Computational Thinking

Logic

Barnaby Martin

Department of Computing Science

Lecture 3

More on Propositional Logic

Durham University

Distribution Laws

- Whereas De Morgan's Laws allow us to simplify formulae with respect to negations
  - we often have "combinations" of disjunctions and conjunctions.

- The Distributive Law of Disjunction over Conjunction is

$a+(b\cdot c)$

$(a+b)(a+c)$

$$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$$ (and similarly $(q \wedge r) \vee p \equiv (q \vee p) \wedge (r \vee p)$)

and the Distributive Law of Conjunction over Disjunction is

$a\cdot(b+c)$

$$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$$ (and similarly $(q \vee r) \wedge p \equiv (q \wedge p) \vee (r \wedge p)$).

$ab+ac$

- Just as before, there are the generalised Distributive Laws

$$X \wedge (Y_1 \vee Y_2 \vee \ldots \vee Y_n) \equiv (X \wedge Y_1) \vee (X \wedge Y_2) \vee \ldots \vee (X \wedge Y_n)$$

$$X \vee (Y_1 \wedge Y_2 \wedge \ldots \wedge Y_n) \equiv (X \vee Y_1) \wedge (X \vee Y_2) \wedge \ldots \wedge (X \vee Y_n).$$

- Of course
  - we can apply these laws to combinations of formulae and to sub-formulae
    - not just with propositional variables.

Functional completeness

Durham University

- We defined propositional logic using the connectives $\{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$
  - but we could have chosen other connectives.
- We say that a set $\mathcal{C}$ of logical connectives is functionally complete if any propositional formula is
  - equivalent to one constructed using *only* the connectives from $\mathcal{C}$.
- In fact, $\{\wedge, \vee, \neg\}$ is functionally complete.

$(\neg p \Rightarrow q) \Leftrightarrow (\neg q \Rightarrow p)$

  - Let φ be a propositional formula involving the variables $p_1, p_2, \ldots, p_n$.
  - Build the truth table for φ and let f be some truth assignment (i.e., row) that evaluates to *true*.

p_1	p_2	$\ldots$	p_n	φ
		$\ldots$		
T	F		F	T
		$\ldots$		

  - Suppose that in this truth assignment f
    - each p_i has the truth value v_i.
  - Build a conjunction χ_f of literals as follows: for each i
    - if v_i is *true* then include the literal p_i in the conjunction χ_f
    - if v_i is *false* then include the literal $\neg p_i$ in the conjunction χ_f.

Example

- Consider the following truth table for φ

p	q	r	s	φ	
T	T	T	T	F	
T	T	T	F	F	
T	T	F	T	T	$\leftarrow f_1$
T	T	F	F	F	
T	F	T	T	F	
T	F	T	F	F	
T	F	F	T	T	$\leftarrow f_2$
T	F	F	F	T	$\leftarrow f_3$

p	q	r	s	φ	
F	T	T	T	F	
F	T	T	F	F	
F	T	F	T	F	
F	T	F	F	T	$\leftarrow f_4$
F	F	T	T	F	
F	F	T	F	F	
F	F	F	T	F	
F	F	F	F	T	$\leftarrow f_5$

- So

$\chi_{f_1} = p \wedge q \wedge \neg r \wedge s$ $\quad \chi_{f_2} = p \wedge \neg q \wedge \neg r \wedge s$ $\quad \chi_{f_3} = p \wedge \neg q \wedge \neg r \wedge \neg s$

$\chi_{f_4} = \neg p \wedge q \wedge \neg r \wedge \neg s$ = True $\quad \chi_{f_5} = \neg p \wedge \neg q \wedge \neg r \wedge \neg s$

and f_1: ? $T \wedge T \wedge \neg\neg F \wedge T$ $\quad f_2 : T \wedge \neg T \wedge \neg\neg F \wedge T$ = False

$$\psi = (p \wedge q \wedge \neg r \wedge s) \vee (p \wedge \neg q \wedge \neg r \wedge s) \vee (p \wedge \neg q \wedge \neg r \wedge \neg s)$$
$$\vee (\neg p \wedge q \wedge \neg r \wedge \neg s) \vee (\neg p \wedge \neg q \wedge \neg r \wedge \neg s)$$

