Informatics 1 - Computation & Logic: Tutorial 4

Satisfiability and Resolution

Week 6: 23-27 October 2017

Please attempt the entire worksheet in advance of the tutorial, and bring all work with you. Tutorials cannot function properly unless you study the material in advance. Attendance at tutorials is **obligatory**; please let the ITO know if you cannot attend.

You may work with others, indeed you should do so; but you must develop your own understanding; you can't phone a friend during the exam. If you do not master the coursework you are unlikely to pass the exams.

Introduction

Resolution is a procedure used to search for a state that satisfies a given set of constraints, Σ. The idea is to choose one variable, X, the resolution variable and to replace the clauses that mention that variable by a set of new clauses, to give a new, simpler, set of constraints Σ_X such that:

- X does not occur in Σ_X
- we can extend any valuation V that satisfies Σ_X to a valuation that satisfies Σ.

To resolve on X, we factor out the literals X and $\neg X$ to write

$$\Sigma = (X \vee \Gamma) \wedge (\neg X \vee \Delta) \wedge \Omega$$

where Γ, Δ, Ω are sets of constraints that do not mention X.

Clearly[1] if we find a state V that satisfies $\Delta \wedge \Omega$ then we can extend V with the value $X = \top$ to produce a state that satisfies Σ. Similarly, if we find a state that satisfies $\Gamma \wedge \Omega$ we can extend V with the value $X = \bot$ to satisfy Σ.

We define $\Sigma_X = (\Gamma \vee \Delta) \wedge \Omega$. This definition means that, if we find a state that satisfies Σ_X it satisfies either Γ or Δ, or both, and also satisfies Ω. So, as pointed out above, we can extend it to a state that satisfies Σ.

If there is no state satisfying Σ_X, then Σ is not satisfiable.

[1] If this is not clear, go back to look again at last week's tutorial activity.

Example 1

Here is an example, where we begin by resolution on the variable A.

$$\begin{aligned}
\Sigma &= (A \vee B \vee \neg D) \wedge (\neg A \vee D \vee E) \wedge (\neg A \vee \neg C \vee E) \wedge (B \vee C \vee E) \wedge (\neg B \vee D \vee \neg E) \\
&= \quad A \vee (B \vee \neg D) && \Gamma = B \vee \neg D \\
&\quad \wedge \neg A \vee ((D \vee E) \wedge (\neg C \vee E)) && \Delta = (D \vee E) \wedge (\neg C \vee E) \\
&\quad \wedge (B \vee C \vee E) \wedge (\neg B \vee D \vee \neg E) && \Omega = (B \vee C \vee E) \wedge (\neg B \vee D \vee \neg E) \\
\Sigma_A &= (\Gamma \vee \Delta) \wedge \Omega \\
&= ((B \vee \neg D) \vee ((D \vee E) \wedge (\neg C \vee E))) \wedge (B \vee C \vee E) \wedge (\neg B \vee D \vee \neg E) \\
&= (B \vee \neg D \vee D \vee E) \wedge (B \vee \neg D \vee \neg C \vee E) \wedge (B \vee C \vee E) \wedge (\neg B \vee D \vee \neg E) \\
&= (B \vee \neg D \vee \neg C \vee E) \wedge (B \vee C \vee E) \wedge (\neg B \vee D \vee \neg E)
\end{aligned}$$

Observe that each pair of clauses, where one includes A and the other includes $\neg A$ (we call this a *resolution pair*), gives rise to a term in $\Gamma \wedge \Delta$ – sometimes this term will be trivial. We have reduced the problem of satisfying Σ to the problem of solving a simpler set of constraints, Σ_A. We will continue, to resolve on the remaining variables until the procedure stops. However, to avoid continually rewriting long equations, we will first simplify our notation.

