Boole och logikens utveckling

Av: Wai-Ming Lak

Inledning

Boolesk algebra är en gren inom diskret matematik. Men det är även sätt att representera logik genom matematiska symboler. Denna algebra har fått sitt namn utav den engelska matematikern George Boole (1815-1864), men har existerat implicit i språk och logik tusentals år dessförinnan. Ur det historiska perspektivet i denna uppsats kommer vi dock att räkna Lullus arbeten om kombinatorik på 1200-talet som startpunkten för Boole-algebrans och dess relaterade algebrors utveckling.

Boolesk algebra är isomorf med mängdalgebra. Addition och multiplikation i boole-algebra motsvaras av unionen respektive snittet i mängdalgebra.

Eftersom boole-algebran går att översätta till strömbrytaralgebra där ström motsvarar 1 och ingen ström motsvarar 0, så kan man bygga logiska kretsar, s.k. operatorer. Detta är grunden till hur alla datorer fungerar på den rent fysiska nivån.

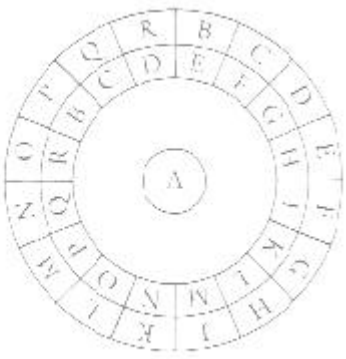
Från enkla logikmaskiner till intervall- och mängdalgebra

Lullus och kombinatorik

Logik, vilket ligger i grund för både matematik och språk, har utan tvekan funnits bland människor sedan urminnes tider. Raimundus Lullus (1235-1315) skrev, efter att ha mottagit gudomliga uppenbarelser på berget Randa på Mallorca, ett verk om kombinatorik, Ars Magna. Detta verk anses av somliga vara den första ansatsen till formell logik.

Nedan är ett exempel på principen av en av Lullus kombinatorikmaskiner:

A-cirkeln representerar Gud; bokstäverna B, C, D, etc står för olika egenskaper hos Gud: B för godhet, C för storhet, D för evighet, etc.
I figuren, där DR visar att Gud är evig och tålmodig, är den inre skivan vriden 3/16 varv motsols från läget BB, CC, DD, etc



Ett modernt exempel på en kombinatorikmaskin är följande:

Med hänsyn till	Börsnoteringarna	Bör man	Diskutera	Ökad investering
Oavsett	Ekonomin	Måste man	Undersöka	Förbättrad utlandsservice
I beaktande av	Dagsläget	Vill styrelsen	Dryfta	Nya rutiner
Med tanke på	Företagsklimatet	Behöver bolaget	Ifrågasätta	Gamla dotterbolag

Genom att kombinera ett ord från varje kolumn får man en mening. Byter man sedan ut ett ord i en kolumn mot ett annat i samma kolumn får man en ny mening. Totalt finns det 4^5 olika meningar man kan bilda (4 ord i varje kolumn, 5 ord i varje mening).

Leibniz intervallalgebra

Det sägs vara utifrån Lullus verk som den då nittonåriga Gottfried Wilhelm von Leibniz (1646-1716) 1666 skrev sin essä ”Dissertio de Arte Combinatoria” (Avhandling om konsten att kombinera). Här ville Leibniz uppfinna ett språk som skulle reducera alla förnuftets sanningar till ett slags kalkyl, dvs. det vi idag kallar symbolisk logik. Leibniz kom dock inte ens i närheten av att lyckas med att nå detta mål.

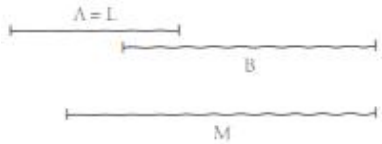
Emellertid påträffades två fragment (numrerade XIX och XX) i Leibniz senare verk (skrivna omkring 1685) där han definierade sin intervallalgebra. I intervallalgebra är mängden man jobbar med definierad som intervall (sträckor). Två intervall kan slås ihop till ett nytt intervall med operationen +. De två grundläggande axiomen för denna algebra är:

B + N = N + B, summan av två intervall är kommutativ

Och

A + A = A, om inget nytt läggs till så uppstår inget nytt, dvs. upprepning ändrar inget.

