PROPOSITIONAL LOGIC

MATHEMATICAL LOGIC

- Logic is the foundation of computation.
- We will use logic for multiple purposes:
  - Describing specifications
  - Describing program executions
  - Mathematical guarantees of logic will translate to guarantees of program correctness
  - Decision procedures for logic will be used for verification.

PROPOSITIONAL LOGIC

Is $p \to q \to r \leftrightarrow (p \land q) \to r$ valid?

Is $p \land \bot \to \neg q \lor \top$ satisfiable?

SYNTAX

Atom	Truth Values - ⊥ : False, ⊤ : True Propositional Variables - p,q,r…
Logical Connectives	∧ : and, ∨ : or, ¬ : not, →: implies, ↔: if and only if(iff)
Literal	Atom or its negation
Formula	A literal or the application of logical connectives to formulae

SEMANTICS

Interpretation I

I : Set of Propositional Variables → { ⊥ , ⊤ }

MODEL OF

Given an interpretation I and Formula F,

$I \models F$	F evaluates to ⊤ under I
$I \not\models F$	F evaluates to ⊥ under I

SEMANTICS: INDUCTIVE DEFINITION

Base Case:

$I \vDash \top$	
$I \nvDash \bot$	
$I \vDash p$	iff I(p)=$\top$
$I \nvDash p$	iff I(p)=$\bot$

Inductive Case:

$I \vDash \neg F$	iff $I \nvDash F$
$I \vDash F_1 \wedge F_2$	iff $I \vDash F_1$ and $I \vDash F_2$
$I \vDash F_1 \vee F_2$	iff $I \vDash F_1$ or $I \vDash F_2$
$I \vDash F_1 \rightarrow F_2$	iff $I \nvDash F_1$ or $I \vDash F_2$
$I \vDash F_1 \leftrightarrow F_2$	iff $I \vDash F_1$ and $I \vDash F_2$, or $I \nvDash F_1$ and $I \nvDash F_2$
Other cases …	

EXAMPLE

$I = \{p : True, q : False\}$ $F = p \land q \rightarrow p \lor \neg q$

Is $I \models F$?

1. $I \not\models q$
2. $I \not\models p \land q$
3. $I \models p \land q \rightarrow p \lor \neg q$

PRECEDENCE OF LOGICAL CONNECTIVES

- We assume the following precedence from highest to lowest:
  - $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$
  - Example: $\neg p \wedge q \rightarrow p \vee q \wedge r$ is the same as $((\neg p) \wedge q) \rightarrow (p \vee (q \wedge r))$.
- We assume that all logical connectives associate to the right.
  - Example: $p \rightarrow q \rightarrow r$ is the same as $p \rightarrow (q \rightarrow r)$
- Parenthesis can be used to change precedence or associativity.

SATISFIABILITY AND VALIDITY

- A formula F is satisfiable iff there exists an interpretation I such that $I \vDash F$.
- A formula F is valid iff for all interpretations I, $I \vDash F$.
- A formula F is valid iff $\neg F$ is unsatisfiable.
  - A Decision Procedure for satisfiability is therefore also a decision procedure for validity. How?

QUESTIONS

- A formula can either be SAT, UNSAT or VALID.
  - Does Validity ⇒ Satisfiability?
  - Does Satisfiability ⇒ Validity?
- Can a decision procedure for Validity be used as a decision procedure for Satisfiability?
  - F is satisfiable iff ¬F is not valid.
- Are the following formulae are sat, unsat or valid?
  - $p \wedge q \rightarrow p \vee q$
  - $p \vee q \rightarrow \neg p \vee \neg q$
  - $(p \rightarrow q \rightarrow r) \wedge (p \wedge q \wedge \neg r)$

MORE TERMINOLOGY

- Formulae F_1 and F_2 are equivalent (denoted by $F_1 \Leftrightarrow F_2$) when the formula $F_1 \leftrightarrow F_2$ is valid.
  - Example: $p \rightarrow q \Leftrightarrow \neg p \vee q$
  - Another definition: F_1 and F_2 are equivalent if for all interpretations I, $I \vDash F_1$ if and only if $I \vDash F_2$.
- Formula F_1 implies F_2 (denoted by $F_1 \Rightarrow F_2$) when the formula $F_1 \rightarrow F_2$ is valid.
  - Example: $(p \rightarrow q) \wedge p \Rightarrow q$
- Formulae F_1 and F_2 are equisatisfiable when F_1 is satisfiable if and only if F_2 is satisfiable.
  - Example: $p \wedge (q \vee r)$ and $q \vee r$ are equisatisfiable

MORE EXAMPLES

- Which of the following are true?
  - $\neg(F_1 \wedge F_2) \Leftrightarrow \neg F_1 \vee \neg F_2$
  - $(F_1 \leftrightarrow F_2) \wedge (F_2 \leftrightarrow F_3) \Rightarrow (F_1 \leftrightarrow F_3)$
  - $p \Leftrightarrow p \wedge q$
  - p and q are equisatisfiable.
- What is the simplest example of two formulae which are not equisatisfiable?

