Baggrundsnote om logiske operatorer

Man kan regne på udsagn ligesom man kan regne på tal. Regneoperationerne kaldes da logiske operatorer. De tre vigtigste logiske operatorer er NOT, AND og OR.

Den første NOT har kun ét argument (og svarer til et fortegnsskift). Den er defineret ved sandhedstavlen

p	NOT p
TRUE	FALSE
FALSE	TRUE

dvs. den skifter sandhedsværdi: TRUE bliver til FALSE og omvendt. Anvendes den to gange fås derfor identiteten:

$$\text{NOT (NOT } p) = p$$

Koder vi sandhedsværdierne med 0 (for FALSE) og 1 (for TRUE) svarer det til regneoperationen

$$\text{NOT } p = 1 - p$$

De to andre logiske operatorer har begge netop to argumenter (og svarer lidt til regneoperationerne plus og gange). De er defineret ud fra sandhedstavlerne

p	q	p AND q
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

p	q	p OR q
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

dvs. AND er kun sand, når begge udsagnene er sande, ligesom OR kun er falsk, når begge udsagnene er falske, dvs. OR er kun sand når mindst et af udsagnene er sandt.

Koder vi igen sandhedsværdierne med 0 og 1 svarer det denne gang til regneoperationerne

$$p \text{ AND } q = p \cdot q$$
$$p \text{ OR } q = p + q - p \cdot q$$

Læg mærke til det ekstra led i regneoperationen svarende til OR, der altså ikke helt opfører sig som plus! Det ekstra led sikrer netop at 1 OR 1 giver 1. Hvis dette led ikke havde været der, havde de logiske operationer opført sig fuldstændigt som gange og plus.

Det er nemt at anvende AND og OR flere gange I træk. Vi behøver ikke engang sætte parenteser (dvs. de opfylder den associative lov).

Det samlede udsagn

$$p_1 \text{ AND } p_2 \text{ AND } \ldots \text{ AND } p_n$$

er sandt, netop når alle udsagnene er sande.

Det samlede udsagn

$$p_1 \text{ OR } p_2 \text{ OR } \ldots \text{OR } p_n$$

er falsk, netop når alle udsagnene er falske (og dermed sandt netop når mindst ét af deludsagnene er sande).

De har altså så at sige modsat opførsel og kan derfor knyttes sammen via NOT operationen (hvor parenteserne på højre side rent faktisk er overflødige):

$$\text{NOT } (p_1 \text{ AND } p_2 \text{ AND } \ldots \text{ AND } p_n) = (\text{NOT } p_1) \text{ OR } (\text{NOT } p_2) \text{ OR } \ldots \text{ OR } \ldots (\text{NOT } p_n)$$
$$\text{NOT } (p_1 \text{ OR } p_2 \text{ OR } \ldots \text{ OR } p_n) = (\text{NOT } p_1) \text{ AND } (\text{NOT } p_2) \text{ AND } \ldots \text{ AND } (\text{NOT } p_n)$$

Det er nemt at formalisere et sammensat udsagn som siger at mindst et af deludsagnene er sandt, idet det sammensatte udsagn

$$p_1 \text{ OR } p_2 \text{ OR } \ldots \text{OR } p_n$$

jo netop betyder at mindst ét af udsagnene er sandt.

Men det er svært at formalisere et sammensat udsagn som siger at *højst* et deludsagnene er sandt eller *netop* et af deludsagnene er sandt.

Man kunne tro at det ville hjælpe at indføre den logiske operator XOR (exclusive or), dvs. enten eller, men ikke dem begge, med sandhedstavlen

p	q	p XOR q
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

dvs. XOR er sand netop når de to udsagn har forskellige sandhedsværdier.

Koder vi sandhedsværdierne med 0 og 1 svarer det til regneoperationen

$$p \text{ XOR } q = p + q - 2p \cdot q = p \cdot (1-q) + (1-p) \cdot q.$$

Den virker fint på to udsagn, men anvender vi den på mange udsagn skal vi passe på. Ser vi fx på tre udsagn fås sandhedstavlen (hvor vi regner fra venstre mod højre):

p	q	p XOR q	r	p XOR q XOR r
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	FALSE	TRUE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE
FALSE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	TRUE	FALSE	FALSE	FALSE
TRUE	TRUE	FALSE	TRUE	TRUE

Problemet ligger i den sidste række, hvor tre gange TRUE giver TRUE, fordi de to første giver FALSE! Udvides ræsonnement ser vi at det sammensatte udsagn

$$p_1 \text{ XOR } p_2 \text{ XOR } \ldots \text{ XOR } p_n$$

er sandt netop når der er et *ulige* antal sande deludsagn.

