Kort intro til digtek-modulen

Importer modulen

In [1]:
```
from digtek import *
```

1 Funskjoner

Digtek-modulen introduserer en BolskFunskjon (<BoolFunction>) type i python. Disse funskjonene kan defineres på forskjellige måter og brukes nesten på samme måte som vanlige funskjoner. I tilegg har bolske funskjoner spesielle egenskaper som gjør at de egnes godt til å regne med i mange 'digitalteknikk' sammenhenger. Det finnes tre forskjellige bolske funskjoner med hver sin måte å definere de på:

1.1 Lambda-Funskjoner

Lambdafunksjoner defineres på følgende måte:

In [2]:
```
foo = LambdaFunction(lambda a,b,c: a and b or c and b)
```

1.2 Minterm-Funksjoner

Mintermfunskjoner defineres ved en tuple/liste/set med mintermer etterfulgt av hvor mange variabler funksjonen har. For eksempel kan en 4-variabel-minterm-funksjon defineres på følgende måte:

In [3]:
```
bar = MintermFunction((6,7,9),4)
```

1.3 Maksterm-Funksjoner

Makstermfunskjoner defineres på akkurat samme måte som en mintermfunskjon, bare at nå definerer vi makstermene i stede for mintermene:

In [4]:
```
baz = MaxtermFunction((3,4),4)
```

1.4 Nyttig om bolske-funskjoner

Det er mulig å printe bolske funksjoner på en fin måte. Da brukes *print()*-funskjonen:

In [5]:
```
print(bar)
print(baz)
```

$$F(x,y,z,w) = \Sigma(9,6,7) = \bar{x}yz\bar{w} + \bar{x}yzw + x\bar{y}\bar{z}w$$

$$F(x,y,z,w) = \Pi(3,4) = (x + y + \bar{z} + \bar{w})(x + \bar{y} + z + w)$$

de bolske-funksjonene har også en innebygd *print()*-funskjon som kan printe med egendefinerte vedier både for variabler og funskjonsnavn. Her er det verdt å merke seg at også *LaTeX* verdier aksepteres som funskjons- og variabel-navn!

In [6]:
```
bar.print(name="\\Upsilon",var=("\\alpha","\\beta","\\gamma","\\delta"))
baz.print(name="FoObAr",var="abxy")
```

$$\Upsilon(\alpha,\beta,\gamma,\delta) = \Sigma(9,6,7) = \bar{\alpha}\beta\gamma\bar{\delta} + \bar{\alpha}\beta\gamma\delta + \alpha\bar{\beta}\bar{\gamma}\delta$$

$$FoObAr(a,b,x,y) = \Pi(3,4) = (a + b + \bar{x} + \bar{y})(a + \bar{b} + x + y)$$

Men! Funskjoner er jo ikke til bare for å se pene ut, vi kan også, som nevnt, bruke funskjonene som vanlige python-funskjoner. Her er det også mulig å sammenligne to funskjoner for å se om de er like/ulike. Dette kan komme godt med på eksamen!

In [7]:
```
print( foo(1,0,1) )
print( bar(*(1,1,0,0)) )
print( foo == bar )
```

0

0

False

Sterkheten til disse funskjonene kommer i form av metodene/funskjonene man kan bruke på de, som fører oss til neste del:

2 Innebygde metoder/funskjoner

2.1 sannhetstabeller

Det er mulig å printe sannhetstabellen til en eller flere funskjoner ved å bruke funksjonen '*table()*'. For å sammenligne flere funskjoner kan man kalle funskjonen med flere **boolFunctions** som argumenter.

In [8]:
```
table(bar,baz)
```

x	y	z	w	F_0	F_1
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	0	1

2.2 karnaugh-diagram

Karnaugh-diagram kan enkelt tegnes ved hjelp av 'karnaugh()'-funskjonen. Som argumenter tar denne funskjonen en bolsk-funskjon, og eventuellt variabelnavn:

In [9]:
```
a = LambdaFunction(lambda x,y,z,w,u,v: x and y and z and w or not x and y and v a
karnaugh(a,var="xyzwuv")
```

		y=0 uv				y=1 uv			
		00	01	11	10	00	01	11	10
x=0	zw 00	0	0	0	0	0	0	0	0
	zw 01	0	0	0	0	0	1	1	0
	zw 11	0	0	0	0	0	1	1	0
	zw 10	0	0	0	0	0	0	0	0
x=1	zw 00	0	0	0	0	0	0	0	0
	zw 01	0	0	0	0	0	0	0	0
	zw 11	0	0	0	0	1	1	1	1
	zw 10	0	0	0	0	0	0	0	0