DECISION PROCEDURES FOR SATISFIABILITY AND VALIDITY

- Two methods
  - Truth Tables: Search for satisfying interpretation
  - Semantic Argument: Rule-based deductive approach
- Modern SAT solvers use combination of both approaches

TRUTH TABLES - EXAMPLE

$$p \wedge q \rightarrow p \vee \neg q$$

p	q	$\neg q$	$p \wedge q$	$p \vee \neg q$	$p \wedge q \rightarrow p \vee \neg q$
0	0	1	0	1	1
0	1	0	0	0	1
1	0	1	0	1	1
1	1	0	1	1	1

TRUTH TABLES - EXAMPLE

$p \wedge q \rightarrow p \vee \neg q$ is valid

p	q	$\neg q$	$p \wedge q$	$p \vee \neg q$	$p \wedge q \rightarrow p \vee \neg q$
0	0	1	0	1	1
0	1	0	0	0	1
1	0	1	0	1	1
1	1	0	1	1	1

SEMANTIC ARGUMENT METHOD

- Deductive approach for showing validity based on proof rules
- Main Idea: Proof by Contradiction.
  - Assume that a falsifying interpretation exists.
  - Use proof rules to deduce more facts.
  - Find contradictory facts.

PROOF RULES (NEGATION)

$$\frac{I \models \neg F}{I \not\models F}$$

PROOF RULES (NEGATION)

PREMISE

$$\frac{I \models \neg F}{I \not\models F}$$

PROOF RULES (NEGATION)

$$\frac{I \models \neg F}{I \not\models F}$$

PREMISE

DEDUCTION

PROOF RULES (NEGATION)

PREMISE

$$\frac{I \models \neg F}{I \not\models F}$$

DEDUCTION

$$\frac{I \not\models \neg F}{I \models F}$$

PROOF RULES (CONJUNCTION)

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}}$$

PROOF RULES (CONJUNCTION)

$$\frac{I \models F \wedge G}{\begin{array}{l} I \models F \\ I \models G \end{array}}$$

$$\frac{I \not\models F \wedge G}{I \not\models F \quad | \quad I \not\models G}$$

PROOF RULES (CONJUNCTION)

$$\frac{I \models F \land G}{\begin{array}{l} I \models F \\ I \models G \end{array}}$$

$$\frac{I \not\models F \land G}{I \not\models F \quad | \quad I \not\models G}$$

BRANCHING:

Need to show a contradiction in every branch

PROOF RULES (DISJUNCTION)

$$\frac{I \models F \vee G}{I \models F \quad | \quad I \models G}$$

$$\frac{I \not\models F \vee G}{\begin{array}{l} I \not\models F \\ I \not\models G \end{array}}$$

PROOF RULES (IMPLICATION)

$$\frac{I \models F \to G}{I \not\models F \quad | \quad I \models G}$$

$$\frac{I \not\models F \to G}{\begin{array}{l} I \models F \\ I \not\models G \end{array}}$$

PROOF RULES (IFF)

$$\frac{I \models F \leftrightarrow G}{I \models F \land G \quad | \quad I \not\models F \lor G}$$

$$\frac{I \not\models F \leftrightarrow G}{I \models F \land \neg G \quad | \quad I \models \neg F \land G}$$

PROOF RULES (CONTRADICTION)

$$\frac{\begin{array}{lcl} I & \models & F \\ I & \not\models & F \end{array}}{I \models \bot}$$

EXAMPLE

Prove that $p \wedge q \rightarrow p \vee \neg q$ is valid

EXAMPLE

Prove that $p \land q \rightarrow p \lor \neg q$ is valid

$$I \not\models p \land q \rightarrow p \lor \neg q$$

EXAMPLE

Prove that $p \land q \to p \lor \neg q$ is valid

$$\frac{I \not\models p \land q \to p \lor \neg q}{I \models p \land q \qquad I \not\models p \lor \neg q}$$

