A logikai absztrakt adattípus

A logikai absztrakt adattípus egy olyan halmazt ad meg, amelynek két eleme van, a *hamis* és az *igaz*. Jelölésben $L=\{hamis, igaz\}$. Röviden az elemeket a *h* (hamis) és az *i* (igaz) jellel jelöljük. A logikai absztrakt adattípus megadása: T=(L,M), ahol L a logikai adatok halmaza (kételemű) és M a rajtuk végezhető műveletek halmaza.
A típus M-ben felsorolt műveletei lehetnek unáris (unér) és bináris (binér) aszerint, hogy hány operandusuk van. (Lehetnek többoperandusú műveletek is – ternáris, kvaternáris, stb.… - , de ezek, mint látni fogjuk, az előzőkkel kifejezhetők.)

Mivel az adattípusnak mindössze két eleme van, ezért az összes műveletét fel tudjuk sorolni. Mivel mindegyik operandus kétféle értéket vehet fel, *n* operandus esetén ez 2^n lehetséges érték *n*-est jelent. Mindegyik esetében szintén kétféle eredmény kapcsolható hozzájuk mind a 2^n helyen, ezért általában *n*-változós műveletből 2^{2^n} számú van.

Unáris művelet négy (2^{2^1}) van, amint az az alábbi táblázatból kiolvasható. A Hamis és az Igaz mindig konstanst eredményez, bármi legyen is az *x* értéke, az Identikus visszaadja magát az *x*-et, tehát az egyetlen érdekes a Negáció, (tagadás, NEM, NOT), aminek a jele a felülvonás a logikai adat neve fölött. Pl.:az *x* adat negáltja (tagadottja, NEM *x*, NOT *x*) $\overline{x}$.

	Hamis	Identikus	Negáció,tagadás NEM, NOT	Igaz
x	*h*	*x*	$\overline{x}$	*i*
h	*h*	*h*	*i*	*i*
i	*h*	*i*	*h*	*i*

Érdekesebb bináris műveletek (a műveleti jeleket a két operandus közé helyeztük):

x	*y*	Diszjunkció (VAGY, OR) $x \vee y$	Konjunkció (ÉS, AND) $x \wedge y$	Antivalencia (KIZÁRÓ VAGY, XOR) $x \oplus y$	Ekvivalencia $x \leftrightarrow y$	Implikáció $x \rightarrow y$	Peirce nyíl (NEM VAGY, NOR) $x \downarrow y$	Scheffer vonás (NEM ÉS, NAND) $x \mid y$
h	*h*	*h*	*h*	*h*	*i*	*i*	*i*	*i*
h	*i*	*i*	*h*	*i*	*h*	*i*	*h*	*i*
i	*h*	*i*	*h*	*i*	*h*	*h*	*h*	*i*
i	*i*	*i*	*i*	*h*	*i*	*i*	*h*	*h*

Bináris műveletből 16-ot (2^{2^2}) lehet felírni. (Találjuk meg a többit!). A műveletek erősorrendje csökkenő erő szerint (prioritás, zárójelezés nélkül írhatók) a következő: NEM, ÉS, VAGY, KIZÁRÓ VAGY, Ekvivalencia, Implikáció.

A NEM, ÉS, VAGY műveletek tulajdonságai:

1.	Kettős tagadás	$\overline{\overline{x}} = x$	
2.	Kommutativitás	$x \vee y = y \vee x$	$x \wedge y = y \wedge x$
3.	Asszociativitás	$(x \vee y) \vee z = x \vee (y \vee z)$	$(x \wedge y) \wedge z = x \wedge (y \wedge z)$
4.	Disztributivitás	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
5.	Idempotencia	$x \vee x = x$	$x \wedge x = x$
6.	Konstansok hatása	$x \vee i = i$ $x \vee h = x$	$x \wedge i = x$ $x \wedge h = h$
7.	Elnyelés	$x \vee (x \wedge y) = x$	$x \wedge (x \vee y) = x$
8.	Ellentmondás		$x \wedge \overline{x} = h$
9.	Harmadik kizárása	$x \vee \overline{x} = i$	
10.	De Morgan szabály	$\overline{x \vee y} = \overline{x} \wedge \overline{y}$	$\overline{x \wedge y} = \overline{x} \vee \overline{y}$

Ezen három művelettel (NEM, ÉS, VAGY) az összes többi (a többváltozósak is) kifejezhetők.