Ett exempel på vad fragment XX innehöll ges här:



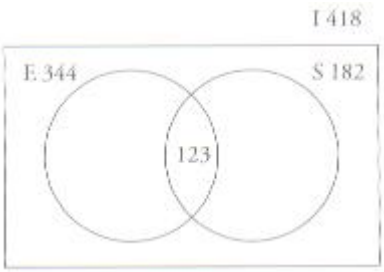
"Proposition 10: Om A = L och B = M, så gäller A + B = L + M"

I anslutning till detta resultat noterar Leibniz att om A + B = L + M och A = L, så följer därav ej att B = M, dvs. man kan inte deducera att B = M bara för att de första två påståendena stämmer.

Mängdalgebra

Denna intervallalgebra sådde fröet till det som vi idag kallar mängdalgebra, som egentligen bara är en vidareutveckling av intervallalgebran. De linjediagram som används för att illustrera och lösa logiska problem i intervallalgebra ersattes av den engelske matematikern John Venns (1834-1923) s.k. Venn-diagram. För att illustrera användbarheten av Venn-diagram vid lösning av logiska problem kan vi nyttja följande exempel: ”Av de 418 turister på en medelhavskryssning behärskade 344 engelska och 182 spanska. 123 turister behärskade både engelska och spanska. Fanns det turister som inte behärskade något av språken?”

För att lösa detta problem med hjälp av mängdalgebra underlättar det om vi först och främst översätter frågan från svenska till mängdalgebrans mer logiska språk. **Grundmängden** är den mängd som innehåller alla övriga mängder involverade i problemet – i detta fall, antalet turister (I). Denna representeras av en rektangel eller ett liggande blankt pappersark. I grundmängden representerar vi delmängder i form av cirklar, i det här fallet är de två delmängderna antalet turister som kan engelska (E) respektive antalet som kan spanska (S). De som både kan spanska och engelska är **snittet** mellan S och E. De som behärskar minst ett av språken är **unionen** mellan S och E, vilket räknas ut genom att ta summan av delmängdernas element subtraherat med snittet, så att vi inte räknar samma element (personer) två gånger. Unionen är alltså 344 + 182 – 123 = 403.



Eftersom unionen är mindre än grundmängden, dvs. de som kan minst ett av språken är mindre i antal än det totala antalet turister, kan vi därav entydigt deducera att det fanns turister som inte kunde något av språken.

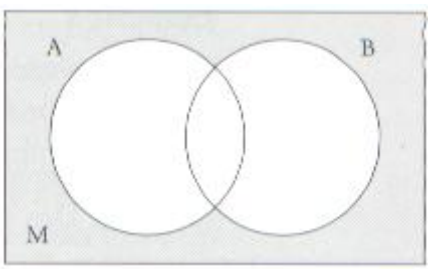
De matematiska symboler som vi idag använder i mängdlära kom från den italienska vektoranalytikern Giuseppe Peano (1858-1932). Med hjälp av dessa symboler kan vi kort och koncist uttrycka följande åtta regler som utgör identiteterna i mängdalgebrans axiomsystem:

$$A \cup A = A \quad (1a)$$
$$A \cap A = A \quad (1b)$$
$$A \cup B = B \cup A \quad (2a)$$
$$A \cap B = B \cap A \quad (2b)$$
$$A \cup (B \cup C) = (A \cup B) \cup C \quad (3a)$$
$$A \cap (B \cap C) = (A \cap B) \cap C \quad (3b)$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \quad (4a)$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad (4b)$$
$$A \cup I = I \quad (5a)$$
$$A \cap \emptyset = \emptyset \quad (5b)$$
$$A \cup \emptyset = A \quad (6a)$$
$$A \cap I = A \quad (6b)$$
$$A \cup A' = I \quad (7a)$$
$$A \cap A' = \emptyset \quad (7b)$$
$$(A')' = A \quad (8)$$

