- Ha R n rangú predikátumváltozó és F formula, akkor $\exists R F$ és $\forall R F$ is formulák.

Struktúra

$\mathcal{A} = (A, I, \varphi)$, ahol A, I mint az elsőrendű esetben, a φ értékelés pedig minden x elsőrendű változóhoz az A egy elemét, minden R n rangú predikátumváltozóhoz pedig egy $A^n \rightarrow \{0, 1\}$ predikátumot rendel.

Legyen $\mathcal{A} = (A, I, \varphi)$ struktúra, F formula. Az $\mathcal{A} \models F$ relációt az elsőrendű esethez hasonlóan definiáljuk az alábbiak figyelembe vételével:

- $\mathcal{A} \models R(t_1, \dots, t_n)$ akkor és csakis akkor, ha $\varphi(R)(\mathcal{A}(t_1), \dots, \mathcal{A}(t_n)) = 1$.
- $\mathcal{A} \models \exists R F$ akkor és csak akkor, ha létezik olyan φ' , mely legfeljebb az R -en tér el φ -től, amelyre $(A, I, \varphi') \models F$.
- $\mathcal{A} \models \forall R F$ akkor és csak akkor, ha bármely olyan φ' esetén, mely legfeljebb az R -en tér el φ -től, $(A, I, \varphi') \models F$.

Az, hogy $\mathcal{A} \models F$ fennáll-e, ismét független azon változók értékétől, melyek nem fordulnak elő szabadon F -ben.

- „Az univerzum végtelen”
- Math says egy halmaz akkor végtelen, ha van önmagába képző injektív, de nem szürjektív leképezése.
- A **leképezés** egy bináris reláció: $\exists R \dots$
- aminek minden bal oldalhoz pontosan egy jobb oldal párosul:

$$\forall x \exists y R(x, y) \wedge \forall x \forall y_1 \forall y_2 (R(x, y_1) \wedge R(x, y_2) \rightarrow y_1 = y_2)$$

- injektív:

$$\forall x_1 \forall x_2 \forall y (R(x_1, y) \wedge R(x_2, y) \rightarrow x_1 = x_2)$$

- nem szürjektív: van, aki nem áll elő képként

$$\exists y \forall x \neg R(x, y)$$

Példa

A természetes számok szokásos struktúrája kielégíti a

$$\forall X((X(\underline{0}) \wedge \forall x(X(x) \rightarrow X(x')))) \rightarrow \forall x X(x))$$

indukciós axiómát.

A Peano-axiómák, plusz a fenti indukciós axióma (az indukciós axióma séma példányai nélkül is) modellje (izomorfizmus erejéig) kizárólag $\mathcal{N}$.

Példa

Egy $G = (V, E)$ gráf csúcsai pontosan akkor színezhető három színnel helyesen (=szomszédos csúcsok különböző színt kapjanak), ha kielégíti a

$$\begin{aligned} \exists X \exists Y \exists Z \big(& \forall x (X(x) \vee Y(x) \vee Z(x)) \wedge \\ & \forall x (\neg(X(x) \wedge Y(x)) \wedge \neg(X(x) \wedge Z(x)) \wedge \neg(Y(x) \wedge Z(x))) \wedge \\ & \forall x \forall y (e(x, y) \rightarrow \neg(X(x) \wedge X(y) \vee Y(x) \wedge Y(y) \vee Z(x) \wedge Z(y))) \\ & \big) \end{aligned}$$

mondatot.

Példa

Egy $G = (V, E)$ gráfban pont akkor van Hamilton-út (= minden csúcson pontosan egyszer áthaladó út), ha kielégíti a

$$\begin{aligned} \exists R \big(& \forall x \neg R(x, x) \wedge \\ & \forall x \forall y (x = y \vee R(x, y) \vee R(y, x)) \wedge \\ & \forall x \forall y \forall z (R(x, y) \wedge R(y, z) \rightarrow R(x, z)) \wedge \\ & \forall x \forall y (R(x, y) \wedge \neg \exists z (R(x, z) \wedge R(z, y)) \rightarrow e(x, y)) \big) \end{aligned}$$

mondatot.

