School of Computing and Information Systems
COMP30026 Models of Computation Tutorial Week 5

21–25 August 2017

Plan

We have carried Question 27 (on Boolean modelling) over from last week. Questions 28 and 29 are about translating English statements into propositional logic. Questions 30 and 31 are entirely optional; just do them if you found XNF (from Lecture 6) intriguing. Question 32 introduces Wang's algorithm for validity checking. It lends itself well to a Haskell implementation, so Question 33 asks you to explore that on Grok. Questions 34–36 deal with first-order predicate logic.

The exercises

27. Recall the graph colouring problem from Week 3's Grok worksheet. How can we encode the three-colouring problem in propositional logic, in CNF to be precise? (One reason we might want to do so is that we can then make use of a so-called SAT solver to determine colourability.) Using propositional variables

    - B_i to mean node i is blue,
    - G_i to mean node i is green,
    - R_i to mean node i is red;
    - E_{ij} to mean i and j are different but connected by an edge,

    write formulas in CNF for these statements:

    (a) Every node (0 to n inclusive) is coloured.
    (b) Every node has at most one colour.
    (c) No two connected nodes have the same colour.

    For a graph with $n + 1$ nodes, what is the size of the CNF formula?

28. Consider these assumptions:

    (a) If Ann can clear 2 meters, she will be selected.
    (b) If Ann trains hard then, if she gets the flu, the selectors will be sympathetic.
    (c) If Ann trains hard and does not get the flu, she can clear 2 meters.
    (d) If the selectors are sympathetic, Ann will be selected.

    Does it follow that Ann will be selected? Does she get selected if she trains hard? Use any of the propositional logic techniques we have discussed, to answer these questions.

29. Consider the following four statements:

    (a) The commissioner cannot attend the function unless he resigns and apologises.
    (b) The commissioner can attend the function if he resigns and apologises.
    (c) The commissioner can attend the function if he resigns.
    (d) The commissioner can attend the function only if he apologises.

    Identify the basic propositions involved and discuss how to translate the statements into propositional logic. In particular, what is the translation of a statement of the form "X does not happen unless Y happens"? Identify cases where one of the statements implies some other statement in the list.

30. (Optional.) In Lecture 6 we saw that conjunctive normal form (CNF) and disjunctive normal form (DNF) are not *canonical*. For example, $(\neg P \vee \neg Q \vee \neg R) \wedge (P \vee Q \vee R) \wedge (P \vee \neg R) \wedge (\neg Q \vee R)$ and the logically equivalent $P \wedge \neg Q$ are both in reduced CNF. It would be nice if equivalent formulas were syntactically identical. For one thing, equivalence checking would then become much simpler, just checking that the given formulas are the same.

    It turns out that we can achieve this if we use exclusive or ($\oplus$) instead of or ($\vee$). XNF (exclusive-or normal form) can express every possible Boolean function. In XNF, a Boolean function is expressed as a sum of products, with addition corresponding to exclusive or, $\oplus$, and multiplication corresponding to conjunction, $\wedge$. For example, instead of $P \wedge (Q \Leftrightarrow R)$ we write $P \oplus (P \wedge Q) \oplus (P \wedge R)$. Actually we usually abbreviate this to $P \oplus PQ \oplus PR$, with the understanding that a "product" PQ is a conjunction. We call an expression like this—one that is written as a sum of products—a *polynomial*. Each summand (like PQ) is a *monomial*.

    The connective $\neg$ is not used in XNF, only $\oplus$ and $\wedge$. To compensate, one of the truth value constants, $\mathbf{t}$, is needed. If φ is a formula in XNF then $\varphi \oplus \mathbf{t}$ is its negation. The other constant, $\mathbf{f}$, is not needed. This is because $\mathbf{f}$ is like zero: It is a "neutral element" for $\oplus$ (that is, $\varphi \oplus \mathbf{f}$ is just φ) and it is an "annihilator" for $\wedge$ ($\varphi \wedge \mathbf{f}$ is just $\mathbf{f}$). So we can't have $\mathbf{f}$ *in* a monomial, because the entire monomial disappears if $\mathbf{f}$ enters it. And we don't need $\mathbf{f}$ *as* a monomial, because "adding" $\mathbf{f}$ really adds nothing.