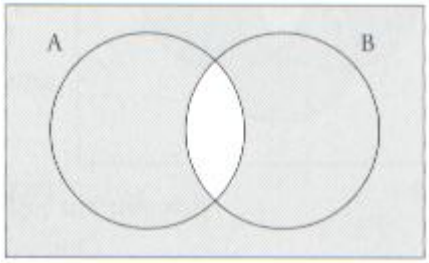
Notera att den första identiteten (1a) är identisk med Leibniz första axiom som vi förut diskuterade. Denna typ av uppfyller Göttingenprofessorn David Hilberts (1862-1943) tre egenskaper som kännetecknar ett optimalt axiomsystem: ”systemet är fullständigt, axiomen är oberoende av varandra oberoende och systemet är motsägelsefritt”.

Två viktiga identiteter inom mängdalgebran kallas för de Morgans lagar. Dessa är uppkallade efter den indiskfödde Augustus de Morgan (1806-1871) som var matematikprofessor i London och kom att betyda mycket för George Boole på 1840-talet. De Morgans lagar kan illustreras med hjälp av Venn-diagram:

(1) $(A \cup B)' = A' \cap B'$



(2) $(A \cap B)' = A' \cup B'$



Boolesk algebra och relaterade algebror

George Boole

Engelsmannen George Boole föddes och växte upp i Lincoln. Han fick undervisning i elementär matematik, latin och grekiska och gav prov på stor språkbegåvning. Redan som 16-åring anställdes han som lärare och vid 20 års ålder startade han en egen skola. Som autodidakt återupptog Boole matematikstudierna och blev senare professor i ett college i Cork efter en rekommendation av de Morgan. År 1847 publicerades hans bok, The Laws of Thought, som kom att placera Boole i centrum för matematisk logik.

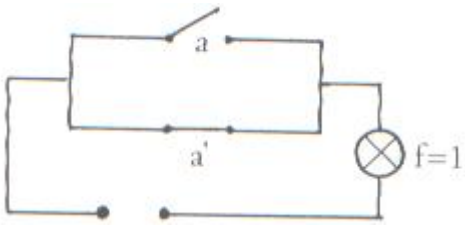
Isomorfi mellan algebror

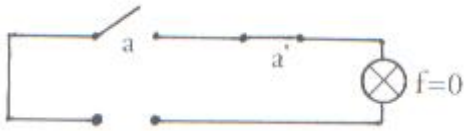
Innan har vi sett att det bara var en ytlig skillnad mellan intervallalgebra och mängdalgebra – samma axiom med olika symboler. Därför är det möjligt att översätta från det ena till det andra. Likadant är det mellan mängdalgebra och boolesk algebra. Unionen skrivs här med operatorsymbolen + och snittet med * (multiplikationstecknet som prioriteras före additionstecknet). En Boole-algebra, B, har 4 axiom som motsvarar de i mängdalgebran:
A1 Operationerna (+) och (*) är kommutativa, vilket motsvarar (2a) och (2b) ovan.
A2 B innehåller identitetselementen 0 och 1 relativt operationerna (+) och (*), (6a) & (6b)
A3 Operationerna (+) och (*) är distributiva över varandra, (4a) & (4b)
A4 För varje element a tillhörande B existerar en additiv och multiplikativ invers a’ i B, (7a) & (7b)

1 motsvarar grundmängden och 0, den tomma mängden. Det är därför logiskt att alla element, dvs. delmängder, måste ligga mellan dessa två. Alltså för alla element a i en boole-algebra gäller $0 \leq a \leq 1$. Boolesk algebra är alltså isomorf med mängdalgebra. Senare kommer vi att diskutera switching algebra och logikalgebra, vilka också är syskonalgebror till mängdalgebran.