Clausal Form

Since $\vee$ is associative, $(A \vee B) \vee C = A \vee (B \vee C)$, commutative, $(A \vee B) = (B \vee A)$, and idempotent, $(A \vee A) = A$, we can represent each non-trivial clause by a finite set of literals. Since $\wedge$ is also associative, commutative and idempotent, we represent a CNF as a finite set of clauses — a set of sets of literals — a so-called *clausal form*. Here are Σ and Σ_A, from Example 1, represented as clausal forms:

$$\Sigma = \big\{\{A, B, \neg D\}, \{\neg A, D, E\}, \{\neg A, \neg C, E\}, \{B, C, E\}, \{\neg B, D, \neg E\}\big\}$$
$$\Sigma_A = \big\{\{B, \neg D, \neg C, E\}, \{B, C, E\}, \{\neg B, D, \neg E\}\big\}$$

States or Valuations

A state or valuation is an assignment of $\top, \bot$ values to propositional letters (atoms). We can represent a state the *conjunction* of any set of literals in which X and $\neg X$ do not both occur. A state V (a set of literals, viewed as a conjunction) satisfies a clausal form iff each clause (a set of literals, viewed as a disjunction) in the clausal form includes at least one of the literals in V.

$$V \models \Sigma \quad \textbf{iff} \quad \textbf{for all } C \in \Sigma \,.\, C \cap V \neq \emptyset \tag{1}$$

To avoid confusion with a non-trivial clause, which represents the *disjunction* of such a set of literals, we will normally write out each state as an explicit conjunction.

We remark on two special cases of 1. The empty clause represents $\bot$, the empty disjunction (of no literals). If $\{\} \in \Sigma = \big\{\{\}, \ldots\big\}$, then Σ can never be satisfied, since $\{\} \cap V = \emptyset$. The empty clausal form, $\{\}$, with no clauses, represents $\top$. It is vacuously satisfied by every state since there is no $C \in \{\}$.

Resolution

When using resolution to solve a set of constraints, we use clausal form to set out our working. Here is how we would set out our resolution on A:

A	
${}^{A}\{A, B, \neg D\}$	~~$\{B, \neg D, D, E\}$~~
${}^{A}\{\neg A, D, E\}$	$\{B, \neg D, \neg C, E\}$
${}^{A}\{\neg A, \neg C, E\}$	
$\{B, C, E\}$	
$\{\neg B, D, \neg E\}$	

Each resolution pair gives rise to a resolvent

$$\frac{\{A, B, \neg D\} \qquad \{\neg A, D, E\}}{\{B, \neg D, D, E\}}\ (A) \qquad \frac{\{A, B, \neg D\} \qquad \{\neg A, \neg C, E\}}{\{B, \neg D, \neg C, E\}}\ (A)$$

We have two resolution pairs so we have two resolvents, but one is trivial; it includes both D and $\neg D$ — we strike it out. The clauses used to form the resolution pairs are annotated, ${}^{A}-$, to show that they have been used for resolution on A —this information wil be useful later. The remaining clauses represent Σ_A.We continue...

A	B	
${}^{A}\{A, B, \neg D\}$	~~$\{B, \neg D, D, E\}$~~	~~$\{C, E, D, \neg E\}$~~
${}^{A}\{\neg A, D, E\}$	${}^{B}\{B, \neg D, \neg C, E\}$	~~$\{\neg D, \neg C, E, D, \neg E\}$~~
${}^{A}\{\neg A, \neg C, E\}$		
${}^{B}\{B, C, E\}$		
${}^{B}\{\neg B, D, \neg E\}$		

Both resolvents are trivial. There are no remaining non-trivial constraints, so we can pick any values for the remaining atoms, C, D, E.

For example, suppose we choose to make them all false: $\neg C \wedge \neg D \wedge \neg E$. We first choose a value of B to make all of the clauses labelled ${}^{B}-$ true —we know that at least one value will work, but we have to check which one. To satisfy the clause $\{B, C, E\}$, when C and E are false, we have to make B true. Since this is forced, we know that the other ${}^{B}-$labelled clauses will be satisfied —but it is good to check in case there are mistakes in our working. In a state such that $B \wedge \neg C \wedge \neg D \wedge \neg E$, every clause that includes at least one of the conjoined literals is satisfied. So it suffices to observe that $\neg E \in \{\neg B, D, \neg E\}$ and $B \in \{B, \neg D, \neg C, E\}$.