Det er altså nemt at formalisere en regel om at et lige eller et ulige antal deludsagn er sande eller falske. Men nu var det jo i stedet for regler af typen: højst ét af deludsagnene eller netop ét af deludsagnene, der skulle være sande (eller falske).

I stedet må vi så opbygge sådanne sammensatte udsagn fra bunden: Hvis vi fx vil opbygge et udsagn som at *netop ét af udsagnene skal være sandt*, så svarer det til at hævde følgende påstand:

Enten er det p_1 som er sand og de øvrige deludsagn, som er falske
eller også
er det p_2 som er sand og de øvrige deludsagn som er falske
eller også …
eller også
er det p_n som er sand og de øvrige deludsagn, som er falske

Men det kan vi jo oversætte til følgende logiske udsagn:

p_1 AND NOT p_2 AND NOT p_3 AND … AND NOT p_n
or
NOT p_1 AND p_2 AND NOT p_3 AND … AND NOT p_n
or
…
or
NOT p_1 AND NOT p_2 AND NOT p_3 AND … AND p_n

Hvis det *er højst et af udsagnene som skal være sandt* så må vi yderligere tilføje udsagnet

$$\text{NOT } p_1 \text{ AND NOT } p_2 \text{ AND } \ldots \text{ AND } p_n$$

der også tillader alle udsagnene at være falske. Læg i øvrigt mærke til, at vi ikke har haft brug for parenteser, fordi det er underforstået at AND binder hårdere end OR (ligesom gange binder hårdere end plus), ligesom det er underforstået at NOT binder hårdere end AND og OR. Det sidste varierer dog lidt fra computer algebra system til computer algebra system. Et fortegnsskift burde binde hårdere end gange og plus, men her skal man passe på! Fx vil mange computer algebra systemer give $-2^4 = -16$, fordi de lader potenser binde hårdere end fortegnsskift. Det samme gælder håndteringen af NOT-operatoren.

Ethvert sammensat udsagn kan nu omskrives på *standardformen*:

(…AND…AND…AND…) OR (…AND…AND…AND…) OR …

hvor de enkelte argumenter enten er et deludsagn *p* eller dets negation NOT *p*. Det svarer til omskrivning af et algebraisk udtryk til en flerleddet størrelse, hvor de enkelte led består af produkter, og hvor leddene er bundet sammen af additioner.

Standardformen er bl.a. bekvem fordi enhver sandhedstavle umiddelbart kan oversættes til et logisk udsagn på standardform (med andre ord: Sandhedstavlen er blot en anden repræsentation af standardformen). Vi kan fx se på sandhedstavlen:

p	*q*	*r*	Sammensat udsagn
TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE
TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE
FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE

Vi starter da med at notere hvilke kombinationer, der giver TRUE (hvis der slet *ikke* er nogen er vi færdige, for så er der blot tale om det konstante udsagn FALSE):

p	*q*	*r*	Sammensat udsagn
TRUE	TRUE	TRUE	TRUE
TRUE	TRUE	FALSE	TRUE
TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	FALSE	FALSE
FALSE	TRUE	TRUE	FALSE
FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	TRUE

Til hver række, der giver TRUE knytter vi så et led, idet vi bruger *p*, hvis deludsagnet *p* har værdien TRUE og NOT *p*, hvis deludsagnet *p* har værdien FALSE osv. Så sandhedstabellen læses således:

Det samlede udsagn er sandt, hvis der enten gælder at
p er sand og q er sand og r er sand (1. række)
eller der gælder
p er sand og q er sand og r er falsk (2. række)
eller der gælder
p er sand og q er falsk og r er sand (3. række)
eller der gælder
p er falsk og q er falsk og r er sand (7. række)
eller der gælder
p er falsk og q er falsk og r er falsk (8. række)

Men det kan vi jo så oversætte led for led:

p	*q*	*r*	Sammensat udsagn	
TRUE	TRUE	TRUE	TRUE	*p* AND *q* AND *r*
TRUE	TRUE	FALSE	TRUE	*p* AND *q* AND NOT *r*
TRUE	FALSE	TRUE	TRUE	*p* AND NOT *q* AND *r*
TRUE	FALSE	FALSE	FALSE	
FALSE	TRUE	TRUE	FALSE	
FALSE	TRUE	FALSE	FALSE	
FALSE	FALSE	TRUE	TRUE	NOT *p* AND NOT *q* AND *r*
FALSE	FALSE	FALSE	TRUE	NOT *p* AND NOT *q* AND NOT *r*

Det sammensatte udsagn er altså givet ved

p AND *q* AND *r*
OR
p AND *q* AND NOT *r*
OR
p AND NOT *q* AND *r*
OR
NOT *p* AND NOT *q* AND *r*
OR
NOT *p* AND NOT *q* AND NOT *r*

