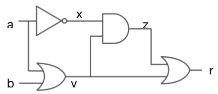
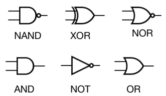
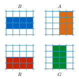




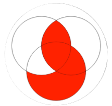


Informatics 1

Boolean Algebra

CNF KM Tseytin

Michael Fourman



$x \vee (y \vee z) = (x \vee y) \vee z$	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	associative
$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	distributive
$x \vee y = y \vee x$	$x \wedge y = y \wedge x$	commutative
$x \vee 0 = x$	$x \wedge 1 = x$	identity
$x \vee 1 = 1$	$x \wedge 0 = 0$	annihilation
$x \vee x = x$	$x \wedge x = x$	idempotent
$x \vee \neg x = 1$	$\neg x \wedge x = 0$	complements
$x \vee (x \wedge y) = x$	$x \wedge (x \vee y) = x$	absorbtion
$\neg(x \vee y) = \neg x \wedge \neg y$	$\neg(x \wedge y) = \neg x \vee \neg y$	de Morgan
$\neg\neg x = x$	$x \to y = \neg x \leftarrow \neg y$	

$\neg(a \to b) = a \wedge \neg b$ $\quad a \leftrightarrow b = (a \to b) \wedge (b \to a)$ $\quad a \to b = \neg a \vee b$

$\neg(a \vee b) = \neg a \wedge \neg b$ $\quad \neg(a \wedge b) = \neg a \vee \neg b$

$\neg 0 = 1$ $\quad \neg\neg a = a$ $\quad \neg 1 = 0$

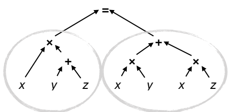
$a \vee 1 = 1$ $\quad a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ $\quad a \wedge 0 = 0$

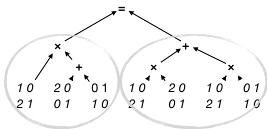
$a \vee 0 = a$ $\quad a \vee \neg a = 1$ $\quad a \wedge \neg a = 0$ $\quad a \wedge 1 = a$

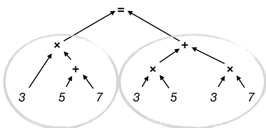
In algebra we make statements about numbers.

$x\,(\,y + z\,) = x\,y + x\,z$

is a statement





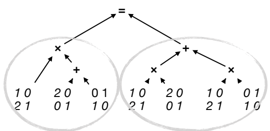


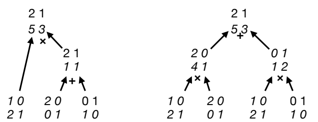
3 × (5 + 7) = 3 × 5 + 3 × 7

3 × 12 = 15 + 21

36 = 36

```
infixl 6 +
infixl 6 -
infixl 7 *
```



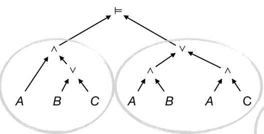


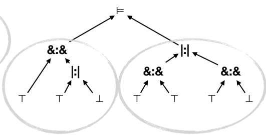
```haskell
-- The datatype 'Wff' a
data Wff a = V a
           | T
           | F
           | Not (Wff a)
           | Wff a :|: Wff a
           | Wff a :&: Wff a
           | Wff a :->: Wff a
           | Wff a :<->: Wff a
           deriving (Eq, Ord)
infixr 3 :&:
infixr 2 :|:
infixr 1 :->:
infixr 0 :<->:
```

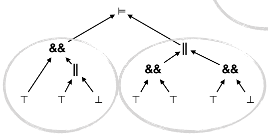
In Logic we make statements about predicates.

