Logic roundup: SAT CNF Tseytin DPLL

In the first part of this tutorial you will combine the code from CL tutorial 6 and FP tutorial 6, and implement and test the Tseytin transformation.

The code in CL7.hs includes material from both earlier tutorials. In particular, we have the following type declarations for our two languages – `Wff`s and `Form`s.

```
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
```

```
data    Literal a = N a | P a
                    deriving (Ord, Eq, Show)

newtype Clause a = Or  [Literal a]
                    deriving (Ord, Eq, Show)

newtype   Form a = And [Clause a]
                    deriving (Eq, Show)
```

The code also includes a version of our dpll algorithm:

```
(<<) :: Ord a => [Clause a] -> Literal a -> [Clause a]
cs << x = [ Or (delete (neg x) ys) | Or ys <- cs, not $ x `elem` ys ]

dpll :: Eq a => Form a -> [[Literal a]]
dpll (And css) =
  let models [] = [[]]
      models cs = case prioritise cs of
        Or []          -> []   -- empty clause: no models
        Or [lit]       -> [    lit : m | m <- models (cs << lit)] -- unit clause
        Or (lit : _)   -> [    lit : m | m <- models (cs << lit)]
                          ++
                          [neg lit : m | m <- models (cs << neg lit)]
  in models css
  where prioritise = minimumBy (comparing (\(Or lits) -> length lits))
```

1. Your first task is to provide a function

   `wffToForm :: Eq a => Wff a -> Form a`

2. Then write a test to check that a Wff is satisfiable (use `satisfiable`) iff the corresponding form, produced by `wffToForm`, has a model (use `dpll`).

   `prop_form_equiv :: Wff Atom -> Bool`

3. The Tseytin procedure introduces a variable for each subformula – or implementation of the Tseytin transform has the following type `tseytinToForm :: Ord a => Wff a -> Form (Wff a)`

   For each compound subformula (one that includes a connective, call the connective $*$) the transform `tt` includes a CNF for the expression $r \leftrightarrow a(*)b$.
   The code given shows the pattern for `:&:`.
   $r \leftrightarrow a \wedge b \equiv (r \vee \neg a \vee \neg b) \wedge (\neg r \vee a) \wedge (\neg r \vee b)$
   We include this and then add `tt a ++ tt b`, the transforms of its subexpressions. .

   ```
   tt r@(a :&: b)    =
     [ Or[P r, N a, N b], Or[N r, P a]
     , Or[N r, P b]] ++ tt a ++ tt b

   tt r@(a :|: b)    = undefined
   tt r@(a :->:  b)  = undefined
   tt r@(a :<->: b)  = undefined
   ```

   Follow this pattern, and use CNFs you have derived earlier to replace each instance of undefined.

4. Now write a test `prop_tseytin_equiv :: Wff Atom -> Bool` to check that a Wff is satisfiable (using the function `satisfiable`) iff its Tseytin transform has a model (found using `dpll`).

5. To complete this part of the exercise, define a reasonable function `size :: Form a -> Int` and use `quickCheck` to find a Wff for which `size (wffToForm wff) > 100 * size (tseytinToForm wff)`.

1 Counting models

If we want to go beyond yes/no questions, it is natural to ask, *How many ...?* We are interested in sets, so we will ask how many elements there are in a set of states defined by a logial formula. So, the question is, *How many states satisfy a given formula?*

This exercise need not involve any Haskell. However, you may well find Haskell is a useful tool for checking your answers. Not that our Haskell type Atom is defined to be `data Atom = A|B|C|D|W|X|Y|Z` – to avoid clashes with other uses of `TFPN`, for example. You an use `String` or `Char` as your atoms.

6. This question concerns the 256 possible truth valuations of the following eight propositional letters A, B, C, D, E, F, G, H. For each of the following expressions, say how many of the 256 valuations satisfy the expression, and briefly explain your reasoning. For example, the expression D is satisfied by 128 of the 256 valuations, Since for each valuation that makes D true there is a matching valuation that make D false.