EXAMPLE

Prove that $p \land q \rightarrow p \lor \neg q$ is valid

$$\frac{I \not\models p \land q \rightarrow p \lor \neg q}{I \models p \land q \qquad I \not\models p \lor \neg q}$$

$$\frac{I \models p \land q}{\begin{array}{c} I \models p \\ I \models q \end{array}} \qquad \frac{I \not\models p \lor \neg q}{\begin{array}{c} I \not\models p \\ I \not\models \neg q \end{array}}$$

EXAMPLE

Prove that $p \land q \to p \lor \neg q$ is valid

$$\dfrac{I \not\models p \land q \to p \lor \neg q}{\dfrac{I \models p \land q}{\begin{array}{c} I \models p \\ I \models q \end{array}} \qquad \dfrac{I \not\models p \lor \neg q}{\begin{array}{c} I \not\models p \\ I \not\models \neg q \end{array}}}$$

CONTRADICTION

EXAMPLE WITH BRANCHING

Prove that $(p \rightarrow q) \land p \rightarrow q$ is valid

EXAMPLE WITH BRANCHING

Prove that $(p \rightarrow q) \wedge p \rightarrow q$ is valid

$$I \not\models (p \rightarrow q) \wedge p \rightarrow q$$

EXAMPLE WITH BRANCHING

Prove that $(p \to q) \wedge p \to q$ is valid

$$I \not\models (p \to q) \wedge p \to q$$

$$I \models (p \to q \wedge p) \qquad I \not\models q$$

EXAMPLE WITH BRANCHING

Prove that $(p \rightarrow q) \land p \rightarrow q$ is valid

$$I \not\models (p \rightarrow q) \land p \rightarrow q$$

$$\frac{I \not\models (p \rightarrow q) \land p \rightarrow q}{I \models (p \rightarrow q \land p) \qquad I \not\models q}$$

$$\frac{I \models (p \rightarrow q \land p)}{I \models (p \rightarrow q) \qquad I \models p}$$

EXAMPLE WITH BRANCHING

Prove that $(p \to q) \wedge p \to q$ is valid

$$I \not\models (p \to q) \wedge p \to q$$

$$I \models (p \to q \wedge p) \qquad I \not\models q$$

$$I \models (p \to q) \qquad I \models p$$

$$I \not\models p \quad \Big| \quad I \models q$$

EXAMPLE WITH BRANCHING

Prove that $(p \rightarrow q) \land p \rightarrow q$ is valid

$$I \not\models (p \rightarrow q) \land p \rightarrow q$$

$$I \models (p \rightarrow q \land p) \qquad I \not\models q$$

$$I \models (p \rightarrow q) \qquad I \models p$$

$$I \not\models p \quad \mid \quad I \models q$$

CONTRADICTION

EXAMPLE WITH BRANCHING

Prove that $(p \rightarrow q) \land p \rightarrow q$ is valid

$$I \not\models (p \rightarrow q) \land p \rightarrow q$$

$$I \models (p \rightarrow q \land p) \qquad I \not\models q$$

$$I \models (p \rightarrow q) \qquad I \models p$$

$$I \not\models p \quad | \quad I \models q$$

CONTRADICTION

Each branch should lead to a contradiction

ANNOUNCEMENTS

- Lectures slides and recorded video lectures are available on the course webpage.
- Chapter 1 of the BM book is uploaded on the course moodle page.
  - Please try Exercises 1.1-1.5.

QUESTIONS

- Is the semantic argument method complete?
- Can we use the semantic argument method for satisfiability?
- What is the time complexity of the semantic argument method?

DECISION PROCEDURES FOR SAT

- We will go through the DPLL algorithm.
  - Davis-Putnam-Logemann-Loveland Algorithm
  - Combines truth table and deductive approaches
  - Requires formulae in Conjunctive Normal Form (CNF)
  - Forms the basis of modern SAT solvers

NORMAL FORMS

- A Normal Form of a formula F is another equivalent formula F′ which obeys some syntactic restrictions.
- Three important normal forms:
    - Negation Normal Form (NNF): Should use only ¬, ∧ , ∨ as the logical connectives, and ¬ should only be applied to literals
    - Disjunctive Normal Form (DNF): Should be a disjunction of conjunction of literals
    - Conjunctive Normal Form (CNF): Should be a conjunction of disjunction of literals

CONJUNCTIVE NORMAL FORM

- A conjunction of disjunction of literals

$$\bigwedge_i \bigvee_j \ell_{i,j} \quad \text{for literals } \ell_{i,j}$$

- Each inner disjunct is also called a clause
- Is every formula in CNF also in NNF?