Now, given that $B \wedge \neg C \wedge \neg D \wedge \neg E$, we are in a position to choose a value for A —we know there is one— that will satisfy all the clauses labelled ${}^{A}-$. This time, our choice is forced by $\{\neg A, D, E\}$; we have to make A false, so our final valuation is given by $\neg A \wedge B \wedge \neg C \wedge \neg D \wedge \neg E$. Again, you should double-check this result, by checking that the two remaining clauses are indeed satisfied by this valuation.

If the constraints are satisfiable, resolution will stop when there are no more resolution pairs. Either there will be no remaining clauses, or every remaining literal will be pure —meaning that its negation does not appear— in which case we must make at least one literal in each remaining clause true, to produce a satisfying valuation (and we can just make them all true). If the constraints are not satisfiable the procedure will eventually produce the empty clause, at which point we can stop.

Homework

1. To derive a satisfying state for Example 1, we chose to make C, D, E all false and then found values for B and A to satisfy the original constraints. Find values for B and A to extend the remaining 7 valuations of CDE so that the origiinal constraints are satisfied.

CDE	B	A	
$\bot\bot\top$	$\bot$	$\top$	or, $A = \bot$
$\bot\top\bot$	$\top$	$\top$	or, $A = \bot$
$\bot\top\top$	$\top$	$\top$	or, $B = \top, A = \bot$, or $B = \bot, A = \top$
$\top\bot\bot$	$\top$	$\bot$	or, $B = \bot$
$\top\bot\top$	$\bot$	$\top$	or, $A = \bot$
$\top\top\bot$	$\top$	$\bot$	
$\top\top\top$	$\top$	$\top$	or, $B = \top, A = \bot$, or $B = \bot, A = \top$

2. In applying resolution, we can choose which variable to resolve, taking the variables in any order. Redo the example, resolving first on E, and then on the remaining variables in reverse-alphabetical order.

$\{A, B, \neg D\}$

$\{\neg A, D, E\}$

$\{\neg A, \neg C, E\}$

$\{B, C, E\}$

$\{\neg B, D, \neg E\}$

E	D	
$^{D}\{A, B, \neg D\}$ $^{E}\{\neg A, D, E\}$ $^{E}\{\neg A, \neg C, E\}$ $^{E}\{B, C, E\}$ $^{E}\{\neg B, D, \neg E\}$	$^{D}\{\neg A, D, \neg B\}$ $^{D}\{\neg A, \neg C, \neg B, D\}$ ~~$\{B, C, \neg B, D\}$~~	~~$\{A, B, \neg A, \neg B\}$~~ ~~$\{A, B, \neg A, \neg C, \neg B\}$~~

3. Our next example includes 6 clauses:

$$\{\{A, B\}, \{A, \neg B, \neg C\}, \{\neg A, D\}, \{\neg B, C, D\}, \{\neg B, \neg D\}, \{\neg A, B, \neg D\}\}$$

(a) Following the same procedure as before, resolve on $ABCD$ in turn.

A	B	C	D	
$^{A}\{A, B\}$ $^{A}\{A, \neg B, \neg C\}$ $^{A}\{\neg A, D\}$ $^{B}\{\neg B, C, D\}$ $^{B}\{\neg B, \neg D\}$ $^{A}\{\neg A, B, \neg D\}$	$^{B}\{B, D\}$ $^{B}\{\neg B, \neg C, D\}$ $^{B}\{B, \neg D\}$ ~~$\{\neg B, \neg C, B, \neg D\}$~~	$^{C}\{\neg C, D\}$ $^{C}\{C, D\}$ ~~$\{D, \neg D\}$~~ ~~$\{\neg C, D, \neg D\}$~~ ~~$\{C, D, \neg D\}$~~ $^{D}\{\neg D\}$	$^{D}\{D\}$	$\{\}$

(b) Was the empty clause found? Yes

(c) Is the clausal form satisfiable? No

4. This question concerns the resolution of the claim that the following set of expressions is consistent —that is to say, that there is at least one valuation that makes all of these expressions true.

$$\{P \to (Q \vee R), Q \to \neg S, S \oplus R, (R?Q : P), (Q \wedge R) \to T\}$$

To apply resolution to this problem you will first have to convert each expression to clausal form.

