While ciklusban maradási feltétel levezetése

De Morgan-azonosságok alapján

Ugye **While** ciklus esetén, pont fordítva kell megadni azt, amit amúgy egy **if**-ben vizsgálnánk, lévén, hogy bent szeretnénk maradni a ciklusmagban mindaddig, amíg az ellenkezője (tehát amit mi keresünk) nem teljesül.

Egyszerű példa:

```
string etel = "";
do
{
    etel = Console.ReadLine();
} while (etel != "alma");
```

Ebben a példában, mindaddig kérünk be a felhasználótól a konzolról valamit, amíg azt nem kapjuk meg, hogy „alma”.

Amikor a felhasználó „alma”-t fog begépelni, akkor a feltétel kiértékelése HAMIS lesz, hiszen az **etel** változó tartalma alma lesz. **alma != alma** pedig hamis kiértékelést fog maga után vonni.

Ez rendben is van, de összetett feltételek esetén már nagyon meg tud bonyolódni a helyzet.

```
while ( (neptunKod != "NEPTUN" || jelszo != "jelszo123") && (neptunKod != "ABC11A" || jelszo != "alma123") );

while ( !(neptunKod == "NEPTUN" && jelszo == "jelszo123" || neptunKod == "ABC11A" && jelszo == "alma123") );
```

A feladat az volt, hogy mindaddig maradjunk a ciklusban és kérjünk be újra és újra adatot a felhasználótól, amíg nem a számunkra megfelelőeket adja.

Ebben az esetben a neptunKod változónak a tartalma meg kell egyezzen a „NEPTUN” karaktersorozattal, a jelszo változónak pedig a „jelszo123” karaktersorozattal.

VAGY

a neptunKod változónak a tartalma meg kell egyezzen a „ABC11A” karaktersorozattal, a jelszo változónak pedig a „alma123” karakterosozattal.

Csak e két esetben léphetünk ki a ciklusból. Tehát, addig szeretnénk bent maradni, amíg ezek nem teljesülnek. Ez azt jelenti, hogy le kell tagadnunk az egész vagy egyes kifejezéseket a feltételben. Egyszerű példa esetén, mint a fentiben ez gond nélkül megy, de összetett feltétel esetén már könnyen bele lehet zavarodni. Nézzük meg, hogy hogyan lehet hibátlanra levezetni saját magunknak!

Opció 1.

Felírjuk az egész feltételt (ahogy magunkban is elmondjuk), majd az **egészet** letagadjuk. Ez látható a while-os példán az alsó sorban. !(…)

Opció 2.

Szeretnénk az egészet tagonként letagadni, azonban ekkor változtatni kell a kifejezések közti logikai kapcsolatokon is! Erre ad jól átlátható módszert, ha De Morgan-azonosságokkal levezetjük magunknak.

Első lépés, állapítsuk meg a kifejezéseket, amelyek majd egy-egy tagként fognak szerepelni.

A tag: `neptunKod == "NEPTUN"`

B tag: `jelszo == "jelszo123"`

C tag: `neptunKod == "ABC11A"`

D tag: `jelszo == "alma123"`

Itt látható, hogy A tag állít valamit, ahogy B, C és D tag is. Például, A tag azt az állítást képviseli, hogy a neptunKod egyenlő-e a NEPTUN karakterlánccal.

Látható az eredeti képen, hogy a tagok között logikai kapcsolatok is fennállnak. És (&&) illetve vagy (||) kapcsolatok. Ezek az új, általunk létrehozott tagok között is megmaradnak:

A && B || C && D

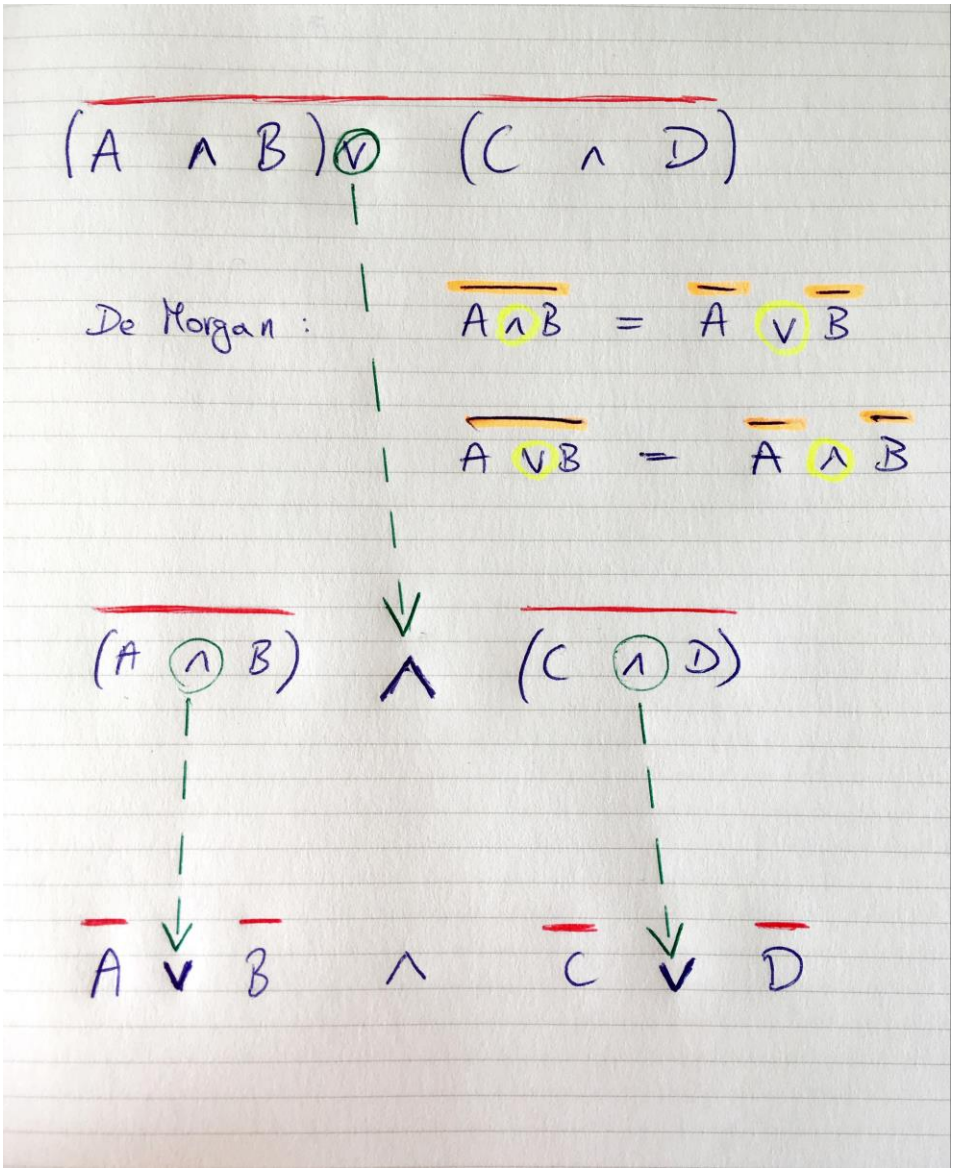
Mivel a De Morgan-azonosságoknál más jelölést használunk, ezért írjuk át mi is:

&& helyett ∧

|| helyett ∨

Tehát átírva teljesen: $A \wedge B \vee C \wedge D$

Mivel ez az egész állítást tagadva volt a fenti példában a ! jel miatt, ezért mi is tagadjuk le.



A végén pedig látható, hogy mire a levezetés végéhez értünk, pont ugyan azt kaptuk, mint a felső esetben:

```
while ( (neptunKod != "NEPTUN" || jelszo != "jelszo123") && (neptunKod != "ABC11A" || jelszo != "alma123") );
```

Az egyes tagok (A, B, C, D) ugyan azokat az állításokat jelölik, viszont mivel itt külön le van tagadva minden egyes állítás, ezért ezt nekünk is meg kell tennünk, amely kódban a != -ként jelenik meg.

Látható, hogy !A és !B tag között VAGY kapcsolat lett, csak úgy, mint !C és !D tag között. A két „nagyobb” tag között pedig VAGY kapcsolat lett.