Példák:

$x \oplus y \equiv (\overline{x} \wedge y) \vee (x \wedge \overline{y})$
$x \rightarrow y \equiv \overline{x} \vee y$
$x \leftrightarrow y \equiv (x \wedge y) \vee (\overline{x} \wedge \overline{y})$

A diszjunktív normálforma

Definíció: Elemi konjunkció
Változók vagy tagadottjainak a konjunkciója, melyben a változók legfeljebb egyszer fordulnak elő.

Definíció: Diszjunktív normálforma (DNF)
Elemi konjunkciók diszjunkciója

Művelettábla alapján DNF előállítása: Ahol az eredmény oszlopban *i* van, azokat az eseteket diszjunkcióval kötjük össze úgy, hogy a változók konjunkcióiból formulát alkotunk. A formulában *i* esetén a változó szerepel, *h* esetén a változó negáltja.

Példa:

x	y	$x \oplus y$
h	h	h
h	i	i
i	h	i
i	i	h

Innen $x \oplus y \equiv (\overline{x} \wedge y) \vee (x \wedge \overline{y})$.

A logikai adatstruktúra

Már eddig is végső soron adatstruktúrával dolgoztunk, hiszen lejegyeztük a hamis és az igaz értéket *h*-val és *i*-vel. Látjuk, hogy igazán semmilyen probléma itt nem adódik, minden művelet ragyogóan működik. A *h* és *i* betűk helyett használhattuk volna a *hamis* és *igaz* szavakat is, az is egy lehetséges adatstruktúra lenne. Ha a szavak angol megfelelőit használnánk (*false* és *true*), akkor írhatnánk az *f* és *t* betűket a h és *i* helyett. Az sem tiltott, hogy a 0 és 1 jeleket alkalmazzuk, mint alább láthatjuk. Tehát az adatstruktúra hűen realizálja az absztrakt adattípust.

A logikai adattípus/adatstruktúra implementációja

A logikai adat (változó) realizálása bitekkel értelemszerűen történhet a hamis = 0, igaz = 1 módon.

Definíció:	**Izomorfizmus**
	Két algebrai struktúrát ***izomorf***nak nevezünk, ha létezik olyan kölcsönösen egyértelmű megfeleltetés a két struktúra elemei között, amely esetén a műveletek is szinkronizálódnak. Ez azt jelenti, hogy ha az egyik struktúra az $(A, \square)$, a másik struktúra a $(B,\circ)$, a kölcsönösen egyértelmű megfeleltetés pedig $f : A \rightarrow B$, akkor fennáll, hogy $f(a_1 \square a_2)=f(a_1)\circ f(a_2)$ minden $a_1 \in A$ és $a_2 \in A$ esetén. Az f megfeleltetést nevezzük ***izomorfizmus***nak.

Az izomorfizmus azt jelenti, hogy a két struktúra azonos szerkezetű a művelet viselkedését tekintve.

Példa : Legyen $A=R^+$, a pozitív valós számok halmaza a szorzás művelettel felruházva, és $B=R$, az összes valós szám halmaza az összeadás művelettel. Akkor az $f : R^+ \rightarrow R$ megfeleltetés, ahol $f(x)=\log(x)$, izomorfizmust valósit meg, hiszen a logaritmus azonosságai szerint: $\log(xy)= \log(x) +\log(y)$ minden pozitív x és y esetén.

Legyen most az A halmaz az L logikai adattípus a negáció, a konjunkció és a diszjunkció műveletével felruházva. $T_L=(L, \{ \bar{x},\wedge,\vee \})$. Legyen a B halmaz a {0,1} számokból álló halmaz, amelyen értelmezzük az alábbi három múveletet, miáltal létrehozunk egy $T_B=(B,\{e(b),\ \ ,\square\})$ adattípust.