ψ_{f_1}: True ∨ False ∨ False ∨ False ∨ False = True

Functional completeness

- We defined propositional logic using the connectives $\{\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$
  - but we could have chosen other connectives.
- We say that a set $\mathcal{C}$ of logical connectives is functionally complete if any propositional formula is
  - equivalent to one constructed using *only* the connectives from $\mathcal{C}$.
- In fact, $\{\wedge, \vee, \neg\}$ is functionally complete.

  - Let φ be a propositional formula involving the variables $p_1, p_2, \ldots, p_n$.
  - Build the truth table for φ and let f be some truth assignment (i.e., row) that evaluates to *true*.

p_1	p_2	$\ldots$	p_n	φ
		$\ldots$		
T	F		F	T
		$\ldots$		

  - Suppose that in this truth assignment f
    - each p_i has the truth value v_i.
  - Build a conjunction χ_f of literals as follows: for each i
    - if v_i is *true* then include the literal p_i in the conjunction χ_f
    - if v_i is *false* then include the literal $\neg p_i$ in the conjunction χ_f.

Functional completeness

- Now let ψ be the disjunction of all those conjunctions χ_f we have just built
  - remember, we only build disjunctions corresponding to the rows of the truth table evaluating to *true*.

- We claim that φ and ψ are logically equivalent.
  - Suppose that f is some truth assignment making φ *true*
    - so, we have indeed built the conjunction χ_f.
  - Key point
    - the only truth assignment making the conjunction χ_f *true* is the truth assignment f itself.
  - In particular, the truth assignment f must make χ_f true
    - e.g., with regard to the truth assignment f in the example, χ_f is

      $$p_1 \wedge \neg p_2 \wedge \ldots \wedge \neg p_n$$

      which is made *true* *only* by the truth assignment f.
  - Hence, f makes ψ *true*.

Functional completeness

- Conversely
  - suppose that *g* is some truth assignment making ψ *true*
    - so, at least one conjunct, χ_f say, is made *true* by *g*
  - but the only truth assignment making χ_f *true* is *f*
    - hence, $f = g$
  - the reason χ_f appears as a conjunct is because *f* makes φ *true*
    - so, $g = f$ is a truth assignment making φ *true*.

- Consequently, for any truth assignment *f*
  - *f* satisfies φ if, and only if, *f* satisfies ψ
    - that is, $\varphi \equiv \psi$.

- Our proof yields even more
  - every formula of propositional logic is equivalent to a formula in disjunctive normal form (d.n.f.)
    - a disjunction of conjunctions of literals
  - also, every truth table is the truth table of some propositional formula.

Conjunctive normal form

- Let φ be some formula of propositional logic.
- The formula $\neg\varphi$ is equivalent to one in disjunctive normal form
  - that is, one of the form

    $$\chi_1 \vee \chi_2 \vee \dots \vee \chi_m$$

    where each χ_i is a conjunction of literals.
- So, φ is equivalent to the formula

  $$\neg(\chi_1 \vee \chi_2 \vee \dots \vee \chi_m)$$

  which in turn, by using generalised De Morgan's Laws, is equivalent to

  $$\neg\chi_1 \wedge \neg\chi_2 \wedge \dots \wedge \neg\chi_m.$$
- Each $\neg\chi_i$ is equivalent to a disjunction of literals
  - by again using generalised De Morgan's Laws.
- Thus
  - every formula of propositional logic is logically equivalent to a conjunction of disjunctions of literals, i.e., a conjunction of clauses
    - that is, every formula of propositional logic is equivalent to a formula in conjunctive normal form (c.n.f.).