Másodrendű logika

- Másodrendű logikában sokkal összetettebb állításokat lehet megfogalmazni, mint elsőrendűben.
- Viszont nincs helyes és teljes következtető rendszer.
- Toolok: Coq, Isabelle/HOL,...
- Hales 1998:
 - „Ágyúgolyó elrendezés” a legsűrűbb
 - 300 oldal matek
 - 40.000 sor programkód (gráfok generálására stb)
 - egy 150-változós függvény minimalizálása, 5.000 gömb-konfigurációra alsó korlát 100.000 LP feladattal
 - A bírálók feladták
 - Hales nem: Flyspeck projekt – teljes formalizálás (2014-re)
- Leroy 2008–2017:
 - CompCert – bizonyítottan helyes C fordító (Coq)
 - Majdnem eléri a gcc -O3 sebességét a benchmarkokon

Floyd-Hoare kalkulus

Code contractok – C + ACSL (ANSI C Spec Lang)

```
//@ requires A>=0 && B>0;
//@ ensures \result == A mod B;
int mod(int A, int B) {
    int Q = 0;
    int R = A;
    //@ assert A>=0 && B>0 && Q==0 && R==A;
    while (R >= B) {
        //@ assert A>=0 && B>0 && R>=B && A==Q*B+R;
        R = R - B;
        Q = Q + 1;
    }
    //@ assert A>=0 && B>0 && R>=0 && R<B && A==Q*B+R;
    return R;
}
```

Code contractok – Java + ESC (Extended Static Checker)

```
public class OrderedArray {  
    int a[];  
    int nb;  
    //@invariant nb >= 0 && nb <= 20  
    //@invariant (\forall int i; (i >= 0 && i < nb-1) ==> a[i] <= a[i+1])  
    public OrderedArray() { a = new int[20]; nb = 0; }  
    public void add(int v) {  
        if (nb >= 20) return;  
        int i;  
        for (i=nb; i > 0 && a[i-1] > v; i--) a[i] = a[i-1];  
        a[i] = v; nb++;  
    }  
}
```

Code contractok – C + ACSL (ANSI C Spec Lang)

```
/*@  
  requires \valid_read(a + (0..n-1));  
  assigns \nothing;  
  
  behavior some:  
    assumes \exists integer i; 0 <= i < n && a[i]==val;  
    ensures 0 <= \result < n;  
    ensures a[\result] == val;  
    ensures \forall integer i; 0 <= i < \result ==> a[i]!=val;  
  
  behavior none:  
    assumes \forall integer i; 0 <= i < n ==> a[i]!=val;  
    ensures \result == n;  
  
  complete behaviors;  
  disjoint behaviors;  
*/  
size_type find(const value_type* a, size_type n, value_type val);
```



```

size_type find(const value_type *a, size_type n, value_type val){
    /*@
    loop invariant 0 <= i <= n;
    loop invariant \forall integer k; 0 <= k < i ==> a[k] != val;
    loop assigns i;
    loop variant n-i;
    */
    for(size_type i = 0; i < n; i++) {
        if(a[i] == val) { return i; }
    }
    return n;
}

```

Code contractok – Dafny (MS, C#)

```
function Fib( n: nat ) : nat {  
  if ( n < 2 ) then n else Fib( n-1 ) + Fib( n-2 )  
}  
  
method computeFib( n: nat ) returns ( x: nat )  
ensures x == Fib( n ); {  
  var i:=0;  
  x := 0;  
  var y:=1;  
  while( i < n )  
  invariant 0 <= i <= n;  
  invariant x == Fib( i );  
  invariant y == Fib( i + 1 ); {  
    x,y := y,x+y;  
    i := i+1;  
  }  
}
```

```

/*@
  ensures A: *a == \old(*b);
  ensures B: *b == \old(*a);
*/
void swap(int *a, int * b){
  int tmp = *a; *a = *b; *b = tmp;
}