$A \wedge (B \vee C) \models (A \wedge B) \vee (A \wedge C)$

is a statement







```haskell
infixr 3 &&
infixr 2 ||
```

```haskell
substitute :: (a -> b) -> Wff a -> Wff b
substitute _ T = T
substitute _ F = F
substitute f (Not p)     =  Not (substitute f p)
substitute f (p :|: q)   =  substitute f p :|: substitute f q
substitute f (p :&: q)   =  substitute f p :&: substitute f q
substitute f (p :->: q)  =  substitute f p :->: substitute f q
substitute f (p :<->: q) =  substitute f p :<->: substitute f q
substitute f (V a)   = V (f a)

evaluate :: Wff Bool -> Bool
evaluate T = True
evaluate F = False
evaluate (Not p)     = not (evaluate p)
evaluate (p :&: q)   = evaluate p && evaluate q
evaluate (p :|: q)   = evaluate p || evaluate q
evaluate (p :->: q)  = evaluate p <= evaluate q
evaluate (p :<->: q) = evaluate p == evaluate q
evaluate (V b)  = b
```

```
substitute :: (a -> b) -> Wff a -> Wff b
substitute _ T = T
substitute _ F = F
substitute f (Not p)     =  Not (substitute f p)
substitute f (p :|: q)   =  substitute f p :|: substitute f q
substitute f (p :&: q)   =  substitute f p :&: substitute f q
substitute f (p :->: q)  =  substitute f p :->: substitute f q
substitute f (p :<->: q) =  substitute f p :<->: substitute f q
substitute f (V a)  = V (f a)
```

```
interpret :: Wff (Pred a) -> Pred a
interpret T = (\_ -> True)
interpret F = (\_ -> False)
interpret (Not p)      = neg (interpret p)
interpret (p :&: q)    = interpret p &:& interpret q
interpret (p :|: q)    = interpret p |:| interpret q
interpret (p :->: q)   = interpret p <:= interpret q
interpret (p :<->: q)  = interpret p =:= interpret q
interpret (V b)   = b
```

To produce conjunctive normal form (CNF)

eliminate $\leftrightarrow$ $\rightarrow$

push negations in

push $\vee$ inside $\wedge$

$\neg(a \to b) = a \wedge \neg b$ $\quad a \leftrightarrow b = (a \to b) \wedge (b \to a)$ $\quad a \to b = \neg a \vee b$

$\neg(a \vee b) = \neg a \wedge \neg b$ $\quad \neg(a \vee b) = \neg a \wedge \neg b$

$\neg 0 = 1$ $\quad \neg\neg a = a$ $\quad \neg 1 = 0$

$a \vee 1 = 1$ $\quad a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ $\quad a \wedge 0 = 0$

$a \vee 0 = a$ $\quad a \vee \neg a = 1$ $\quad a \wedge \neg a = 0$ $\quad a \wedge 1 = a$

---

an example

$\boxed{r \leftrightarrow a \wedge g}$

eliminate $\leftrightarrow$ $\rightarrow$

push negations in

push $\vee$ inside $\wedge$

$\neg(a \to b) = a \wedge \neg b$ $\quad a \leftrightarrow b = (a \to b) \wedge (b \to a)$ $\quad a \to b = \neg a \vee b$

$\neg(a \vee b) = \neg a \wedge \neg b$ $\quad \neg(a \vee b) = \neg a \wedge \neg b$

$\neg 0 = 1$ $\quad \neg\neg a = a$ $\quad \neg 1 = 0$

$a \vee 1 = 1$ $\quad a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ $\quad a \wedge 0 = 0$

$a \vee 0 = a$ $\quad a \vee \neg a = 1$ $\quad a \wedge \neg a = 0$ $\quad a \wedge 1 = a$

$\neg(a \to b) = a \land \neg b$ $\quad$ $a \leftrightarrow b = (a \to b) \land (b \to a)$ $\quad$ $a \to b = \neg a \lor b$

$$\boxed{r \leftrightarrow a \land g} \equiv \boxed{(r \to a \land g) \land (a \land g \to r)}$$

```
impElim :: Wff a -> Wff a
impElim (Not p)     = Not (impElim p)
impElim (p :|: q)   = impElim p          :|: impElim q
impElim (p :&: q)   = impElim p          :&: impElim q
impElim (p :->: q)  = Not (impElim p)    :|: impElim q
impElim (p :<->: q) = impElim (p :->: q) :&: impElim (q :->: p)
impElim  x          = x -- (V a), T, F
```