Switching algebra

MIT-professorn Claude Shannon (1916-2001) gjorde 1938 i sin master thesis en banbrytande upptäckt som kopplade samman boole-algebra med elektriska kretsar. Genom att studera enkla kopplingsnät med två strömbrytare och en lampa, fann Shannon att ellärans två grundläggande kopplingar seriekopplingen och parallellkopplingen motsvarar boole-algebrans * respektive + operationer. Om vi kallar de två strömbrytarna för a och b och deras läge för 1 och 0 för på respektive av; och vi kallar lampan f och dess läge 1 och 0 för lyser respektive lyser inte så kan vi illustrera några exempel från boole-algebran med hjälp av kopplingsscheman:

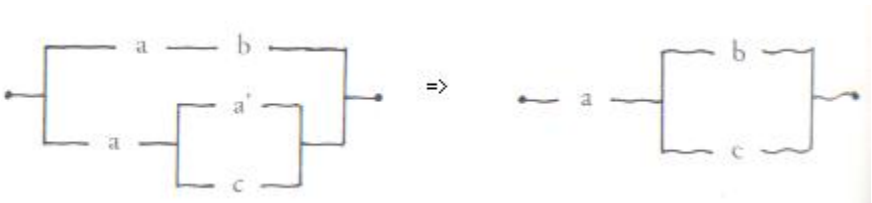




Parallellkoppling: a + a’ = f
a=0, a’=1, 0 + 1 = 1
=> f = 1

Seriekoppling: a * a’ = f
a=0, a’=1, 0 * 1 = 0
=> f = 0

Genom att använda detta samband kan man ställa upp elektriska kretsar I form av booleska ekvationer och sedan förenkla. På så vis förenklar man även kretsen och elektriska apparater kan då göras mindre och effektivare både när det kommer till funktion och kostnad. Ett enkelt exempel på hur ett kontaktnätsförenkling går till ser vi här:
Till att börja med har vi ab + a(a’ + c) =
= ab + aa’ + ac =
= ab + 0 + ac =
= ab + ac = a(b + c)



Logikalgebra
Förutom pionjärerna de Morgan och Boole, har även Bertrand Russell (1872-1970) bidragit mycket till den symboliska logiken. 1910-1913 utgav han det tre band stora verket Principia Mathematica, vilket senare kom att kallas ”den symboliska logikens bibel”. Logikalgebran översätter logiska resonemang från dess verbala form till en symbolisk form.

Ett påstående inom logiken är en utsaga som definitivt och otvetydigt antingen är sann (1) eller falsk (0), och motsvaras av mängdlärans grundmängd respektive tomma mängd. Låt p och q vara två olika påståenden. Det finns tre operationer i logikalgebran: negation p’ (ej p), konjunktion p ^ q (både p och q) samt disjunktion p v q (p eller q). Dessa är uppenbarligen isomorfa med de booleska ’, * och +.

Implikationer och argumentering
p -> q (om p, så q) är en implikation. Via jämförelse av sanningstabeller (tabulerande av alla möjliga värden och kombinationer av p och q) kan man förenkla implikationer till de ovanstående tre operationerna. p -> q = p’ v q.

p	q	p -> q	p’ v q
1	1	1	1
1	0	0	0
0	0	1	1
0	0	1	1

Det är viktigt att notera att implikationen inte är kommutativ. p -> q är inte lika med q -> p (konversen). I de fall p -> q = q -> p noteras detta med p <-> q (q om och endast om p).

Vid olika typer av argumentering används implikationer för att deducera slutsatser. Modus ponens är ett argumentationsschema som ser ut på följande vis:

Exempel	Med symboler
Jag har ätit mig mätt	p
Om jag har ätit mig mätt är jag sömnig	p -> q
Jag är sömnig	q

Syllogismlagen är ett annat argumentationsschema:

Exempel	Med symboler
Om jag tittar på tv så tittar jag på sport	a -> b
Om jag tittar på sport så äter jag chips	b -> c
Om jag tittar på tv så äter jag chips	a -> c

Man kan förstås konstruera argumentationsscheman med många fler påståenden och implikationer. Ju mer komplicerat argumentationsschemat blir desto mer underlättar det att med symboler komma fram till giltiga slutsatser.