1. Ellentett elem képzése unáris művelet : $e(b)=1-b$, $b \in B$ a kivonás művelete révén.
2. Szorzás bináris művelet: $b_1 \cdot b_2$, $b_1, b_2 \in B$ a számok szorzási műveletének megfelelően.
3. Bitösszegzés bináris művelet : $b_1 \square b_2= b_1 - b_1 \cdot b_2+b_2 \quad b_1, b_2 \in B$ a számokra érvényes szokásos összeadás, kivonás és szorzás művelete révén.

Ekkor a most definiált három művelet a negáció, konjunkció, és diszjunkció műveletének megfeleltetve, valamint a logikai mennyiségeknek a biteket a fenti módon megfeleltetve a logikai adattípus és a bit adattípus között izomorfizmust hoztunk létre. Azt mondjuk, hogy a logikai adatokat bitekkel modellezzük. Amilyen törvényszerűséget találunk az egyikben, az izomorfizmus révén megtaláljuk a törvény párját a másikban. (Mutassuk meg, hogy a $\square$ művelet ekvivalens a $\max(b_1, b_2)$ művelettel.)

A logikai típus nagyon fontos, mert az értékeket feszültségszintekhez lehet társítani, a műveleteket pedig úgynevezett kapuáramkörökkel valósíthatjuk meg. A kapuáramkör fizikai (technikai) felépítése lényegtelen a számunkra, az változott az idők folyamán (relék, diódák,

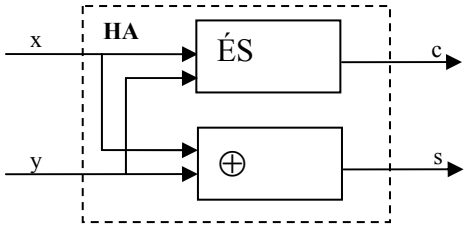
tranzisztorok, stb.). Sematikusan úgy jelölhetjük őket, mint egy dobozba zárt átalakító szerkezet, amelynek vannak bemenetei és kimenetei. A bemeneteken bemenő jeleket dolgozzák fel a rendeltetésüknek megfelelően, és az eredmény megjelenik a kimeneteken.
Példa kapuáramkörökre:

Mivel mindegyik művelet kifejezhető ezen háromm művelettel, ezzel elérjük, hogy kevés alaptípusból bonyolult rendeltetésű kapuáramköröket építhetünk fel.
Kapuáramkörökből felépíthető az úgynevezett félösszeadó (Half Adder). Feladata egyetlen bitpozíción képezni a két bit összegét és az átvitelt, tehát a művelet mindig kétjegyű eredményt képez, melynek alacsonyabb helyiértékű bitje az összegbit (*s*), magasabb helyiértékű bitje az átvitelbit (Carry bit, *c*).

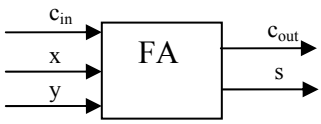
Láthatóan
$c = x \wedge y$
$s = x \oplus y$

Művelettábla

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



A félösszeadó több-bites számok összeadásakor csak fél munkát végez, helyesen csak a legalacsonyabb bitpozíción működik. A további pozíciókon három bitet kell összeadni, a két összeadandó bitet és az előző pozícióról jövő átvitelbitet. A teljes összeadó (Full Adder) ezt valósítja meg.



Művelettáblája:

x	y	c_{in}	c_{out}	s
0	0	0	0	0
0	1	0	0	1
1	0	0	0	1
1	1	0	1	0
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	1	1	1

Felírva a két eredményoszlopra a diszjunktív normálformákat, tulajdonképpen megkapjuk a műveletek egy lehetséges kapuzását.

$$c_{out} = \left(x \wedge y \wedge \overline{c_{in}}\right) \vee \left(\bar{x} \wedge y \wedge c_{in}\right) \vee \left(x \wedge \bar{y} \wedge c_{in}\right) \vee \left(x \wedge y \wedge c_{in}\right)$$
$$s = \left(\bar{x} \wedge y \wedge \overline{c_{in}}\right) \vee \left(x \wedge \bar{y} \wedge \overline{c_{in}}\right) \vee \left(\bar{x} \wedge \bar{y} \wedge c_{in}\right) \vee \left(x \wedge y \wedge c_{in}\right)$$

Ezt a két, látszólag bonyolult formulát le lehet egyszerűsíteni, és a teljes összeadó felépíthető két félösszeadó és egy VAGY kapuáramkörből. Előtte azonban egy segéd formulát vezetünk le.