    In this "Boolean ring" algebra, we are really dealing with arithmetic modulo 2, with $\wedge$ playing the role of multiplication, and $\oplus$ playing to role of addition. Given this, express the following in XNF:

    (a) $\neg P$ (c) $P \wedge \neg Q$ (e) $P \vee Q$

    (b) $P \wedge Q$ (d) $P \Leftrightarrow Q$ (f) $P \vee (Q \wedge R)$

31. (Optional.) Note that $\wedge$ distributes over $\oplus$, but not the other way round. Also note carefully "cancelling out" properties. If we conjoin the monomials PQR and $PRST$, we get $PQRST$; that is, "duplicates disappear". However, if we have $\varphi = \mu_1 \oplus \mu_2 \oplus \mu_3$ and $\varphi' = \mu_2 \oplus \mu_3 \oplus \mu_4$ and we want to find $\varphi \oplus \varphi'$, then we find that "duplicates cancel out pairwise": $\varphi \oplus \varphi' = \mu_1 \oplus \mu_4$. This happens because $\varphi \oplus \varphi = \mathbf{f}$ for all Boolean functions φ.

    Given this, turn the following into XNF:

    (a) $\neg(P \oplus Q)$ (c) $(PQ \oplus PQR \oplus R) \wedge (P \oplus Q)$ (e) $Q \vee (P \oplus PQ)$

    (b) $(P \oplus Q) \wedge R$ (d) $Q \wedge (P \oplus PQ \oplus \mathbf{t})$

32. One approach ("Wang's Algorithm") to validity checking for propositional logic manipulates so-called *sequents*. A sequent is a pair of (possibly empty) sets of propositional formulas (we will use lists for sets here). It is usually written

$$F_1, F_2, \ldots, F_n \vdash G_1, G_2, \ldots, G_m \qquad (n \geq 0, m \geq 0)$$

    where F_i and G_j are propositional formulas. You may think of this as the conditional

$$(F_1 \wedge F_2 \wedge \ldots \wedge F_n) \Rightarrow (G_1 \vee G_2 \vee \ldots \vee G_m)$$

    So an empty left-hand side corresponds to true, while an empty right-hand side corresponds to false. (Explain why that is the natural choice.)

    A sequent is *fully reduced* iff every F_i and every G_j is a propositional letter. A fully reduced sequent is *tautological* (or valid) exactly when there is a pair (i, j) such that $F_i = G_j$, that is, some propositional letter occurs on both sides of $\vdash$.

    The method consists of manipulating *sets* (or lists) of sequents, starting with $\{ \vdash F\}$, where F is the input formula. Since F is tautological exactly when the sequent $\vdash F$ is tautological, every terminating procedure that "reduces" sequents in a sound way will be a decision procedure.

The following table shows how sequents can be reduced. In the table, F and F' stand for formulas, whereas S and S' stand for (possibly empty) sequences of formulas. We use Haskell-style "cons" notation: $F : S$ is the sequence of formulas that has F as its first element (head) and S as its tail.

Given sequent	Derived sequent(s)
$S \vdash (F \vee F') : S'$	$S \vdash F : F' : S'$
$S \vdash (F \wedge F') : S'$	$S \vdash F : S'$ $S \vdash F' : S'$
$S \vdash \neg F : S'$	$F : S \vdash S'$
$(F \wedge F') : S \vdash S'$	$F : F' : S \vdash S'$
$(F \vee F') : S \vdash S'$	$F : S \vdash S'$ $F' : S \vdash S'$
$\neg F : S \vdash S'$	$S \vdash F : S'$

Our decision procedure applies these rules until a set of fully reduced sequents have been obtained. These are all tautological iff the original formula was.

(a) Use the decision procedure to show that the disjunction $\neg(\neg P \wedge Q) \vee (\neg P \vee R)$ is a tautology.

(b) Extend the table so that it includes rules for dealing with "$\Rightarrow$," "$\Leftrightarrow$,"and "$\oplus$" directly.

33. On Grok you will find a Week 5 Worksheet which asks you to explore a Haskell implementation of Wang's algorithm.

34. For each of the following predicate logic formulas, give an interpretation that makes the formula true, and one that makes it false:

(a) $\forall x \forall y(P(x, y))$

(b) $\forall x \exists y(P(x, y) \wedge P(y, x))$

(c) $(\forall x \exists y P(x, y)) \wedge (\forall x \exists y P(y, x))$

35. Show that $\forall x(P(x)) \models \exists y(P(y))$ holds. Does $\exists x(P(x)) \models \forall y(P(y))$ also hold? Recall that part of our definition of *interpretation* is that the *domain* is non-empty.

36. Turn the closed formula $\forall x \forall y \exists z \Big(P(x) \Rightarrow \forall y \forall z(Q(y, z)) \Big)$ into a simpler, equivalent formula of form $\varphi \Rightarrow \psi$.