

Mathematical Logic

Lecture 8: Low complexity subclasses of SAT

Ashutosh Gupta

TIFR, India

Compile date: 2015-09-07

Where are we and where are we going?

We have seen

- ▶ propositional logic
- ▶ proof methods for the logic
- ▶ soundness, completeness, and complexity of the methods

We will see

- ▶ Low complexity subclasses of SAT

Special classes of formulas

We will discuss the following subclasses whose SAT problems are polynomial

- ▶ 2-SAT
- ▶ Horn clauses
- ▶ XOR-SAT

Topic 8.1

2-SAT

2-SAT

Definition 8.1

A *2-sat formula* is a CNF formula that has *only* binary clauses

2-SAT

Definition 8.1

A *2-sat formula* is a CNF formula that has *only* binary clauses

We assume that unit clauses are replaced by clauses with repeated literals.

Example 8.1

- ▶ $(\neg p \vee q) \wedge (\neg q \vee r) \wedge (\neg r \vee p) \wedge (r \vee q)$ is a 2-SAT formula
- ▶ $(p \vee p) \wedge (\neg p \vee \neg p)$ is a 2-SAT formula

Implication graph

Definition 8.2

Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.

The *implication graph* (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\overline{\neg p} = p$.

Implication graph

Definition 8.2

Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.
The **implication graph** (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\overline{\neg p} = p$.

Example 8.2

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

p

$\neg p$

r

$\neg r$

Implication graph

Definition 8.2

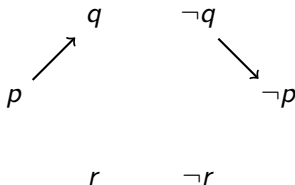
Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.
The **implication graph** (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\neg \bar{p} = p$.

Example 8.2

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



Implication graph

Definition 8.2

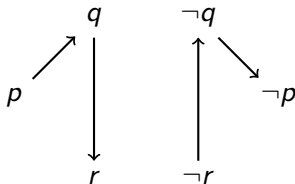
Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.
The **implication graph** (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\neg \bar{p} = p$.

Example 8.2

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



Implication graph

Definition 8.2

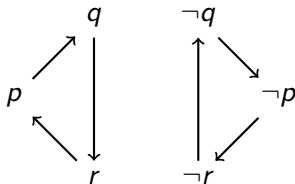
Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.
The **implication graph** (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\neg \bar{p} = p$.

Example 8.2

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



Implication graph

Definition 8.2

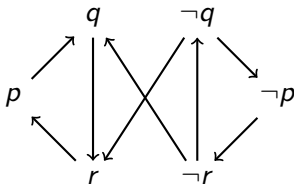
Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.
The **implication graph** (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\neg \bar{p} = p$.

Example 8.2

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



Implication graph

Definition 8.2

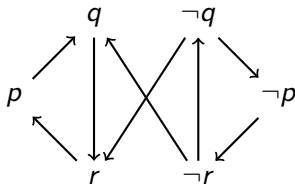
Let F be a 2-SAT formula s.t. $\mathbf{Vars}(F) = \{p_1, \dots, p_n\}$.
The **implication graph** (V, E) for F is defined as follows.

- ▶ $V = \{p_1, \dots, p_n, \neg p_1, \dots, \neg p_n\}$
- ▶ $E = \{(\bar{\ell}_1, \ell_2), (\bar{\ell}_2, \ell_1) \mid (\ell_1 \vee \ell_2) \in F\},$

where $\bar{p} = \neg p$ and $\neg \bar{p} = p$.

Example 8.2

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



Exercise 8.1

Draw implication graphs of the following

1. $(p \vee q) \wedge (\neg p \vee \neg q)$
2. $(p \vee \neg q) \wedge (q \vee p) \wedge (\neg p \vee \neg r) \wedge (r \vee \neg p)$
3. $(p \vee p) \wedge (\neg p \vee \neg p)$
4. $(p \vee \neg p) \wedge (p \vee \neg p)$

Properties of implication graph

Consider a formula F and its implication graph (V, E) .

Theorem 8.1

If there is a path from ℓ_1 to ℓ_2 in (V, E) then there is a path from $\bar{\ell}_2$ to $\bar{\ell}_1$.

Properties of implication graph

Consider a formula F and its implication graph (V, E) .

Theorem 8.1

If there is a path from ℓ_1 to ℓ_2 in (V, E) then there is a path from $\bar{\ell}_2$ to $\bar{\ell}_1$.

Exercise 8.2

a. Prove the above theorem.

Properties of implication graph

Consider a formula F and its implication graph (V, E) .

Theorem 8.1

If there is a path from ℓ_1 to ℓ_2 in (V, E) then there is a path from $\bar{\ell}_2$ to $\bar{\ell}_1$.

Exercise 8.2

a. *Prove the above theorem.*

b. *Does the above theorem imply*

if there is a path from p to $\neg p$ in (V, E) then there is a path from $\neg p$ to p ?

Properties of implication graph

Consider a formula F and its implication graph (V, E) .

Theorem 8.1

If there is a path from ℓ_1 to ℓ_2 in (V, E) then there is a path from $\bar{\ell}_2$ to $\bar{\ell}_1$.

Exercise 8.2

a. *Prove the above theorem.*

b. *Does the above theorem imply*

if there is a path from p to $\neg p$ in (V, E) then there is a path from $\neg p$ to p ?