CNF CONVERSION

- We can use distribution of $\vee$ over $\wedge$ to obtain formula in CNF
  - $F_1 \vee (F_2 \wedge F_3) \Leftrightarrow (F_1 \vee F_2) \wedge (F_1 \vee F_3)$
  - Causes exponential blowup.
- Tseitin's transformation algorithm can be used to obtain an equisatisfiable CNF formula linear in size
  - BM Chapter 1

TRUTH TABLE BASED METHOD

Decision Procedure for Satisfiability:
Returns true if F is SAT, false if F is UNSAT

```
SAT(F){
  if (F = ⊤) return true;
  if (F = ⊥) return false;
  Choose a variable p in F;
  return SAT(F[⊤/p]) ∨ SAT(F[⊥/p]);
}
```

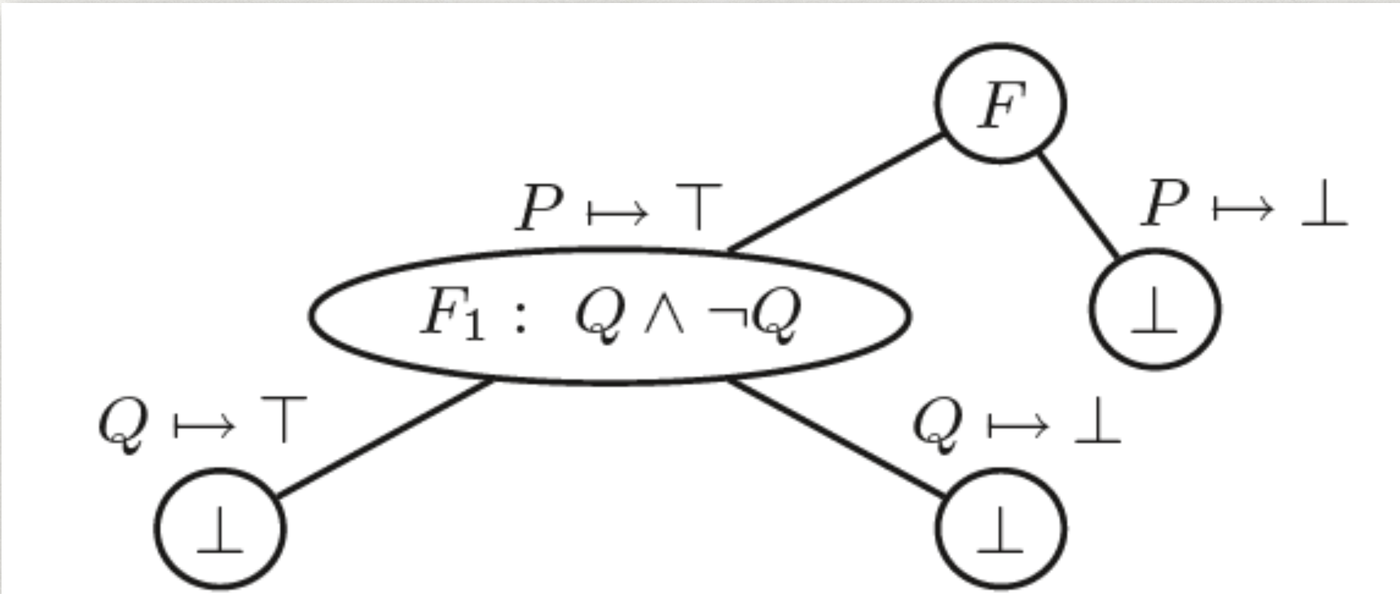
F[G/P] : G REPLACES EVERY OCCURRENCE OF P IN F, THEN SIMPLIFY

SIMPLIFICATION

- Following equivalences can be used to simplify:
  - $F \wedge \bot \Leftrightarrow \bot$
  - $F \wedge \top \Leftrightarrow F$
  - $F \vee \bot \Leftrightarrow F$
  - $F \vee \top \Leftrightarrow \top$
- Note that these equivalences would be applied syntactically.
  - That is, if the formula contains a $\top$ or $\bot$, it would be re-written according to the above equivalences.