   (a) $A \vee B$ (b) $A \to B$ (b) $(A \to B) \wedge (C \to D)$ (d) $A \oplus B$

7. For the following questions, use the arrow rule. This is explained in the first video of these four. For more details see overleaf. If you still have problems understanding this let me know and I will post more video examples.

   (a) $(A \to B) \wedge (B \to C)$
   (b) $(A \to B) \wedge (B \to C) \wedge (C \to D)$
   (c) $(A \to B) \wedge (C \to B)$
   (d) $(A \to B) \wedge (B \to C) \wedge (D \to B)$
   (e) $(A \to B) \wedge (A \to C)$
   (f) $(A \to B) \wedge (B \to C) \wedge (D \to C)$
   (g) $(A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (A \to E) \wedge (E \to D)$
   (h) $(\neg A \to B) \wedge (B \to \neg C) \wedge (C \to D) \wedge (A \to \neg E) \wedge (E \to D)$
   (i) $(A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (A \to E) \wedge (E \to D) \wedge (D \to F)$
   (j) $(A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (A \to \neg B) \wedge (\neg B \to D) \wedge (F \to A)$
   (k) $(A \to B) \wedge (\neg B \to C) \wedge (C \to D) \wedge (\neg A \to E) \wedge (E \to D) \wedge (F \to A)$
   (l) $(A \to B) \wedge (B \to A) \wedge (C \to D) \wedge (D \to E) \wedge (E \to F) \wedge (F \to G) \wedge (G \to H)$
   (m) $(A \to B) \wedge (B \to A) \wedge (C \to D) \wedge (D \to C) \wedge (E \to F) \wedge (F \to G) \wedge (G \to H)$
   (n) $(H \to A) \wedge (A \to B \wedge C) \wedge (B \vee C \to D) \wedge (A \to E) \wedge (E \to F) \wedge (F \to G) \wedge (G \to H)$

   You will find more examples in past papers for the past few years.

The arrow rule

The arrow rule applies to 2-SAT problems. It lets us identify and count the satisfying valuations for a 2-SAT problem. We begin by converting each clause with two literals to two implications, both equivalent to the clause – each implication is the contrapositive of the other.

In mathematical terms, the arrows generate a preorder: if there is an arrow $A \to B$ then $A \leq B$.

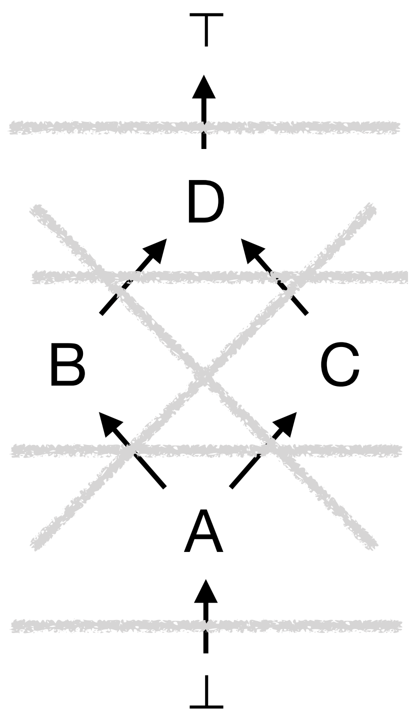
To apply the arrow rule, make a diagram with a directed graph whose nodes are all the literals, together with $\top$ and $\bot$, and whose edges are arrows representing the implications we have identified, with additional arrows from $\bot$ to each minimal node and from each maximal node to $\top$.

A *legal cut* of this diagram is a line that must separate $\top$ from $\bot$, and each literal from its negation, and such that each arrow that crosses the line goes from the side of $\bot$ (below) to the side of $\top$ (above).

The idea is that a valuation V :: Atom$-$ $>$ Bool satisfies every one of the implications iff it preserves the ordering: if $A \to B$ then $VA \leq VB$.

For example,

$$(A \to B) \wedge (B \to D) \wedge (A \to C) \wedge (C \to D)$$



gives the diagram shown here. Any legal cut corresponds to a valuation, making the literals above the line (on the same side as $\top$) true, and those below the line (on the same side as $\bot$) false. These are exactly the valuations that satisfy all of the original clauses.