$\neg(a \to b) = a \land \neg b$ $\quad$ $a \leftrightarrow b = (a \to b) \land (b \to a)$ $\quad$ $a \to b = \neg a \lor b$

$$\begin{aligned} r \leftrightarrow a \land g &\equiv \boxed{(r \to a \land g)} \land \boxed{(a \land g \to r)} \\ &\equiv \boxed{(\neg r \lor (a \land g))} \land \boxed{(\neg(a \land g) \lor r)} \end{aligned}$$

$\neg(a \lor b) = \neg a \land \neg b$ $\quad$ $\neg(a \lor b) = \neg a \land \neg b$

$\neg 0 = 1$ $\quad$ $\neg\neg a = a$ $\quad$ $\neg 1 = 0$

$$\begin{aligned} r \leftrightarrow a \land g &\equiv (r \to a \land g) \land (a \land g \to r) \\ &\equiv (\neg r \lor (a \land g)) \land \boxed{(\neg(a \land g) \lor r)} \\ &\equiv (\neg r \lor (a \land g)) \land \boxed{(\neg a \lor \neg g \lor r)} \end{aligned}$$

```
toNNF :: Wff a -> Wff a
toNNF (Not T)         = F
toNNF (Not F)         = T
toNNF (Not (Not p))   = toNNF p
toNNF (Not (p :&: q)) = toNNF (Not p) :|: toNNF (Not q)
toNNF (Not (p :|: q)) = toNNF (Not p) :&: toNNF (Not q)
toNNF (p :&: q)       = toNNF p :&: toNNF q
toNNF (p :|: q)       = toNNF p :|: toNNF q
toNNF  p              = p -- (V a), (Not (V a)), T, F
```

$a \lor 1 = 1$ $\quad a \lor (b \land c) = (a \lor b) \land (a \lor c)$ $\quad a \land 0 = 0$

$a \lor 0 = a$ $\quad a \lor \neg a = 1$ $\quad a \land \neg a = 0$ $\quad a \land 1 = a$

$$
\begin{aligned}
r \leftrightarrow a \land g &\equiv (r \to a \land g) \land (a \land g \to r) \\
&\equiv (\neg r \lor (a \land g)) \land (\neg(a \land g) \lor r) \\
&\equiv \boxed{(\neg r \lor (a \land g))} \land (\neg a \lor \neg g \lor r) \\
&\equiv \boxed{(\neg r \lor a) \land (\neg r \lor g)} \land (\neg a \lor \neg g \lor r)
\end{aligned}
$$

```
toCNFList :: Eq a => Wff a -> [[Wff a]]
toCNFList p = cnf (toNNF p)
    where
      cnf F           =  [[]]
      cnf T           =  []
      cnf (V n)       =  [[V n]]
      cnf (Not (V n)) =  [[Not (V n)]]
      cnf (p :&: q)   =  nub (cnf p ++ cnf q)
      cnf (p :|: q)   =  [nub $ x ++ y | x <- cnf p, y <- cnf q]
```

$a \lor 1 = 1$ $\quad a \lor (b \land c) = (a \lor b) \land (a \lor c)$ $\quad a \land 0 = 0$