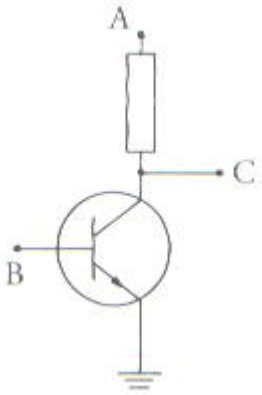
En annan typ av bevisföring är den indirekta bevisföringen. För att bevisa en implikation, p -> q med indirekt bevisföring antar vi först att implikationen är felaktig. Härifrån försöker man sedan hitta en motsägelse, vilket leder till att implikationen måste vara sann. Om man däremot vill visa att implikationen är felaktig antar man först att den är sann och försöker sedan hitta ett motexempel. Ett klassiskt exempel på indirekt bevisföring gavs av Euler (1707-1783). Påståendet lyder ”polynomet n^2 + n + 41” genererar primtal för alla icke-negative heltal n. Men n = 40 ger inget primtal, vilket är ett motexempel och därmed är påståendet falskt.

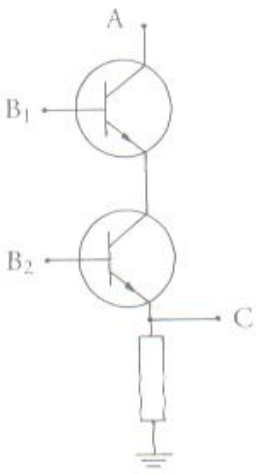
Tillämpning inom datateknik

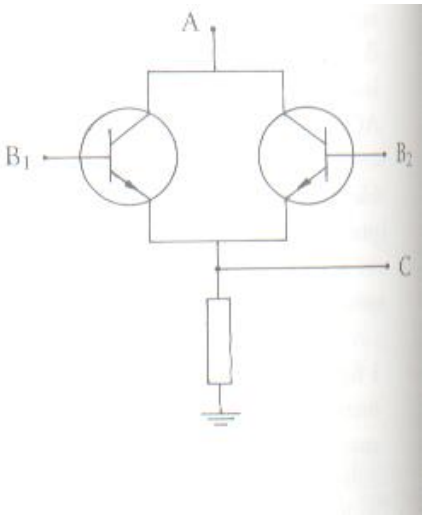
Logiska kretsar

Rent fysiskt implementeras logikalgebra med hjälp av s.k. grindar. De är konstruerade av dioder och transistorer (i vilka halvledare, t ex kisel, var en viktig komponent) för deras förmåga att snabbt släppa fram eller spärra ström. Rent principiellt fungerar grindarna på följande vis:

Kopplingsschemat nedan visar en NOT-grind (dvs. boole-logikens ’ operator). På punkten A vilar en utifrån pålagd potential vilket vanligtvis är 5 volt. B motsvarar den variabel som ska negeras. Om B underskrider ett kritiskt värde, t ex, 2 volt, fungerar transistorn som en oerhört stor resistans och stryper strömmen genom transistorn. Utgångspotentialen C antar då samma värde som A. Om B däremot överskrider det kritiska värdet fungerar transistorn som en ledare och utgångspotentialen I C blir då ca 0 volt. Med likartade argument fungerar AND (*) och OR (+) grindar. Egentligen kan alla dessa grindar härledas ur en enda grind, nämligen den s.k. NAND-grinden (AND negerat), med hjälp av de Morgans lagar. Vi håller oss dock till dessa tre grindar eftersom det gör det enklare att överblicka hur mer komplexa grindnät fungerar.







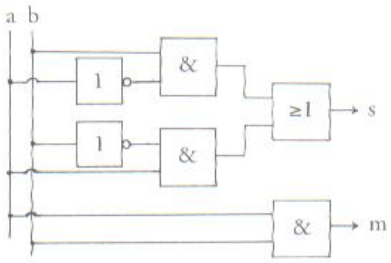
NOT-grind (1o) AND-grind (&) OR-grind (≥1)