Állítás: $\left(\bar{x} \wedge \bar{y}\right) \vee \left(x \wedge y\right) = \overline{x \oplus y}$

Bizonyítás: Láttuk, hogy $x \oplus y = \left(\overline{x} \wedge y\right) \vee \left(x \wedge \overline{y}\right)$. Akkor

$$\left(\overline{x} \wedge \overline{y}\right) \vee \left(x \wedge y\right) = \overline{\overline{\left(\overline{x} \wedge \overline{y}\right) \vee \left(x \wedge y\right)}} = \overline{\overline{\left(\overline{x} \wedge \overline{y}\right)} \wedge \overline{\left(x \wedge y\right)}} = \overline{\left(x \vee y\right) \wedge \left(\overline{x} \vee \overline{y}\right)} =$$

$$= \underbrace{\left(x \wedge \overline{x}\right)}_{h} \vee \left(y \wedge \overline{x}\right) \vee \left(x \wedge \overline{y}\right) \vee \underbrace{\left(y \wedge \overline{y}\right)}_{h} = \left(y \wedge \overline{x}\right) \vee \left(x \wedge \overline{y}\right) = \overline{x \oplus y}$$

■

Ezután a teljes összeadó levezetése az alábbi lehet:
Jelölje s', c' és s'', c'' az első, valamint a második félösszeadó által adott eredmény összegbitet és az átvitelbitet.. Akkor

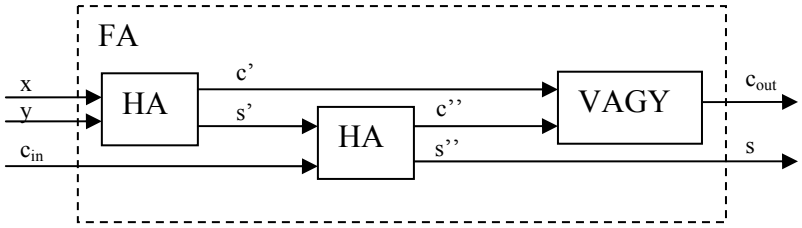
$$c_{out} = \left(x \wedge y \wedge \overline{c_{in}}\right) \vee \left(\overline{x} \wedge y \wedge c_{in}\right) \vee \left(x \wedge \overline{y} \wedge c_{in}\right) \vee \left(x \wedge y \wedge c_{in}\right) =$$

$$= \left(x \wedge y\right) \wedge \left(\underbrace{\overline{c_{in}} \wedge c_{in}}_{i}\right) \vee \left[\underbrace{\left(\overline{x} \wedge y\right) \vee \left(x \wedge \overline{y}\right)}_{x \oplus y}\right] \wedge c_{in} = \underbrace{\left(x \wedge y\right)}_{c'} \vee \underbrace{\left(\underbrace{x \oplus y}_{s'} \wedge c_{in}\right)}_{c''} = c' \vee c''$$

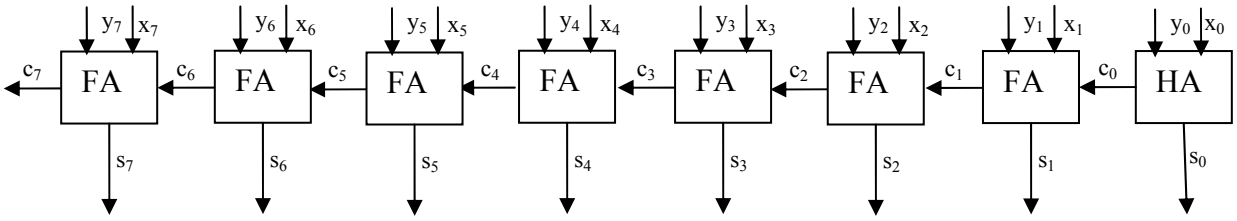
$$s = \left(\overline{x} \wedge y \wedge \overline{c_{in}}\right) \vee \left(x \wedge \overline{y} \wedge \overline{c_{in}}\right) \vee \left(\overline{x} \wedge \overline{y} \wedge c_{in}\right) \vee \left(x \wedge y \wedge c_{in}\right) =$$