We could also draw the opposite graph, with negated atoms and reversed arrows. We would arrive a t the same valuations of the atoms. In this example, the positive and negative occurrences of each atom lie in separate components of the graph, and it suffices to consider one component. In some examples the graph does not split in this way and we have to consider the entire graph, with each arrow and its contrapositive.

Here, there are six satisfying valuations corresponding to the six grey lines shown; these are the legal cuts for this diagram.

This tutorial exercise sheet was written by Michael Fourman. please send comments to `michael@ed.ac.uk`

1. Say how many of the 256 valuations satisfy the expression.

(a) $A \vee B$ (b) $A \to B$ (b) $(A \to B) \wedge (C \to D)$ (d) $A \oplus B$

1(a) If we only consider the truth values of A and B, there are 3 valuations that satisfy $A \vee B$ i.e. $\{A = \top, B = \top\}, \{A = \top, B = \bot\}, \{A = \bot, B = \top\}$. The truth values aren't constraint so they can be either true or false, thus the final answer is $3 \times 2^6 = 192$

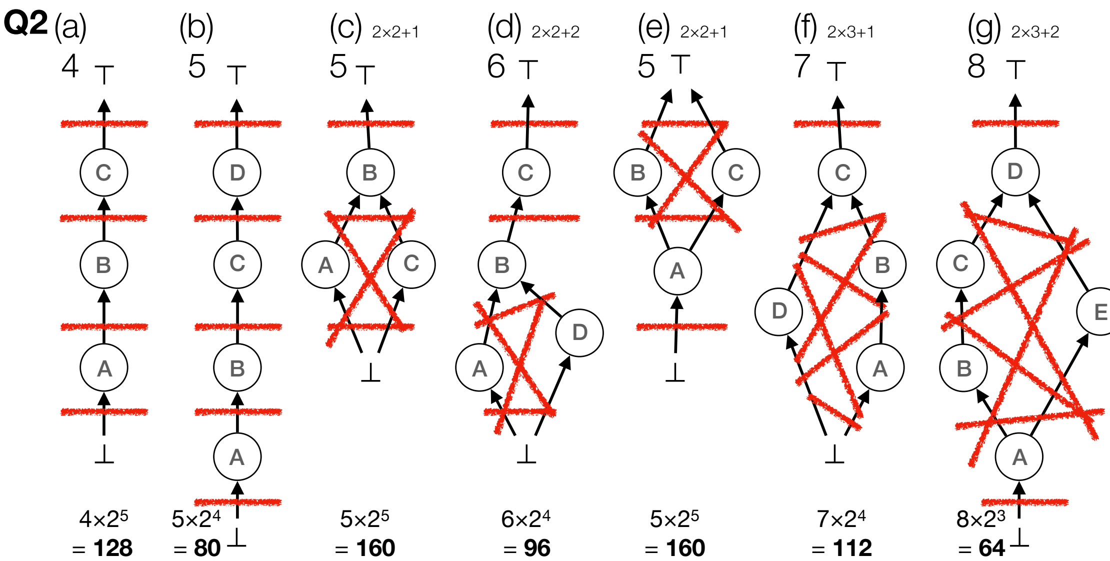
1(b) Again 3 valuations to make $A \to B$ true (when we consider only A and B) with 6 unconstrained variables. $3 \times 2^6 = 192$ satisfying valuations in total

1(c) 3 valuations to satisfy $A \to B$ and 3 valuations to satisfy $C \to D$ with 4 unconstrained variables, $3 \times 3 \times 2^4 = 144$ satisfying valuations.

1(d) 2 valuations to satisfy $A \oplus B$, so $2 \times 2^6 = 128$ satisfying valuations.