De tre logiske operatorer NOT, AND og OR er altså tilstrækkelige til at opbygge et vilkårligt logisk udsagn. I kraft af identiteterne

$$p \text{ OR } q = \text{NOT (NOT } p \text{ AND Not } q)$$

$$p \text{ AND } q = \text{NOT (NOT } p \text{ OR NOT } q)$$

kan man faktisk undvære enten OR eller AND (men ikke dem begge!). Men hvis man ønsker det kan man også nøjes med en enkelt binær operator, blot ikke AND (eller OR). Inden vi viser det kan vi lige danne os et overblik over alle de mulige binære logiske operatorer. Der er præcis fire mulige kombinationer af sandhedsværdierne for *p* og *q*. Da hver af dem har netop to mulige sandhedsværdier bliver der i alt $2^4 = 16$ mulige binære logiske operationer. Halvdelen af dem er symmetriske, dvs. de afhænger ikke af rækkefølgen for *p* og *q*, dvs. de skelner ikke mellem hvem af dem der er sand og hvem af dem der er falsk, idet tilfælde hvor den ene er sand og den anden falsk.

p	*q*	T	AND	XOR	NOR	OR	EQ ⇔	NAND	F	CIMP q⇒p	IMP p⇒q	*p*	*q*	NOT q	NOT p		
T	T	T	T	F	F	T	T	F	F	T	T	T	T	F	F	F	F
T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
F	T	T	F	T	F	T	F	T	F	F	T	F	T	F	T	F	T
F	F	T	F	F	T	F	T	T	F	T	T	F	F	T	T	F	F

Her står NOR og NAND for negationen til OR og AND. Tilsvarende står EQ for Equivalent, hvilket er negationen til XOR og hævder at de to udsagn har samme sandhedsværdi (hvor XOR hævdede de havde modsat sandhedsværdi). Ækvivalensoperatoren EQ har samme problem som den eksklusive enten eller operation: Den kan *ikke* uden videre udvides til flere argumenter (uden at mi-

ste sin umiddelbare betydning, nemlig at alle argumenterne har samme sandhedsværdi). Endelig står IMP for imply (implikation), dvs. medfører kommandoen, hvor vi slutter forlæns fra p til q: Hvis p så q. Tilsvarende står CIMP for Converse implication, hvor vi slutter baglæns fra q til p, dvs. hvis q så p.

Implikationen er en fuldstændig central logisk operator (og også den sværeste at tilegne sig), idet den indgår i formaliseringen af et matematisk bevis. Hvis vi fx vil bevise en sætning q kan vi gøre det *direkte* ved at gå ud fra en kendt sætning p og så vise at p medfører q:

DIREKTE BEVIS: p AND (p IMP q) IMP q

Tilsvarende kan vi vise en sætning q ved hjælp af et *indirekte* bevis, idet vi viser at den modsatte sætning NOT q medfører en modstrid med en kendt sætning p.

INDIREKTE BEVIS: p AND (NOT q IMP NOT p) IMP q

De to sidste operationer har ikke traditionelle navne, men der er selvfølgelig bare tale om negationerne til de to implikationer.

Til slut vil vi se lidt på oversættelsen fra dagligdags sprog til logiske udsagn. En sådan oversættelse er behæftet med de samme tvetydigheder som alle andre oversættelser. Her noterer vi os bl.a. at man i almindelige sætninger ikke bruger parenteser som man gør i logiske udsagn, hvorfor rækkefølgen af operationerne kan være tvetydig. Oversættelsen er heller ikke entydig fordi det dagligdags sprog er mere fleksibelt og kan kæde udsagn sammen på flere måder. I stedet for at bruge 'og' kan man fx kæde flere deludsagn sammen ved at adskille dem med kommaer. Endelig er dagligdags ord ofte tvetydige. Fx er det uklart om en sætning af formen

enten … eller …

udelukker at begge deludsagnene kan være opfyldt. Vi kunne være tilbøjelige til at mene at brugen af ordet 'enten' typisk skærper sætningen og derfor udelukker at de begge kan være opfyldt, men praksis er ikke entydig. I logik fokuserer vi på OR (i modsætning til XOR), fordi den umiddelbart kan udvides til mange deludsagn uden at miste sin grundlæggende betydning.

ikke …	NOT p
enten … eller …	p OR q
både … og …	p AND q
hverken … eller …	p NOR q = NOT (p OR q)
hvis … så …	p IMP q
… medfører …	p IMP q
… forudsat …	p CIMP q = q IMP p
… med mindre …	p NEQ q = p XOR q
… eller … eller …	p_1 OR p_2 OR p_3 …
… og … og …	p_1 AND p_2 AND p_3 …