```

```

> frama-c -wp swap.c swap1.h
# frama-c -wp [...]
[kernel] Parsing FRAMAC_SHARE/libc/__fc_builtin_for_normalization.i
[kernel] Parsing swap.c (with preprocessing)
[kernel] Parsing swap1.h (with preprocessing)
[wp] Running WP plugin...
[wp] Loading driver 'share/wp.driver'
[wp] Collecting axiomatic usage
[wp] warning: Missing RTE guards
[wp] 2 goals scheduled
[wp] [Alt-Ergo] Goal typed_swap_post_A : Valid
[wp] [Qed] Goal typed_swap_post_B : Valid
[wp] Proved goals: 2 / 2
      Qed:          1
      Alt-Ergo:     1

```

Functions	WP	Alt-Ergo	Total	Success
swap	1	1 (12)	2	100%

```
/*@  
requires \valid(a) && valid(b);  
ensures A: *a == \old(*b);  
ensures B: *b == \old(*a);  
assigns *a, *b;  
*/  
void swap(int *a, int * b){  
int tmp = *a; *a = *b; *b = tmp;  
}
```

```

> frama-c -wp -wp-rte swap.c swap2.h
# frama-c -wp -wp-rte [...]
[kernel] Parsing FRAMAC_SHARE/libc/_fc_builtin_for_normalization.i
[kernel] Parsing swap.c (with preprocessing)
[kernel] Parsing swap2.h (with preprocessing)
[wp] Running WP plugin...
[wp] Loading driver 'share/wp.driver'
[wp] Collecting axiomatic usage
[rte] annotating function swap
[wp] 9 goals scheduled
[wp] [Alt-Ergo] Goal typed_swap_post_A : Valid
[wp] [Qed] Goal typed_swap_post_B : Valid
[wp] [Alt-Ergo] Goal typed_swap_assert_rte_mem_access : Valid
[wp] [Qed] Goal typed_swap_assert_rte_mem_access_2 : Valid
[wp] [Alt-Ergo] Goal typed_swap_assert_rte_mem_access_3 : Valid
[wp] [Qed] Goal typed_swap_assert_rte_mem_access_4 : Valid
[wp] [Qed] Goal typed_swap_assign_part1 : Valid
[wp] [Qed] Goal typed_swap_assign_part2 : Valid
[wp] [Qed] Goal typed_swap_assign_part3 : Valid
[wp] Proved goals: 9 / 9
      Qed:          6
      Alt-Ergo:     3

```

Functions	WP	Alt-Ergo	Total	Success
swap	6	3 (17)	9	100%

Floyd-Hoare kalkulus: a while programok szintaktikája

Ami a fenti verifikáló programok magját alkotja, a **Floyd-Hoare kalkulus**. Egy egyszerű programozási nyelvre, a **while** kódokra definiáljuk, de ez van kiterjesztve C-re, Java-ra stb.

Legyen adott egy elsőrendű nyelv (azaz a függvény- és relációszimbólumok egy-egy megszámlálható halmaza).

A **while programok** az alábbiak:

- $x := t$, ahol x változó, t term,
- $P_1; P_2$, ahol P_1, P_2 programok,
- **if** r **then** P_1 **else** P_2 , ahol r kvantormentes formula, P_1, P_2 programok,
- **while** r **do** P , ahol r kvantormentes formula, P program.

Floyd-Hoare kalkulus

- Legyen P program, $\mathcal{A} = (A, I)$ egy struktúra.
- Ekkor P meghatároz az **értékadások** fölött egy $[[P]]$ relációt: $\varphi[[P]]\psi$ pontosan akkor, ha
 - a változók értékeit φ szerint beállítva,
 - majd futtatva P -t,
 - P futása befejeződik és ψ adja a változók végértékét.

Bármely φ -hez legfeljebb egy olyan ψ létezik, melyre $\varphi[[P]]\psi$.

Például:

- $\varphi(x) = 1, \varphi(y) = 2, \dots$
- $P = x := y + 1; y := x$
- $\phi(x) = 3, \phi(y) = 3, \dots$

akkor $\varphi[[P]]\phi$.

Floyd-Hoare kalkulus: szemantika formális definíciója

Legyen P program, $\mathcal{A}$ struktúra. Ha φ értékadás, akkor legyen $\mathcal{A}_\varphi$ az (A, I, φ) struktúra.