$$= \left[\underbrace{\left(\overline{x} \wedge y\right) \vee \left(x \wedge \overline{y}\right)}_{x \oplus y}\right] \wedge \overline{c_{in}} \vee \left[\underbrace{\left(\overline{x} \wedge \overline{y}\right) \vee \left(x \wedge y\right)}_{\overline{x \oplus y}}\right] \wedge c_{in} = \underbrace{\left(x \oplus y\right)}_{s'} \oplus c_{in} = s' \oplus c_{in} = s''$$

A teljes összeadó sematikus ábrája:



Egy egybyte-os (nyolcbites) teljes összeadó ezután összerakható a megfelelő késleltető áramkörök közbeiktatásával az alábbi séma szerint.



Szintén kifejezhető az összes művelet csupán a (NEM, VAGY), vagy csupán a (NEM, ÉS), műveletekkel. Sőt, egyféle művelet is elegendő erre, ha a (NEM VAGY = Pierce nyíl) vagy pedig a (NEM ÉS = Scheffer vonás) műveletet választjuk. (Próbáljuk meg általuk kifejezni a három (NEM, ÉS, VAGY) alapműveletet!)

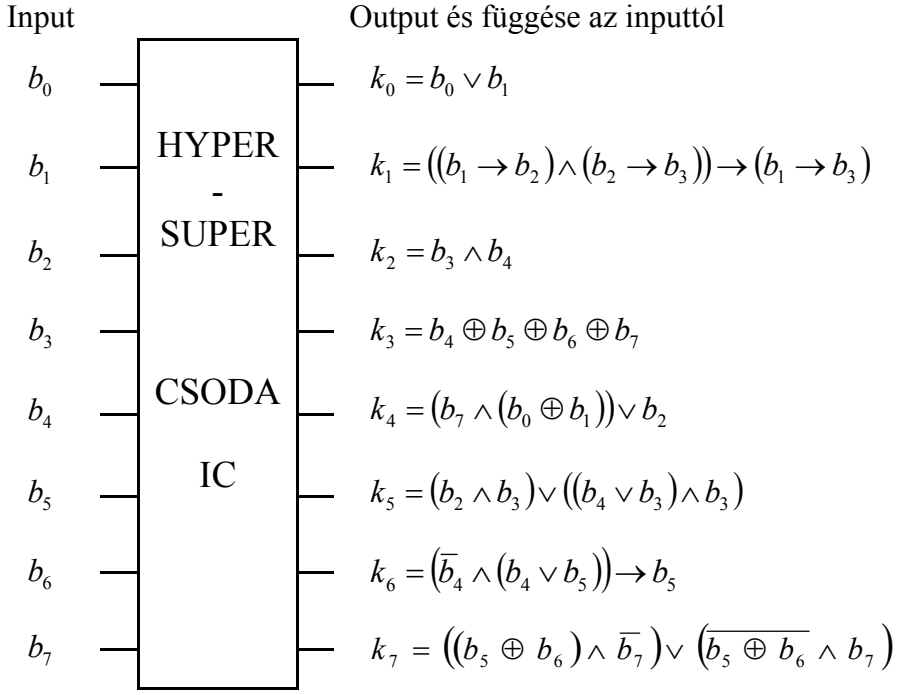
A fenti 1.-10. tulajdonságokat formula átalakítási szabályoknak is tekinthetjük, amelyeket a munkánk során felhasználhatunk. Valójában ezen szabályok mélyebb értelemmel bírnak, mivel más, általánosabb struktúrák definíciójaként lépnek fel. A fentiek alapján más összefüggéseket is kimutathatunk.

Példa: $x \leftrightarrow y \equiv \overline{x \oplus y}$

Bizonyítsuk be, hogy a kizáró vagy műveletre teljesül az asszociativitás, a kommutativitás és hogy $x \oplus x = h$ és $x \oplus y \oplus x = y$!