(a) Express each expression in clausal form.

i. $P \to (Q \vee R)$
$\neg P \vee (Q \vee R)$ by arrow elimination
$\neg P \vee Q \vee R$ by associativity
$\{\{\neg P, Q, R\}\}$

ii. $Q \to \neg S$
$\neg Q \vee \neg S$ by arrow elim
$\{\{\neg Q, \neg S\}\}$

iii. $S \oplus R$ $\{\{S, R\}, \{\neg S, \neg R\}\}$

iv. $(R?Q : P)$
$\{\{\neg R, Q\}, \{R, P\}\}$

v. $(Q \wedge R) \to T$
$\neg(Q \wedge R) \vee T$ by arrow elim
$(\neg Q \vee \neg R) \vee T$ by De Morgan
$\neg Q \vee \neg R \vee T$ by associativity
$\{\{\neg Q, \neg R, T\}\}$

(b) Use resolution to determine whether the conjunction of the expressions is consistent.
You should start from the union of the clausal forms given in your answers to 4a.

P	Q	R	S	
${}^{P}\{\neg P, Q, R\}$ ${}^{Q}\{\neg Q, \neg S\}$ ${}^{R}\{S, R\}$ ${}^{R}\{\neg S, \neg R\}$ ${}^{Q}\{\neg R, Q\}$ ${}^{P}\{R, P\}$ ${}^{Q}\{\neg Q, \neg R, T\}$	${}^{Q}\{Q, R\}$	$\{\neg R, \neg S\}$ (dup) ${}^{R}\{\neg R, T\}$ ~~$\{R, \neg R, T\}$~~ ${}^{R}\{R, \neg S\}$	${}^{S}\{\neg S\}$ ${}^{S}\{S, T\}$ ~~$\{S, \neg S\}$~~ ~~$\{\neg S, T\}$~~	$\{T\}$

Note that $\{\neg Q, \neg S\}$ and $\{\neg R, Q\}$ resolve on Q to $\{\neg R, \neg S\}$, but this is omitted as this clause is already in the resolution pool.
$\{\neg R, T\}$ and $\{R, \neg S\}$ resolve to give $\{\neg S, T\}$, but this is omitted as it is subsumed by $\{\neg S\}$.

(c) Is the original claim correct?

yes

(d) If it is, produce a satisfying valuation.

$S = \bot,\ T = \top,\ R = \top,$
$Q = \top,\ P = *\ .$

For the final question, we return to entailment, first mentioned in tutorial 2. There we gave the following definition.

> Given expressions φ, θ, ψ, we say φ and θ **entail** ψ (for which we write $\varphi, \theta \models \psi$) iff every state that satisfies both φ and θ also satisfies ψ. When working with an entailment $\varphi, \theta \models \psi$, we refer to φ, θ as its **premises** and ψ as its **conclusion**.

Here we will generalise this slightly to allow a finite set of expressions on either side of the 'gate', $\models$. We say that $\Gamma \models \Delta$ iff every state that satisfies every expression in Γ satisfies at least one expression in Δ.

We will normally omit the set-brackets $\{\}$, and other set-notation, when writing entilments. For example, if φ, ψ, θ are expressions, we write

$\Gamma \models \varphi$	**for**	$\Gamma \models \{\varphi\}$
$\Gamma, \theta \models \varphi, \psi$	**for**	$\Gamma \cup \{\theta\} \models \{\varphi, \psi\}$

Now consider what this definition of $\models$ means in some special cases.