$\varphi[[P]]\psi$ pontosan a következő esetekben teljesül:

- $P = x := t$ és $\psi = \varphi[x \mapsto \mathcal{A}_\varphi(t)]$.
(Azaz: ψ annyiban tér el φ -től, hogy benne x új értéke a t értéke (A, I, φ) -ben.)
- $P = P_1; P_2$ és van olyan τ értékadás, melyre $\varphi[[P_1]]\tau$ és $\tau[[P_2]]\psi$.
(Azaz: ha φ -n indítva P_1 lefut, a változók értéke ekkor τ , majd ekkor indítva P_2 -t az is lefut és a változók értéke ψ lesz.)
- $P = \text{if } r \text{ then } P_1 \text{ else } P_2$ és

$$\begin{array}{ll} \varphi[[P_1]]\psi & \text{és} \quad \mathcal{A}_\varphi(r) = 1 \quad , \text{ vagy} \\ \varphi[[P_2]]\psi & \text{és} \quad \mathcal{A}_\varphi(r) = 0. \end{array}$$

(azaz: vagy r igaz és P_1 készít φ -ből ψ -t, vagy r hamis és P_2 készít φ -ből ψ -t.)

- $P = \text{while } r \text{ do } P_1$ és vannak olyan $\tau_0, \tau_1, \dots, \tau_n$ értékadások, $n \geq 0$ (note: n -szer fut le a ciklus), hogy
 - $\tau_0 = \varphi$: kezdetben a változók értéke φ
 - $\tau_n = \psi$: kilépéskor a változók értéke ψ
 - minden $i = 0, \dots, n-1$ -re $\mathcal{A}_{\tau_i}(r) = 1$: ezért hajtódik végre a ciklusmag n -szer
 - $\mathcal{A}_{\tau_n}(r) = 0$: ezért lép ki a ciklus az n . iteráció után
 - minden $i = 0, \dots, n-1$ -re $\tau_i[[P_1]]\tau_{i+1}$: a ciklusmag egyszeri lefutása τ_i -ből τ_{i+1} -et készít

Parciális helyességi kifejezések az

$$\{F\}P\{G\}$$

alakú hármasok, ahol P program, F és G elsőrendű formulák.

Azt mondjuk, hogy az $\{F\}P\{G\}$ parciális helyességi kifejezés **teljesül** (vagy **érvényes**) az $\mathcal{A} = (A, I)$ struktúrában, vagy $\mathcal{A}$ **kielégíti** az $\{F\}P\{G\}$ parciális helyességi kifejezést, jelben $\mathcal{A} \models \{F\}P\{G\}$, ha valahányszor φ, ψ olyan értékelések, hogy

$$(A, I, \varphi) \models F \text{ és } \varphi[[P]]\psi,$$

fennáll, hogy

$$(A, I, \psi) \models G.$$

Példa

$$P = y := 1; \text{ while } x > 0 \text{ do } (y := y \times x; x := x - 1)$$

Ekkor a standard struktúrában:

$$\begin{aligned} \varphi[[P]]\psi \quad \Leftrightarrow \quad & ((\varphi(x) < 0, \psi(x) = \varphi(x), \psi(y) = 1) \\ & \vee (\varphi(x) \geq 0, \psi(x) = 0, \psi(y) = x!)) \\ & \wedge (\varphi(z) = \psi(z), z \notin \{x, y\}) \end{aligned}$$

Az előző P programra és az egész számok standard $\mathcal{A}$ struktúrájára:

$$\mathcal{A} \models \{x = z \wedge x \geq 0\} P \{y = z! \wedge x = 0\}$$

$$\mathcal{A} \models \{x = z\} P \{y = z! \vee y = 1\}$$

Példa

Az egész számok szokásos struktúrájában érvényes:

$$\{a \geq 0\}$$

$$x := 0;$$

$$y := 1;$$

$$\text{while } y \leq a \text{ do}$$

$$x := x + 1; y := y + 2x + 1$$

$$\{0 \leq x^2 \leq a < (x + 1)^2\}$$

Példa

$$P' = \text{while } x \neq 100 \text{ do } x := x + 2$$

Ekkor a standard struktúrában érvényesek:

$$\{x = z\}P'\{x = 100\}, \quad \{\uparrow\}P'\{x = 100\}$$

Totális helyességi kifejezésnek nevezünk egy

$$[F]P[G]$$

hármast, ahol P program, F , G formulák.