Numerisk beräkning

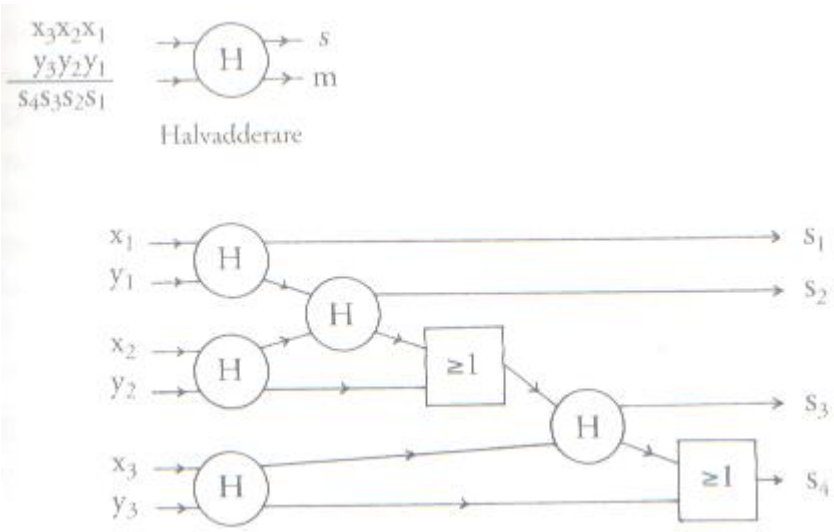
De första datorerna var bara gigantiska miniräknare, och i många språk, t ex engelska, betyder dator ”räknare”. Med hjälp av ovanstående grindar (förkortningssymboler inom parantes) kan man konstruera s.k. grindnät för att hantera aritmetik, vilket kan ses som en tämligen elementär del av logiken. Vad som verkar otroligt men inte desto mindre är sant är att all kalkyl, oavsett graden av komplexitet, kan härledas ur de fyra grundläggande additionerna 0 + 0, 0 + 1, 1 + 0 och 1 + 1. De fyra summorna är 0, 1, 1 respektive 2. I det binära talsystemet representeras 2 av 10 och vi får därmed en minnessiffra (m) och en andra entalssiffra (s). Om vi ställer upp en funktionstabell och jämför med Boole-funktionerna ab och a’b + ab’ får vi följande:

a	b	M(a,b)	s(a,b)	ab	a’b + ab’
0	0	0	0	0	0
0	1	0	1	0	1
1	0	0	1	0	1
1	1	1	0	1	0

Alltså, m = ab och s = a’b + ab’. Med hjälp av detta kan vi bygga en s.k. half adder (halvadderare) som summerar två ensiffriga tal:



Ett exempel på hur man kan kombinera fem half adders och två OR-grindar för att addera två tresiffriga tal $|x_3|x_2|x_1| + |y_3|y_2|y_1|$, där siffrorna 3, 2 och 1 representerar 4-tal, 2-tal respektive ental i det binära talsystemet, ges här:



På liknande vis kan grindnätet byggas ut så att addition med ett större antal siffror kan utföras.

Programmering

Genom operativsystemens implementering av kompilerad kod kan programmerare på ett mycket bekvämt manipulera datorns minne till att göra och köra program. Minnet består av bitar vilka kan vara antingen på (1) eller av (0). Härav ser vi att datorer, eftersom de är byggda av logiska kretsar, också är isomorfa med boolesk algebra på ren funktionell nivå. Men även programmeraren kan komma att behöva den booleska logiken då han eller hon programmerar. I programmeringsspråket C, t ex, så utgör logikoperatorerna en stor del av grunden till språket och är mycket viktigt att kunna då man exempelvis ska hantera s.k. flaggor. T ex, är det inte svårt att se att s.k. if-satser inom programmering är samma sak som implikationer i logiken.

Den formella logiken lever alltså kvar än idag, om inte så mycket hos matematiker, så åtminstone hos datateknikerna. Implicit, finns logiken förstås överallt i vardagen.

Källor

Fraleigh, John B. *Abstract Algebra.* Addison-Wesley, 2000.

Katz, Victor J. *A History of Mathematics*. Addison-Wesley, 1998.

Ulin, Bengt. *Boolesk Algebra.* Ekelunds förlag, 2001.

Wolfram, Stephen. ”Mathematical Notation: Past and Future”. < http://www.stephenwolfram.com/publications/talks/mathml/mathml2.html>, 2004-08-01