2. Use the arrow rule to say how many of the 256 valuations satisfy the expression.
   (a) $(A \to B) \wedge (B \to C)$
   (b) $(A \to B) \wedge (B \to C) \wedge (C \to D)$
   (c) $(A \to B) \wedge (C \to B)$
   (d) $(A \to B) \wedge (B \to C) \wedge (D \to B)$
   (e) $(A \to B) \wedge (A \to C)$
   (f) $(A \to B) \wedge (B \to C) \wedge (D \to C)$
   (g) $(A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (A \to E) \wedge (E \to D)$

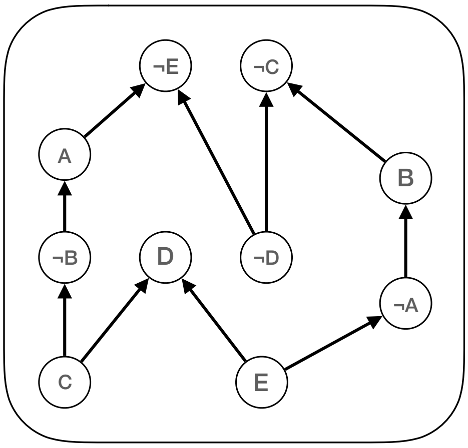
2(a)-2(g)



(h) $(\neg A \to B) \wedge (B \to \neg C) \wedge (C \to D) \wedge (A \to \neg E) \wedge (E \to D)$

2(h)

When we have mixtures of positive and negative literals, the arrow diagram may be hard to decipher. We have to add the contrapositive of each arrow and draw the graph. If the graph separates into two independent components with each atom occurring positive in one component and negative in the other, then we can just solve either componnet as before (the other component will be its dual, mirror-image). Otherwise we have to consider the entire graph and look for lines that separate each literal from its negation.



The diagram for this example is challenging.

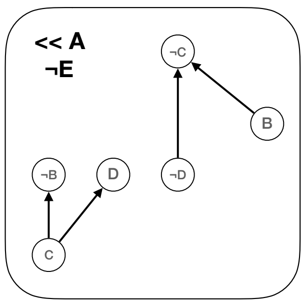
If the diagrams get too complicated we can simplify any of these problems. Suppose A is one of the letters, then the total number of satisfying valuations N must be the sum of the number of satisfying valuations that make A true, and the number of satisfying valuations that make A false. In each case (making A true or false) putting in the value (true or false) for A will give a simpler problem.

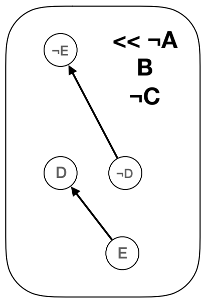
For example, in (2h)

$$(\neg A \to B) \wedge (B \to \neg C) \wedge (C \to D) \wedge (A \to \neg E) \wedge (E \to D)$$

we will divide the problem into two simpler cases.

If A is true then $\neg A \to B$ reduces to true, and the $A \to \neg E$ reduces to $\neg E$. Making A and $\neg E$ true, reduces our graph to two complementary components. We can use either, for example, $(B \to \neg C) \wedge (C \to D)$. This is a familiar pattern with 5 satisfying valuations. If we now set $\neg A$ true then B and $\neg C$ follow. Again we have to disjoint components, this time with three satisfying valuations.





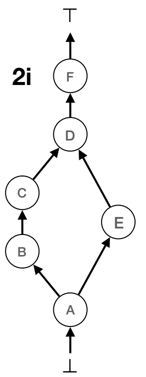
So, we have 5 satisfying valuations of B, C, D, with A and $\neg E$, and three satisfying valuations of D, E, with $\neg A$ and B, and $\neg C$. This gives a total of $5 + 3 = 8$ valuations of the five letters, A, B, C, D, E, so there are 64 valuations of our eight letters.

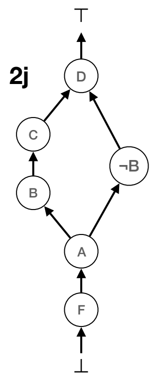
(i) $(A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (A \to E) \wedge (E \to D) \wedge (D \to F)$

(j) $(A \to B) \wedge (B \to C) \wedge (C \to D) \wedge (A \to \neg B) \wedge (\neg B \to D) \wedge (F \to A)$

2(i)-2(j)

Questions (2i) and (2j) are similar to (f) above – but with room for extra lines above and below. In the first case (2i), we have $2 \times 3 + 3 = 9$ valuations for the six letters giving $9 \times 4 = 36$ satisfying valuations.
Question (2j) has a sting in the tail. The graph includes nodes for both B and $\neg B$; any legal valuation must separate these two – making one true and the other false. There are only 3 legal valuations for the five atoms mentioned, so $3 \times 2^3 = 24$ valuation of the eight variables.