Azt mondjuk, hogy az $\mathcal{A} = (A, I)$ struktúra **kielégíti** az $[F]P[G]$ totális helyességi kifejezést, ha tetszőleges olyan φ értékelésre, melyre $\mathcal{A}_\varphi \models F$, létezik olyan (egyértelműen meghatározott) ψ értékelés, hogy $\varphi[[P]]\psi$ és $\mathcal{A}_\psi \models G$. Jelölés: $\mathcal{A} \models [F]P[G]$.

Tehát a totális helyességi kifejezés megköveteli azt is, hogy P megálljon, ha olyan inputon futtatjuk, melyre F igaz.

Példa

Az előző P , P' programokra és az $\mathcal{A}$ standard struktúrára:

$$\mathcal{A} \models [x = z \wedge x \geq 0]P[y = z!]$$

$$\mathcal{A} \not\models [\uparrow]P'[x = 100]$$

$$\mathcal{A} \models [x \leq 100 \wedge (\exists u \ x = 2u)]P'[x = 100]$$

Megjegyzés

$\mathcal{A} \models [F]P[\uparrow]$ akkor és csakis akkor teljesül, ha P **megáll** minden olyan φ esetén, amelyre $\mathcal{A}_\varphi \models F$. Jelölés: $[F]P \searrow$.

Tehát $\mathcal{A} \models [F]P[G]$ akkor és csak akkor, ha $\mathcal{A} \models \{F\}P\{G\}$ és $\mathcal{A} \models [F]P \searrow$.

A Hoare-féle szabályok

- Értékdás

$$\overline{\{F[x/t]\}x := t\{F\}}$$

- Kompozíció

$$\frac{\{F\}P_1\{H\} \quad \{H\}P_2\{G\}}{\{F\}P_1; P_2\{G\}}$$

- Feltételes utasítás

$$\frac{\{F \wedge r\}P_1\{G\} \quad \{F \wedge \neg r\}P_2\{G\}}{\{F\}\text{if } r \text{ then } P_1 \text{ else } P_2\{G\}}$$

A Hoare-féle szabályok, folytatás

- Ciklus

$$\frac{\{F \wedge r\}P\{F\}}{\{F\}\text{while } r \text{ do } P\{F \wedge \neg r\}}$$

- **Monotonitás** Tegyük fel, hogy $\forall(F \rightarrow F')$ és $\forall(G' \rightarrow G)$ az $\mathcal{A}$ elsőrendű elméletében vannak. Akkor:

$$\frac{\{F'\}P\{G'\}}{\{F\}P\{G\}}$$

Legyen $\mathcal{A}$ elsőrendű struktúra. Azt mondjuk, hogy az $\{F\}P\{G\}$ parciális helyességi kifejezés **levezethető** (vagy **bizonyítható**) $\text{Th}(\mathcal{A})$ -ból,

$$\text{Th}(\mathcal{A}) \vdash \{F\}P\{G\},$$

ha létezik a parciális helyességi kifejezések olyan

$$E_0, E_1, \dots, E_n$$

sorozata, hogy $E_n = \{F\}P\{G\}$ és minden $i > 0$ -ra E_i a fenti szabályok valamelyikével áll elő az $E_0, E_1, \dots, E_{i-1}$ kifejezésekből és a $\text{Th}(\mathcal{A})$ formulahalmazból.