$a \lor 0 = a$ $\quad a \lor \neg a = 1$ $\quad a \land \neg a = 0$ $\quad a \land 1 = a$

$$
\begin{aligned}
r \leftrightarrow a \land g &\equiv (r \to a \land g) \land (a \land g \to r) \\
&\equiv (\neg r \lor (a \land g)) \land (\neg(a \land g) \lor r) \\
&\equiv \boxed{(\neg r \lor (a \land g))} \land (\neg a \lor \neg g \lor r) \\
&\equiv \boxed{(\neg r \lor a) \land (\neg r \lor g)} \land (\neg a \lor \neg g \lor r)
\end{aligned}
$$

```
toCNFList :: Eq a => Wff a -> [[Wff a]]
toCNFList p = cnf (toNNF p)
    where
      cnf F           =  [[]]
      cnf T           =  []
      cnf (V n)       =  [[V n]]
      cnf (Not (V n)) =  [[Not (V n)]]
      cnf (p :&: q)   =  nub (cnf p ++ cnf q)
      cnf (p :|: q)   =  [nub $ x ++ y | x <- cnf p, y <- cnf q]
```


```
listsToCNF :: [[Wff a]] -> Wff a
listsToCNF xss | null xss     = T
               | any null xss = F -- some xss null
               | otherwise    =
                   (foldl1 (:&:) . map (foldl1 (:|:))) xss
```

$$r \leftrightarrow a \land g \equiv (r \to a \land g) \land (a \land g \to r)$$

$\neg(a \to b) = a \land \neg b$ $\quad a \leftrightarrow b = (a \to b) \land (b \to a)$ $\quad a \to b = \neg a \lor b$

$$\equiv (\neg r \lor (a \land g)) \land (\neg(a \land g) \lor r)$$

$\neg(a \lor b) = \neg a \land \neg b$ $\quad \neg(a \lor b) = \neg a \land \neg b$

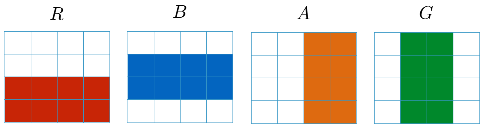
$\neg 0 = 1$ $\quad \neg\neg a = a$ $\quad \neg 1 = 0$

$$\equiv (\neg r \lor (a \land g)) \land (\neg a \lor \neg g \lor r)$$

$a \lor 1 = 1$ $\quad a \lor (b \land c) = (a \lor b) \land (a \lor c)$ $\quad a \land 0 = 0$

$a \lor 0 = a$ $\quad a \lor \neg a = 1$ $\quad a \land \neg a = 0$ $\quad a \land 1 = a$

$$\equiv (\neg r \lor a) \land (\neg r \lor g) \land (\neg a \lor \neg g \lor r)$$



RB \ AG	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

RB \ AG	00	01	11	10
00	1	1	1	1
01	1	1	0	0
11	0	0	1	1
10	0	0	0	0

RB \ AG	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

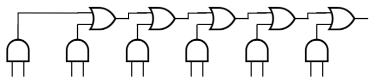
$$r \leftrightarrow (a \wedge b)$$

$$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$$

$$a \wedge b \vee c \wedge d \vee e \wedge f \vee g \wedge h \vee j \wedge k \vee m \wedge n$$

$$(a \wedge b) \vee (c \wedge d) \vee (e \wedge f) \vee (g \wedge h) \vee (j \wedge k) \vee (m \wedge n)$$

How many clauses in the CNF?



$$a \wedge b \vee c \wedge d \vee e \wedge f \vee g \wedge h \vee j \wedge k \vee m \wedge n$$

$$(a \wedge b) \vee (c \wedge d) \vee (e \wedge f) \vee (g \wedge h) \vee (j \wedge k) \vee (m \wedge n)$$

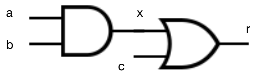
How many clauses in the CNF?

$2^8 = 64$

How many clauses to describe the circuit?

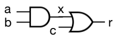
If we start from an expression then
we can draw an equivalent circuit with:
a wire for each subexpression,
a logic gate for each operator,
and an input for each variable.

$R = (A \wedge B) \vee G$



If we start from an expression then
we can draw an equivalent circuit with:
a wire for each subexpression,
a logic gate for each operator,
and an input for each variable.

$R = (A \wedge B) \vee G$



$r \leftrightarrow (a \wedge b)$

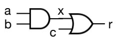
$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$

$r \leftrightarrow (a \vee b)$

$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$

If we start from an expression then
we can draw an equivalent circuit with:
a wire for each subexpression,
a logic gate for each operator,
and an input for each variable.