EXAMPLE

- SAT$((P \to Q) \wedge P \wedge \neg Q)$
- $F = (\neg P \vee Q) \wedge P \wedge \neg Q$
- $F[\top / P] \triangleq (\bot \vee Q) \wedge \top \wedge \neg Q \equiv Q \wedge \neg Q$



SIMPLIFICATION MAY SAVE BRANCHING ON SOME OCCASIONS

DEDUCTION: CLAUSAL RESOLUTION

FOR CNF

$$\frac{I \vDash p \vee F \qquad I \vDash \neg p \vee G}{I \vDash F \vee G}$$

[CLAUSAL RESOLUTION]

- Given a CNF Formula $F = C_1, C_2, \ldots C_n$, if C' is a resolvent deduced from F, then $F' = C_1, C_2, \ldots, C_n, C'$ is equivalent to F.
- Example: $F = (\neg P \vee Q) \wedge P \wedge \neg Q$
  - Rewritten as $F = (\neg P \vee Q) \wedge (P \vee \bot) \wedge \neg Q$
  - Resolvent: $(Q \vee \bot) = Q$
  - $F' = (\neg P \vee Q) \wedge P \wedge \neg Q \wedge Q \rightarrow$ The next resolvent will be $\bot$.
- Idea: Repeatedly apply clausal resolution until no more new clauses can be deduced. If $\bot$ is never deduced, then the formula is satisfiable.

DEDUCTION: UNIT RESOLUTION

FOR CNF

$$\frac{I \models p \qquad I \models \neg p \vee F}{I \models F}$$

[UNIT RESOLUTION]

In Unit Resolution, the resolvent replaces the original clause

BOOLEAN CONSTRAINT PROPAGATION (BCP)

FOR CNF

$$I \models p \land (\neg p \lor q) \land (r \lor \neg q \lor s)$$

BOOLEAN CONSTRAINT PROPAGATION (BCP)

FOR CNF

$$\frac{I \models p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)}{I \models q \wedge (r \vee \neg q \vee s)} \quad \text{[UNIT RESOLUTION]}$$

BOOLEAN CONSTRAINT PROPAGATION (BCP)

FOR CNF

$$I \vDash p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)$$

$$I \vDash q \wedge (r \vee \neg q \vee s)$$

[UNIT RESOLUTION]

$$I \vDash r \vee s$$

BOOLEAN CONSTRAINT PROPAGATION (BCP)

FOR CNF

$$\frac{I \vDash p \wedge (\neg p \vee q) \wedge (r \vee \neg q \vee s)}{\frac{I \vDash q \wedge (r \vee \neg q \vee s)}{I \vDash r \vee s}}$$

[UNIT RESOLUTION]

FIND A SATISFYING INTERPRETATION

PURE LITERAL PROPAGATION (PLP)

FOR CNF

- If a variable appears only positively or negatively in a formula, then all clauses containing the variable can be removed.
  - p appears positively if every $p-$literal is just p
  - p appears negatively if every $p-$literal is $\neg p$
- Removing such clauses from F results in a equisatisfiable formula F'
  - Why?
  - Are F and F' equivalent?

DPLL

FOR CNF

Decision Procedure for Satisfiability of CNF Formula:
Returns true if F is SAT, false if F is UNSAT

```
SAT(F){

  F’ = PLP(F);

  F’’ = BCP(F’);

  if (F’’ = ⊤) return true;

  if (F’’ = ⊥) return false;

  Choose a variable p in F’’;

  return SAT(F’’[⊤/p]) ∨ SAT(F’’[⊥/p]);

}
```

EXAMPLE

$$F : (\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$$

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[True/q] = $r \wedge \neg r \wedge (p \vee \neg r)$
- SAT(F[True/q])
  - After PLP: $r \wedge \neg r$
  - After BCP: False
  - Return False and backtrack to previous call

```
SAT(F){

  F' = PLP(F);

  F'' = BCP(F');

  if (F'' = ⊤) return true;

  if (F'' = ⊥) return false;

   Choose a variable p in
F'';

   return SAT(F''[⊤/p]) ∨
  SAT(F''[⊥/p]);
}
```

EXAMPLE

$$F : (\neg p \lor q \lor r) \land (\neg q \lor r) \land (\neg q \lor \neg r) \land (p \lor \neg q \lor \neg r)$$