5. What does the definition mean in the following examples?

   (a) The case where Δ is empty.
   What does it mean to say that $\Gamma \models \emptyset$
   (which we usually write simply as $\Gamma \models$)?

   The conjunction $\bigwedge \Gamma$ of all expressions in Γ is contradictory.

   (b) The case where Γ is empty.
   What does it mean to say that $\emptyset \models \Delta$
   (which we usually write simply as $\models \Delta$)?

   The disjunction $\bigvee \Delta$ of all expressions in Δ is a tautology.

   (c) The case in which Γ is a set of literals.
   Is it true that $A, \neg B, C \models (A \oplus B) \leftrightarrow C$?

   Yes

   In general, if Γ is a set of literals how can we interpret $\Gamma \models \varphi$?

   The valuation represented by $\bigwedge \Gamma$ makes φ true.

   (d) What if both Γ and Δ are sets of literals?
   In this case, what does it mean to say that $\Gamma \models \Delta$?
   Hint: think about valuations and clauses.

   The intersection, $\Gamma \cap \Delta$ is non-empty: $\Gamma \cap \Delta \neq \emptyset$

   (e) What does it mean to say that $\Gamma \not\models \varphi$?
   Can you give a set of constraints that is satisfiable iff $\Gamma \not\models \varphi$?

   It means that some valuation satisfying each $\theta \in \Gamma$ does not satisfy φ .
   $\Gamma \not\models \varphi$ iff $\Gamma, \neg\varphi$ is satisfiable.

... see over ...

Tutorial Activity

1. As usual, you should start by working as a group to identify and resolve any problems with the homework

2. The main activity is designed to highlight one of the shortcomings of resolution. You will see that resolution makes heavy weather of a relatively simple combinatorial problem.

Consider this map of Australian States. It is coloured, with four colours, but two adjacent states (states that share a common border) have the same colour. We want a better colouring.

You have probably heard of the four-colour theorem — any planar map can be coloured with four colours so that no two adjacent states have the same colour. However, here we have not coloured the sea, and to extend a four-colouring of the states to include the sea we would need a fifth colour.

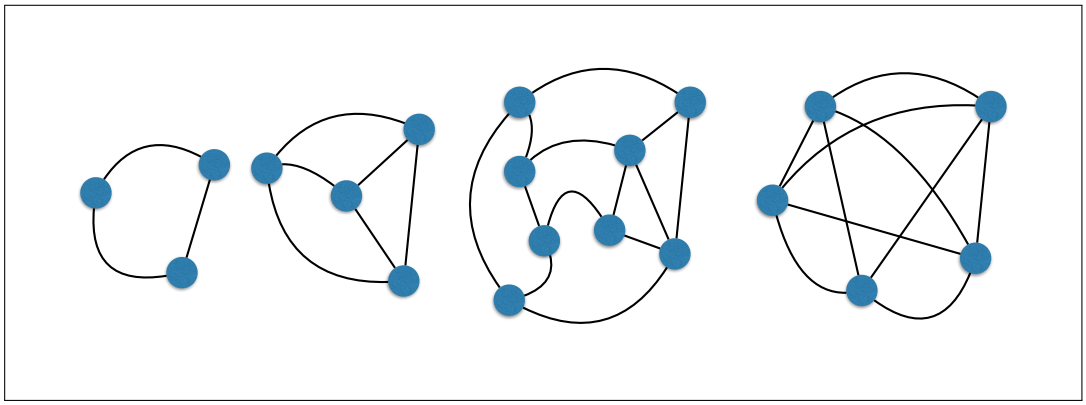
So, there must be a 3-colouring of the states. Of course, it is easy to find one, but we want to use this problem to learn something about resolution.

... see over ....