(k) $(A \to B) \wedge (\neg B \to C) \wedge (C \to D) \wedge (\neg A \to E) \wedge (E \to D) \wedge (F \to A)$

2(k) Our original wff is equivalent to:

$$(\neg B \to \neg A) \wedge (\neg B \to C) \wedge (C \to D) \wedge (\neg A \to E) \wedge (E \to D) \wedge (\neg A \to \neg F)$$

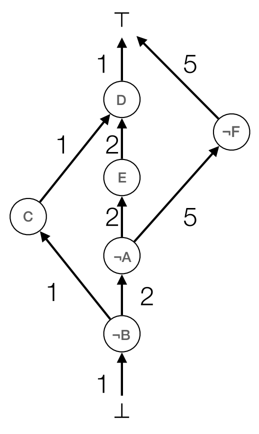
To derive this we have used contraposition to convert the original to a problem where each atom occurs only positively (C, D, E) or only negatively $(\neg A, \neg B, \neg F)$.

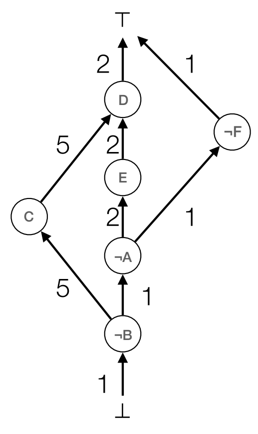
We could simply be lucky, and notice that this transformation is possible. However, you can manage, even without luck. Just construct the graph to include every arrow and its contrapositive. If the simplification is possible the graph will separate into two components, dual to each other. In that case you can choose to solve either component.

Instead of drawing lines (there are too many to keep track of them all), we count the possibilities. We give the same diagram twice, with different annotations. Looking at the same construction from two different sides.

In the left-hand diagram we label each arrow with the numberof paths that obey the arrow rule cut through that arrow, coming from the left of the diagram. The numbers on the arrows to the right of a region are each the sum of the numbers on the arrows to the left of that region. Our answer is the sum of the numbers down the right-hand edge of the diagram.
The right-hand diagram does the same, but working from the right. Here we take the sum of the numbers along the left-hand edge as the answer.

In each case we get the same answer, 13 (what a surprise)! This is for six letters so the final answer is $13 \times 2^2 = 52$.





(l) $(A \to B) \wedge (B \to A) \wedge (C \to D) \wedge (D \to E) \wedge (E \to F) \wedge (F \to G) \wedge (G \to H)$

2(l)
We can use the arrow rule to solve this:
A and B form an independent cycle – they must both take the same value, either true or false. The remaining six letters form a chain, with 7 satisfying valauations. In total we have $2 \times 7 = 14$ satisfying valuations.

(m) $(A \to B) \wedge (B \to A) \wedge (C \to D) \wedge (D \to E) \wedge (E \to F) \wedge (F \to G) \wedge (G \to H)$

2(m)
Here we have two independent cycles, A, B, and C, D, and a chain of four letters, E, F, G, H . thus we have $2 \times 2 \times 5 = 20$ valuations in total

---

(n) $(H \to A) \wedge (A \to B \wedge C) \wedge (B \vee C \to D) \wedge (A \to E) \wedge (E \to F) \wedge (F \to G) \wedge (G \to H)$

2(n)

Note that

$$A \to B \wedge C \equiv (A \to B) \wedge (A \to C)$$
$$(B \vee C \to D) \equiv (B \to D) \wedge (C \to D)$$

So we have the following implications

$$\begin{aligned} &(H \to A) \wedge (A \to B) \wedge (A \to C) \\ &\quad \wedge (B \to D) \wedge (C \to D) \wedge (A \to E) \\ &\quad \wedge (E \to F) \wedge (F \to G) \wedge (G \to H) \end{aligned} \quad (1)$$

These give the graph shown, which has 6 satisfying valuations.