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  -

```
SAT(F){
  F' = PLP(F);
  F'' = BCP(F');
  if (F'' = ⊤) return true;
  if (F'' = ⊥) return false;
  Choose a variable p in F'';
  return SAT(F''[⊤/p]) ∨ SAT(F''[⊥/p]);
}
```

EXAMPLE

$$F : (\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$$

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[False/q] = $\neg p \vee r$

```
SAT(F){

  F’ = PLP(F);

  F’’ = BCP(F’);

  if (F’’ = ⊤) return true;

  if (F’’ = ⊥) return false;

  Choose a variable p in
F’’;

  return SAT(F’’[⊤/p]) ∨
  SAT(F’’[⊥/p]);
}
```

EXAMPLE

$$F : (\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$$

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[False/q] = $\neg p \vee r$
- SAT(F[False/q])
  -

```
SAT(F){
  F' = PLP(F);
  F'' = BCP(F');
  if (F'' = ⊤) return true;
  if (F'' = ⊥) return false;
  Choose a variable p in F'';
  return SAT(F''[⊤/p]) ∨
  SAT(F''[⊥/p]);
}
```

EXAMPLE

$$F : (\neg p \vee q \vee r) \wedge (\neg q \vee r) \wedge (\neg q \vee \neg r) \wedge (p \vee \neg q \vee \neg r)$$

- SAT(F)
  - No PLP or BCP.
  - q ← CHOOSE.
  - F[False/q] = $\neg p \vee r$
- SAT(F[False/q])
  - After PLP: True
  - Satisfiable!

```
SAT(F){

  F' = PLP(F);

  F'' = BCP(F');

  if (F'' = ⊤) return true;

  if (F'' = ⊥) return false;

  Choose a variable p in F'';

  return SAT(F''[⊤/p]) ∨
  SAT(F''[⊥/p]);
}
```

DPLL IS JUST THE STARTING POINT!

- Modern SAT solvers use a variety of approaches to further improve performance
  - Non-chronological back tracking
  - Conflict-driven clause learning (CDCL)
  - Heuristics to CHOOSE appropriate variables and assignments
- Current SAT solvers can solve problems with millions of clauses in reasonable amount of time on average.

ENCODING PROBLEMS IN PL

- Even though PL is relatively straightforward, many problems in diverse areas can be encoded in PL.
    - Problems in graph theory and combinatorics, games such as Sudoku, problems in biotechnology and bioinformatics, etc.
    - There exists a reduction from every NP-Complete problem to SAT.
- As an example, let us try to encode the graph-colouring problem in PL.

GRAPH COLOURING IN PL

- In the graph colouring problem, the goal is to assign colours to vertices such that no two adjacent vertices have the same colour.
- Formally, consider graph $G = \langle V, E \rangle$
  - Vertices, $V = \{v_1, \ldots, v_n\}$
  - Edges, $E = \{e_1, \ldots, e_l\} \subseteq V \times V$
  - Colours, $C = \{c_1, \ldots, c_m\}$
- Assign each vertex $v \in V$ a color color$(v) \in C$ such that
  - for edge $e = (v, w) \in E$, color$(v) \neq$ color(w).

GRAPH COLOURING IN PL

- We use binary variable p_v^c to denote that vertex v has been assigned color c.
- Properties that the colouring should satisfy:
  - Each vertex must be coloured from the set C.
  - Each vertex must be assigned at most one colour.
  - Two adjacent vertices must be assigned different colours.

GRAPH COLOURING IN PL

- Each vertex must be coloured from the set C.

$$(p_{v_1}^{c_1} \vee p_{v_1}^{c_2} \vee \ldots \vee p_{v_1}^{c_m}) \wedge \ldots \wedge (p_{v_n}^{c_1} \vee p_{v_n}^{c_2} \vee \ldots \vee p_{v_n}^{c_m})$$

- Each vertex must be assigned at most one colour.

$$\bigwedge_{i=1}^{n} \bigvee_{1 \leq j < k \leq m} p_{v_i}^{c_j} \rightarrow \neg p_{v_i}^{c_k}$$

- Two adjacent vertices must be assigned different colours.

$$\bigwedge_{(v,v') \in E} \bigwedge_{k=1}^{m} \neg (p_{v}^{c_k} \wedge p_{v'}^{c_k})$$

GRAPH COLOURING IN PL

- An optimisation: We can omit the at-most one colour constraint.
  - This is because if there is a valid colouring which assigns more than one colour, then there is also a valid colouring assigning exactly one colour.
  - The original formula and the optimised formula are equisatisfiable.