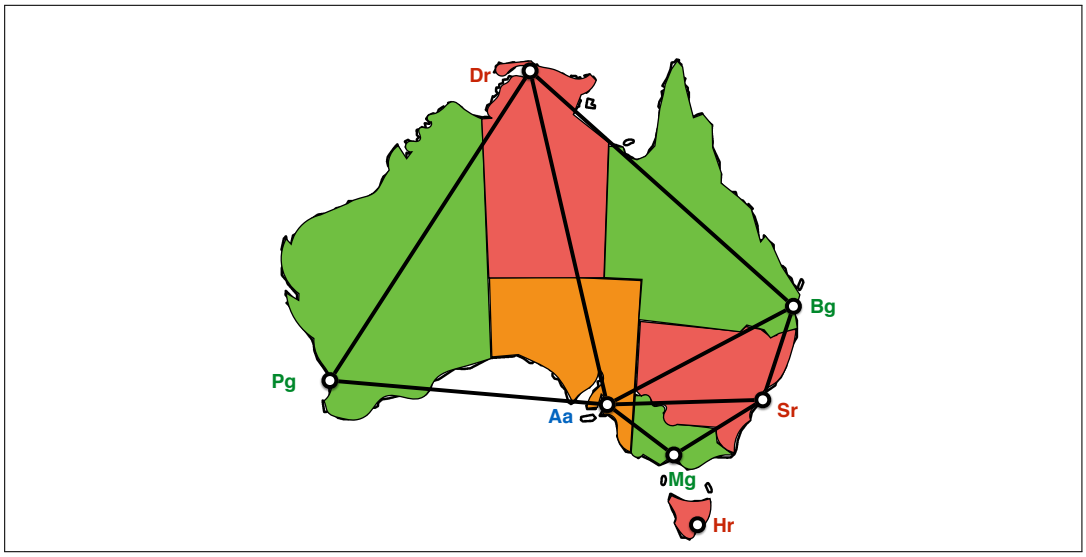
We first throw away some irrelevant detail by converting this to a problem about graphs. An undirected graph is just a symmetric binary relation E on a set N of nodes, we call this an adjacency relation. If $E(a, b)$ we say there is an edge $\langle a, b \rangle$, between a and b. We can draw a diagram to help us visualise the adjacency relation.

For example, a triangle has three nodes and three edges, a tetrahedron has four nodes and six edges, and a cube has eight nodes and twelve edges.

(a) Draw these three graphs.
Can you draw them so that no edges cross each other?
(If you can, they are *planar* graphs.)
Can you find an example of a non-planar graph?



(b) To present our example as a graph, let the nodes be the seven states, and let X and Y be adjacent iff they share a common border. Draw the graph. Use the initial letters of the state capitals as names for the seven nodes, M, S, H, D, P, A, B.

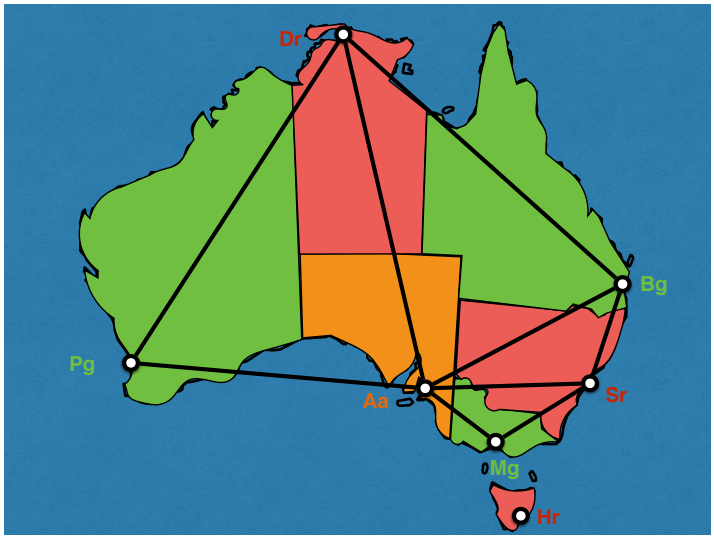


(c) How many edges are there in your Australia graph?

