Informatics 1 - CL: Tutorial 5

Counting

Week 7: 29 October – 2 November 2018[1]

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 to some of the smaller examples.

1. 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) $A \to B$ (c) $(A \to B) \wedge (C \to D)$ (d) $A \oplus B$

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

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

   (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.

[1] Version 1.3, 23rd October 2018

Tutorial Activities

1. As usual, buddy-up and check through your anwers to the homework exercises.

   Ask others in your group, or call on one of the tutors if you have unresolved questions.

More activities to be added ...

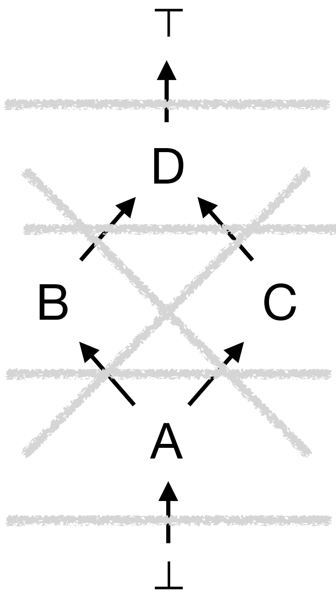
The arrow rule

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 – one implications is the contrapositive of the other.

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).

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.

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

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