Példa

Az egész számok szokásos struktúrájában érvényes:

$$\begin{aligned} &\{a \geq 0\} \\ &x := 0; \\ &y := 1; \\ &\text{while } y \leq a \text{ do} \\ &\quad x := x + 1; \ y := y + 2x + 1 \\ &\{0 \leq x^2 \leq a < (x + 1)^2\} \end{aligned}$$

Ehhez pár lépés (értékadás, kompozíció):

$$(x + 1)^2 \leq a \wedge y + 2(x + 1) + 1 = (x + 1 + 1)^2$$

$$x := x + 1$$

$$x^2 \leq a \wedge y + 2x + 1 = (x + 1)^2$$

$$y := y + 2x + 1$$

$$x^2 \leq a \wedge y = (x + 1)^2$$

Mivel az egész számokra igaz ez (Peano axiómákból is kijön!):

$$\begin{aligned}x^2 \leq a \wedge y = (x+1)^2 \wedge y \leq a \\ \Downarrow \\ (x+1)^2 \leq a \wedge y + 2(x+1) + 1 = (x+1+1)^2\end{aligned}$$

Ezért a monotonitás szabály szerint:

$$\begin{aligned}x^2 \leq a \wedge y = (x+1)^2 \wedge y \leq a \\ \textcolor{red}{x} := \textcolor{red}{x} + 1 \\ \textcolor{red}{y} := \textcolor{red}{y} + 2\textcolor{red}{x} + 1 \\ x^2 \leq a \wedge y = (x+1)^2\end{aligned}$$

Most a ciklus szabályt alkalmazzuk:

$$\begin{aligned} & x^2 \leq a \wedge y = (x+1)^2 \\ & \text{while } y \leq a \text{ do} \\ & \quad x := x + 1 \\ & \quad y := y + 2x + 1 \\ & x^2 \leq a \wedge y = (x+1)^2 \wedge \neg(y \leq a) \end{aligned}$$

Monotonitás:

$$\begin{aligned} & 0 \leq a \wedge y = 1 \wedge x = 0 \\ & \text{while } y \leq a \text{ do} \\ & \quad x := x + 1 \\ & \quad y := y + 2x + 1 \\ & x^2 \leq a \wedge y = (x+1)^2 \wedge a < (x+1)^2 \end{aligned}$$

Levezetés példa

$$0 \leq a \wedge 1 = 1 \wedge 0 = 0$$

$$x := 0$$

$$0 \leq a \wedge 1 = 1 \wedge x = 0$$

$$y := 1$$

$$0 \leq a \wedge y = 1 \wedge x = 0$$

$$\text{while } y \leq a \text{ do}$$

$$x := x + 1$$

$$y := y + 2x + 1$$

$$x^2 \leq a \wedge y = (x + 1)^2 \wedge a < (x + 1)^2$$

Monotonitás:

$$0 \leq a$$

$$x := 0; y := 1$$

$$\text{while } y \leq a \text{ do}$$

$$x := x + 1$$

$$y := y + 2x + 1$$

$$x^2 \leq a \wedge y = (x + 1)^2 \wedge a < (x + 1)^2$$

Tétel

Ha $\text{Th}(\mathcal{A}) \vdash \{F\}P\{G\}$, akkor $\mathcal{A} \models \{F\}P\{G\}$.

A tétel megfordítása általában nem igaz, de érvényes az ún. **expresszív** struktúrákra, azaz azon $\mathcal{A} = (A, I)$ struktúrákra, amelyekre igaz a következő: Tetszőleges P programhoz és G formulához létezik olyan F formula, hogy bármely φ értékelésre $\mathcal{A}_\varphi \models F$ akkor és csak akkor, ha $[[P]]$ nem értelmezett φ -n, vagy ha értelmezett, akkor arra a ψ -re, melyre $\varphi[[P]]\psi$, teljesül, hogy $\mathcal{A}_\psi \models G$.

Pl. az egész számok (vagy a természetes számok) standard struktúrája expresszív.

Tétel (Cook)

Ha $\mathcal{A}$ expresszív, akkor $\mathcal{A} \models \{F\}P\{G\}$ esetén $\text{Th}(\mathcal{A}) \vdash \{F\}P\{G\}$.

A totális helyesség szabályai

A ciklus szabály kivételével hasonlóak a parciális helyesség szabályaihoz.

Az új ciklus szabály: tegyük fel, hogy az $\mathcal{A} = (A, I)$ struktúrában $I(<)$ egy **jól megalapozott** részbenrendezés. Ekkor:

$$\frac{[F \wedge r \wedge t = z_0] P[F \wedge t < z_0]}{[F] \text{while } r \text{ do } P[F \wedge \neg r]},$$

ahol z_0 máshol nem fordul elő.