9

We can now replace the map colouring problem by the graph colouring problem. Can we colour each node so that no pair of adjacent nodes have the same colour — or equivalently, so that any two adjacent nodes have different colours?

(d) As a first exercise, suppose we have three colours, $C = \{r, a, g\}$ (red, amber, green). Without thinking about logic (for a brief moment) can you count the number of three-colourings of your graph?
Hint: If you arbitrarily fix the colouring of two adjacent nodes, can you extend your colouring to the entire graph?



(e) How many distinct colourings are there of this graph?

$18 = 3 \times 2 \times 3$ — 6 ways of colouring 2 adjacent nodes, times 3 ways of colouring Hobart. the rest is then fixed.

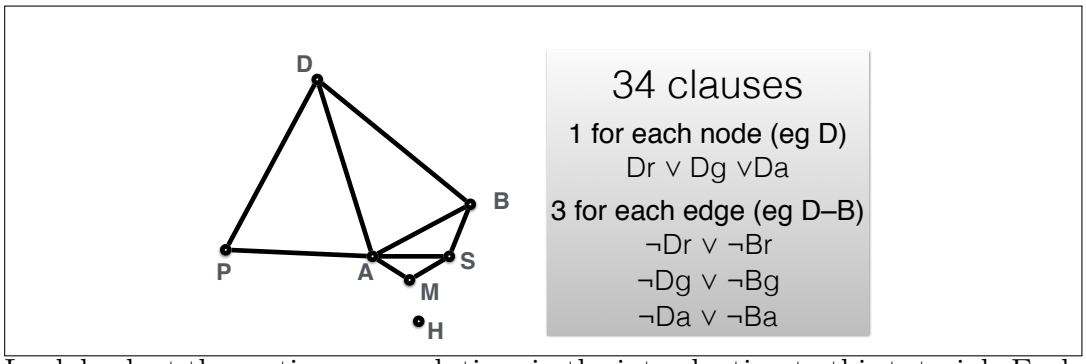
Now return to logic. Consider a propositional language with 21 atoms; for each Node X and each colour y we introduce an atom Xy whose intended interpretation is that the state whose capital is X can be coloured with colour y. For example, a solution might make Hr, Ha, Hg all true, since it doesn't matter what colour we give to the isolated island of Tasmania.

... see over ....

(f) First construct a set of constraints, Σ, in CNF, such that for any valuation V satisfying these constraints,

- **for each** $X \in N$. **there is some** $y \in C$.
  such that X may be coloured with y.
- **for each** edge $(X, Y) \in E$. **for each** $y \in C$.
  it is not the case that both X and Y can be coloured with y.

Σ should include one clause for each node (we will call these the node-clauses) and three (one for each colour) for each edge of the graph(we will call these the edge-clauses). Describe these clauses, but don't bother to write them all out in full.



Look back at the section on resolution, in the introduction to this tutorial. Each resolution pair gives rise to a resolvent. Check that any state satistfying the two paired clauses will also satisfy the resolvent. Rather than embark immediately on a full resolution, we can use this fact to produce clauses that will be satisfied by any state satisfying the constraints.

(g) For example, arbitrarily fix the colours for two adjacent nodes by adding two unit clauses $Pr \wedge Aa$ to your constraints. Use resolution to find constraints satisfied by any colouring that extends this choice.
Hint: first use resolution to determine the colour of a third node, by resolving the constraints given below. Then explain how resolution on the full set of constraints would generate enough information to lead to a full colouring.

Dr	Pr	Da	Aa	
^{Pr}Pr ^{Aa}Aa $^{Dr}Dr, Da, Dg$ $^{Dr}\neg Pr, \neg Dr$ $^{Da}\neg Aa, \neg Da$	$^{Pr}\neg Pr, Da, Dg$	$^{Da}Da, Dg$	$^{Aa}\neg Aa, Dg$	Dg

(h) Before attempting resolution on these constraints to solve the general satisfaction problem it is useful to extimate the magnitude of the task.