$R = (A \wedge B) \vee G$



$r \leftrightarrow (a \wedge b)$

$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$

$r \leftrightarrow (a \vee b)$

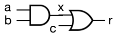
$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$

$x \leftrightarrow (a \wedge b)$

If we start from an expression then
we can draw an equivalent circuit with:

$R = (A \wedge B) \vee G$

a wire for each subexpression,
a logic gate for each operator,
and an input for each variable.



$r \leftrightarrow (a \wedge b)$
$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$

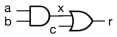
$r \leftrightarrow (a \vee b)$
$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$

$x \leftrightarrow (a \wedge b)$
$(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$

If we start from an expression then
we can draw an equivalent circuit with:

$R = (A \wedge B) \vee G$

a wire for each subexpression,
a logic gate for each operator,
and an input for each variable.



$r \leftrightarrow (a \wedge b)$
$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$

$r \leftrightarrow (a \vee b)$
$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$

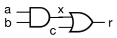
$x \leftrightarrow (a \wedge b)$
$(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$

$$r \leftrightarrow (x \vee c)$$

If we start from an expression then
we can draw an equivalent circuit with:

$R = (A \wedge B) \vee G$

a wire for each subexpression,
a logic gate for each operator,
and an input for each variable.



$r \leftrightarrow (a \wedge b)$
$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$

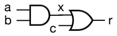
$r \leftrightarrow (a \vee b)$
$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$

$x \leftrightarrow (a \wedge b)$
$(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$

$r \leftrightarrow (x \vee c)$
$(\neg r \vee x \vee c) \wedge (r \vee \neg x) \wedge (r \vee \neg c)$

If we start from an expression then we can draw an equivalent circuit with:

$R = (A \wedge B) \vee G$



a wire for each subexpression, a logic gate for each operator, and an input for each variable.

$r \leftrightarrow (a \wedge b)$

$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$

$r \leftrightarrow (a \vee b)$

$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$

$x \leftrightarrow (a \wedge b)$

$(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b)$

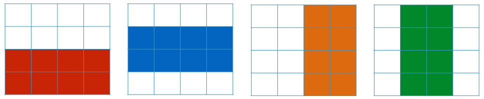
$r \leftrightarrow (x \vee c)$

$(\neg r \vee x \vee c) \wedge (r \vee \neg x) \wedge (r \vee \neg c)$

Combine the two CNF, with R = True

$$(x \vee \neg a \vee \neg b) \wedge (\neg x \vee a) \wedge (\neg x \vee b) \wedge (x \vee c)$$

R B A G



RB \ AG	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

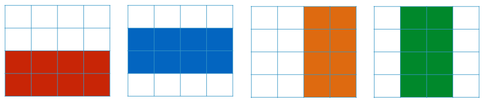
AG	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

AG	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	1
10	0	0	1	1

$$r \leftrightarrow (a \vee b)$$

$$(\neg r \vee a \vee b) \wedge (r \vee \neg a) \wedge (r \vee \neg b)$$

R B A G



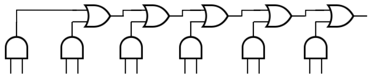
RB \ AG	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

AG	00	01	11	10
00	1	1	1	1
01	1	1	0	0
11	0	0	1	1
10	0	0	0	0

AG	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

$$r \leftrightarrow (a \wedge b)$$

$$(r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$$



$a \wedge b \vee c \wedge d \vee e \wedge f \vee g \wedge h \vee j \wedge k \vee m \wedge n$

$(a \wedge b) \vee (c \wedge d) \vee (e \wedge f) \vee (g \wedge h) \vee (j \wedge k) \vee (m \wedge n)$

How many clauses in the CNF?

$2^8 = 64$

How many clauses to describe the circuit?

11 x 3 = 33 (before simplification)