A spot of practice

- We wish to convert the formula $\varphi = ((\neg p \wedge q) \vee r) \wedge \neg((r \wedge p) \vee \neg q)$ into c.n.f.

p	q	r	$((\neg$	p	$\wedge$	$q)$	$\vee$	$r)$	$\wedge$	$\neg(($	r	$\wedge$	$p)$	$\vee$	$\neg$	$q)$	$\neg\varphi$
T	T	T	F	T	F	T	T	T	**F**	F	T	T	T	T	F	T	**T**
T	T	F	F	T	F	T	F	F	**F**	T	F	F	T	F	F	T	**T**
T	F	T	F	T	F	F	T	T	**F**	F	T	T	T	T	T	F	**T**
T	F	F	F	T	F	F	F	F	**F**	F	F	F	T	T	T	F	**T**
F	T	T	T	F	T	T	T	T	**T**	T	T	F	F	F	F	T	**F**
F	T	F	T	F	T	T	T	F	**T**	T	F	F	F	F	F	T	**F**
F	F	T	T	F	F	F	T	T	**F**	F	T	F	F	T	T	F	**T**
F	F	F	T	F	F	F	F	F	**F**	F	F	F	F	T	T	F	**T**

- So, $\neg\varphi$ is equivalent to

$$(p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r)$$
$$\vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r).$$

- Hence, φ is equivalent to the c.n.f. formula

$$(\neg p \vee \neg q \vee \neg r) \wedge (\neg p \vee \neg q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee q \vee r)$$
$$\wedge (p \vee q \vee \neg r) \wedge (p \vee q \vee r).$$

Converting to c.n.f. syntactically

- We can often establish normal forms "syntactically".
- Consider the formula

$$
\begin{aligned}
\varphi \quad & ((\neg p \wedge q) \vee r) \wedge \neg((r \wedge p) \vee \neg q) \\
& \equiv ((\neg p \vee r) \wedge (q \vee r)) \wedge (\neg(r \wedge p) \wedge q) \\
& \equiv (\neg p \vee r) \wedge (q \vee r) \wedge ((\neg r \vee \neg p) \wedge q) \\
& \equiv (\neg p \vee r) \wedge (q \vee r) \wedge ((\neg r \wedge q) \vee (\neg p \wedge q)) \\
& \equiv (\neg p \vee r) \wedge (q \vee r) \wedge (((\neg r \wedge q) \vee \neg p) \wedge ((\neg r \wedge q) \vee q)) \\
& \equiv (\neg p \vee r) \wedge (q \vee r) \wedge (\neg r \vee \neg p) \wedge (q \vee \neg p) \wedge (\neg r \vee q) \wedge (q \vee q) \\
& \equiv (\neg p \vee r) \wedge (q \vee r) \wedge (\neg r \vee \neg p) \wedge (q \vee \neg p) \wedge (\neg r \vee q) \wedge q
\end{aligned}
$$

- In the "semantic" approach, i.e., using truth tables
  - we are stuck with the exponentially-sized truth table.
- However, with the "syntactic" approach, i.e., using known equivalences
  - we can often achieve our aims much more quickly
    - though this often requires cunning!

An application: SAT-solving

- The power of propositional logic is quite remarkable
  – computationally complex problems can be described using the logic.
- The aim of SAT-solving is
  – to encode a problem X as a propositional formula φ so that
    - a solution to X corresponds to φ having a satisfying truth assignment

  - to employ algorithms to solve the satisfiability problem (SAT) for φ (and so X).
- The SAT problem is to decide if a propositional formula has a satisfying truth assignment. It is extremely hard to solve.
  – in fact, it is **NP**-complete, even if the formula is given in c.n.f.
    - so takes time exponential in the size of the formula to solve (probably!).
- However, modern-day SAT-solvers can give extremely good results
  – note that all modern day SAT-solvers need their inputs to be in c.n.f.
- SAT-solving is a thriving research area
  – `http://www.satlive.org`.