For each node clause, how many edge clauses can it be paired with to make a resolution pair? Can you estimate how many different clauses we might produce by resolution from Σ?

Each node clause pairs with three edge clauses for each edge from that node.
Each resolvent then pairs with the two remaining edge clauses, for each edge.
If there are e edges from some node, we can generate at least $6e^2$ clauses — too many for manual processing.

(i) Suppose we have a 3-colouring of the Australia graph that makes Brisbane green (for example). Can you deduce that any other nodes will have the same colour?

The vertices of any triangle must use all three colours;
P and M must have the same colour as B

(j) Express one such deduction as an entailment, $\varphi, \Sigma \models \theta$, where φ and θ are atoms of our language.
Which constraints from Σ are needed to derive this entailment?

For example, $Ba, \Sigma \models Pa$
We need $Da \vee Dr \vee Dg$, $Aa \vee Ar \vee Ag$, $Pa \vee Pr \vee Pg$,
together with $\neg Ba \vee \neg Da$, $\neg Ba \vee \neg Aa$,
to constrain the possible colours for D and A,
and the edge constraints for r and g
for each of the three edges of the triangle PDA.

(k) Compare your results, and discuss whether/how fixing the two colours in part (2g) affects the process of resolution.

This example shows how a 'combinatorial explosion' can affect resolution.
This happens whenever there are symmetries in the problem.
By chosing two colours arbitrarily, we fix one of the six possible colourings.
In the second example, where we fix only one colour, we have to do more work, because we have to cover two possibilities for the colours of D and A.

Practical SAT solvers no longer use resolution directly, but they do use these ideas to perform an efficient search for a satisfying solution.
For an example, see:
`https://www.msoos.org/2013/09/minisat-in-your-browser/`

This tutorial exercise sheet was written by Dave Cochran and Michael Fourman, drawing on material from an earlier tutorials produced by Paolo Besana, Thomas French, and Areti Manataki. Send comments to `Michael.Fourman@ed.ac.uk`

Fuller explanation for 2(h) – by Sára Decova.

Let's take a state node S with e neighbours X^i where $1 \leq i \leq e$.

A **node clause** for state S is $\{S_r, S_a, S_g\}$ - meaning a state is either red, amber, or green. There are e edges from that node to its neighbours. An edge to a neighbour X^i is represented by three **edge clauses**: $\{\neg S_r, \neg X_r^i\}$, $\{\neg S_a, \neg X_a^i\}$, $\{\neg S_g, \neg X_g^i\}$.

Notice that the node clause for S shares a literal with every edge clause (in particular the S-literals), and the sign of the literals is oposite. Using resolution, we can therefore pair the node clause with any of the **3e** neighbouring edge clauses.

Let's say we pick S_r and neighbour j. The corresponding edge clause is $\{\neg S_r, \neg X_r^j\}$ and the resolution yields:

$$\begin{array}{l|l} {}^{S}r\{\neg S_r, \neg X_r^j\} & \{\neg \mathbf{X_r^j}, \mathbf{S_a}, \mathbf{S_g}\} \\ {}^{S}r\{S_r, S_a, S_g\} & \end{array}$$

The resulting clause has two S-literals left. As we did resolution on "red", the next resolution can be on S_a or S_g (amber or green). There are exactly **2e** edge clauses containing either $\neg S_a$ or $\neg S_g$, two for every neighbour.

The resultant clause will have one S-literal corresponding to one colour. This literal occurs exactly in one clause for each neighbour. So, for the last resolution step, we have a choice of **e** such clauses.

To sumarise, we have:

- 3e options for first resoltion from node clause - choose any edge clause for any neighbour
- 2e options for second resolution - any of the 2 remaining colours for each neighbour
- e options for third (last) resolution - 1 remaining colour for each neighbour

Together there are $3e * 2e * e = 6e^3$ possible clauses after resolution on **neighbouring edges only**.