FELADATOK

1. a. Adja meg művelettábláival mind a 16 bináris logikai műveletet!
   b. Fejezze ki mind a négy unáris műveletet a NEM, ÉS, VAGY bináris műveletekkel!
2. Fejezze ki a felsorolt hét bináris műveletet a NEM, ÉS, VAGY műveletekkel!
3. Fejezze ki a felsorolt hét bináris műveletet csak a NEM-mel, és a VAGY-gyal!
4. a. Fejezze ki a NEM, ÉS, VAGY műveleteket csak a Peirce nyíllal!
   b. Fejezze ki a NEM, ÉS, VAGY műveleteket csak a Scheffer vonással!
5. A formulák átalakításával lássa be, hogy az alábbi egyenlőségek azonosságok!
   a. $\overline{x \vee y \vee z} = \bar{x} \wedge \bar{y} \wedge \bar{z}$ és $\overline{x_1 \vee x_2 \vee \ldots \vee x_n} = \bar{x}_1 \wedge \bar{x}_2 \wedge \ldots \wedge \bar{x}_n$
   b. $x \to (y \vee z) = (x \to y) \vee (x \to z)$
   c. $x \wedge (y \leftrightarrow z) = (x \wedge y) \leftrightarrow (x \wedge z)$
   d. $(\bar{x} \wedge \bar{z}) \vee (x \wedge y) \vee (x \wedge \bar{z}) = (x \wedge \overline{y \wedge z}) \vee (\bar{x} \wedge z)$
   e. $(x \to y) \to ((x \wedge \bar{y}) \oplus (x \leftrightarrow \bar{y})) = (x \vee y) \wedge (\bar{x} \vee \bar{y})$
6. Adott egy IC (integrált áramköri tok), amelynek 8 bemeneti lába (b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7), és 8 kimeneti lába (k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7) van. A bemenetek is és a kimenetek is egy-egy byte-ba foglalhatók össze, a byte bitjeit az indexek sorszámozzák. A bemenetek és a kimenetek közötti összefüggéseket is megadtuk a kimenetek mellett.



   A bemenetre adott jeleket, mint egybyte-os előjel nélküli egész számot adjuk meg, mely bitjeire bontandó. Határozza meg a kimenet bitjeit és adja meg azt a bemenet mintájára egybyte-os előjel nélküli egész számként decimálisan az alábbi inputok esetére: 159, 163, 152, 89, 108, 205, 75, 96
7. Igazolja a művelettáblák segítségével a NEM, ÉS, VAGY műveletek tulajdonságai fentebbi táblázatában felsorolt 1.-10. tulajdonságot!

8. Tervezzen hárombemenetes többségi elven működő szavazó automatát, azaz a három bemenet ismeretében a kimeneten az jelenjen meg, amelyik bemenetből több van. (Többségi elv érvényesül.) Igyekezzen minél egyszerűbbre tervezni!
9. Tervezzen olyan automatát, amely 3 bemenő jel (0, 1) esetén azokat számoknak tekintve a kimeneten a legkisebbet (legnagyobbat) jeleníti meg!
10. Tervezzen olyan automatát, amely 4 bemenő jel (0, 1) esetén a kimeneten 1-et jelenít meg, ha a bemenetben az egyesek száma páros és nullát, egyébként! (Páros paritás jelző.)
11. Tervezzen olyan automatát, amely 3 bemenő jel (0, 1) esetén akkor ad egyest a kimeneten, ha mindegyik bemenet egyes, egyébként zérust ad! (Mindenki megszavazta.)
12. Tervezzen olyan automatát, amely 3 bemenő jel (0, 1) esetén akkor ad egyest a kimeneten, ha mindhárom bemenet egyforma (zérus, vagy egyes), egyébként zérust ad! (Mindenki egyetértett, ugyanúgy vélekedett.)
13. Tervezzen olyan automatát, amely 3 bemenő jel (0, 1) esetén akkor ad egyest a kimeneten, ha bármelyik bemenet egyes volt, egyébként zérust ad! (Vétó.)