An application: SAT-solving

- Consider the graph G shown opposite where the problem is
  - to decide whether the vertices can be coloured red, yellow, or blue such that
    - if two vertices are joined by an edge then they must be coloured differently.
- Consider the formula φ defined as

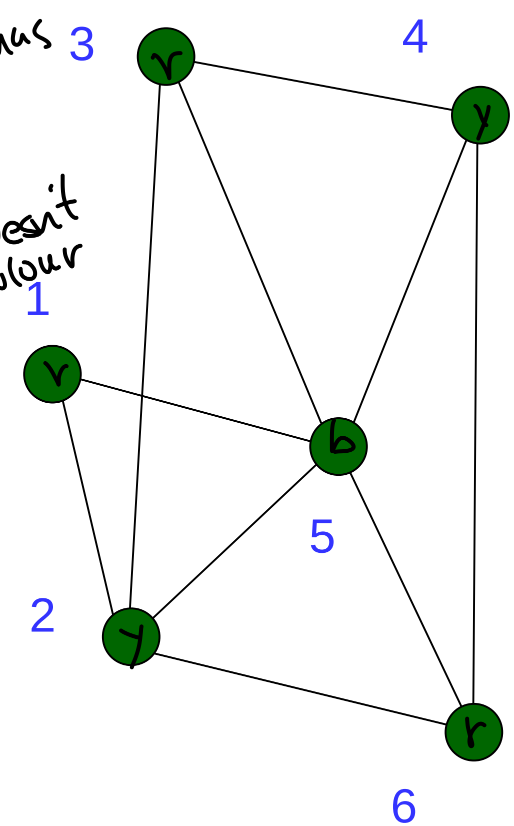
the balls are either red, yellow or blue

$$(r_1 \vee y_1 \vee b_1) \wedge (r_2 \vee y_2 \vee b_2) \wedge \ldots \wedge (r_6 \vee y_6 \vee b_6)$$
$$\wedge\ (\neg r_1 \vee \neg y_1) \wedge (\neg r_1 \vee \neg b_1) \wedge (\neg b_1 \vee \neg y_1)$$
$$\wedge\ (\neg r_2 \vee \neg y_2) \wedge (\neg r_2 \vee \neg b_2) \wedge (\neg b_2 \vee \neg y_2)$$
$$\wedge\ \ldots \wedge (\neg r_6 \vee \neg y_6) \wedge (\neg r_6 \vee \neg b_6) \wedge (\neg b_6 \vee \neg y_6)$$

each ball only has one colour

$$\wedge\ (\neg r_1 \vee \neg r_2) \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg y_1 \vee \neg y_2)$$
$$\wedge\ (\neg r_1 \vee \neg r_5) \wedge (\neg b_1 \vee \neg b_5) \wedge (\neg y_1 \vee \neg y_5)$$
$$\wedge\ \ldots \wedge (\neg r_5 \vee \neg r_6) \wedge (\neg b_5 \vee \neg b_6) \wedge (\neg y_5 \vee \neg y_6)$$

neighbours doesn't share same colour



- It is not hard to prove that
  - G can be 3-coloured

  if and only if
  - φ has a satisfying truth assignment.

r y r y b r $\xrightarrow{f}$ φ $\rightarrow$ True

An application: SAT-solving

Durham University

- A clause is a non-tautological disjunction of literals.
- If every clause contains exactly k literals, then we obtain the k-SAT problem.
- It is known that k-SAT is polynomial-time solvable if k=2 but NP-complete for k>=3.
- Suppose we consider formulas where
  - every clause contains exactly k distinct literals
  - every variable appears in at most s clauses

  This yields the (k,s)-SAT problem.

v. g. b s.t(— — —),
(v,g), (v.b), (g,b)

2-2-SAT

- It is known: every instance of (3,3)-SAT is satisfiable, but (3,4)-SAT is NP-complete.
- Iwama and Takaki (Satisfiability of 3CNF formulas with small clause/variable-ratio. DIMACS Series in Disc. Math. and Theoret. Comput. Sc, 35 (1997) 315–334) proved that

  - every instance of (3,4)-SAT with at most 3 variables occurring in four clauses is satisfiable.

  - there exists an instance of (3,4)-SAT with 9 variables occurring in four clauses that is not satisfiable.

Research question: Can we close this gap?

See also S. Hoory and S. Szeider, Computing unsatisfiable k-SAT instances with few occurrences per variable, Theoretical Computer Science 337(2005) 347–359.