Theorem 8.2

*For every strongly connected component (scc) $S \subseteq V$ in (V, E) , there is another scc S^c , called **complementary component**, that has exactly the set of literals that are negation of the literals in S .*

Properties of implication graph

Consider a formula F and its implication graph (V, E) .

Theorem 8.1

If there is a path from ℓ_1 to ℓ_2 in (V, E) then there is a path from $\bar{\ell}_2$ to $\bar{\ell}_1$.

Exercise 8.2

a. *Prove the above theorem.*

b. *Does the above theorem imply*

if there is a path from p to $\neg p$ in (V, E) then there is a path from $\neg p$ to p ?

Theorem 8.2

*For every strongly connected component(scc) $S \subseteq V$ in (V, E) , there is another scc S^c , called **complementary component**, that has exactly the set of literals that are negation of the literals in S .*

Proof.

Due to theorem 8.1.



Properties of implication graph (contd.)

Theorem 8.3

For each model $m \models F$, if there is a path from ℓ_1 to ℓ_2 in (V, E) then if $m(\ell_1) = 1$ then $m(\ell_2) = 1$.

Properties of implication graph (contd.)

Theorem 8.3

For each model $m \models F$, if there is a path from ℓ_1 to ℓ_2 in (V, E) then if $m(\ell_1) = 1$ then $m(\ell_2) = 1$.

Theorem 8.4

For each model $m \models F$ and each scc S in (V, E) , either for each $\ell \in S$ $m(\ell) = 1$ or for each $\ell \in S$ $m(\ell) = 0$.

Properties of implication graph (contd.)

Theorem 8.3

For each model $m \models F$, if there is a path from ℓ_1 to ℓ_2 in (V, E) then if $m(\ell_1) = 1$ then $m(\ell_2) = 1$.

Theorem 8.4

For each model $m \models F$ and each scc S in (V, E) , either for each $\ell \in S$ $m(\ell) = 1$ or for each $\ell \in S$ $m(\ell) = 0$.

Exercise 8.3

Prove the above theorems.

Reduced implication graph

Definition 8.3

The *reduced implication DAG* (V^R, E^R) is a graph over scc's of (V, E) and is defined as follows.

- ▶ $V^R = \{S \mid S \text{ is a scc in } (V, E)\}$
- ▶ $E^R = \{(S, S') \mid \text{there are } \ell \in S \text{ and } \ell' \in S' \text{ s.t. } (\ell, \ell') \in E\}$

Reduced implication graph

Definition 8.3

The *reduced implication DAG* (V^R, E^R) is a graph over scc's of (V, E) and is defined as follows.

- ▶ $V^R = \{S \mid S \text{ is a scc in } (V, E)\}$
- ▶ $E^R = \{(S, S') \mid \text{there are } \ell \in S \text{ and } \ell' \in S' \text{ s.t. } (\ell, \ell') \in E\}$

Theorem 8.5

If $(S, S') \in E^R$ then $(S'^c, S^c) \in E^R$

Reduced implication graph

Definition 8.3

The *reduced implication DAG* (V^R, E^R) is a graph over scc's of (V, E) and is defined as follows.

- ▶ $V^R = \{S \mid S \text{ is a scc in } (V, E)\}$
- ▶ $E^R = \{(S, S') \mid \text{there are } \ell \in S \text{ and } \ell' \in S' \text{ s.t. } (\ell, \ell') \in E\}$

Theorem 8.5

If $(S, S') \in E^R$ then $(S'^c, S^c) \in E^R$

Exercise 8.4

Prove the above theorem.

2-SAT satisfiability

Theorem 8.6

A 2-SAT formula F is unsat iff there is a scc S in its implication graph (V, E) such that $\{p, \neg p\} \subseteq S$ for some p .

2-SAT satisfiability

Theorem 8.6

A 2-SAT formula F is unsat iff there is a scc S in its implication graph (V, E) such that $\{p, \neg p\} \subseteq S$ for some p .

Proof.

Reverse direction

2-SAT satisfiability

Theorem 8.6

A 2-SAT formula F is unsat iff there is a scc S in its implication graph (V, E) such that $\{p, \neg p\} \subseteq S$ for some p .

Proof.

Reverse direction

There must be a path that goes from p to $\neg p$.

$$p \longrightarrow \ell_1 \text{ ----- } \ell_n \longrightarrow \neg p$$

2-SAT satisfiability

Theorem 8.6

A 2-SAT formula F is unsat iff there is a scc S in its implication graph (V, E) such that $\{p, \neg p\} \subseteq S$ for some p .

Proof.

Reverse direction

There must be a path that goes from p to $\neg p$.

$$p \longrightarrow \ell_1 \text{ -----} \rightarrow \ell_n \longrightarrow \neg p$$

By applying resolution on corresponding clauses, we can derive $\neg p$.

$$\frac{(\neg p \vee \ell_1) \quad (\neg \ell_1 \vee \ell_2) \quad \dots \quad (\neg \ell_{n-1} \vee \ell_n) \quad (\neg \ell_n \vee p)}{\neg p}$$

2-SAT satisfiability

Theorem 8.6

A 2-SAT formula F is unsat iff there is a scc S in its implication graph (V, E) such that $\{p, \neg p\} \subseteq S$ for some p .

Proof.

Reverse direction

There must be a path that goes from p to $\neg p$.

$$p \longrightarrow \ell_1 \text{ ----- } \ell_n \longrightarrow \neg p$$

By applying resolution on corresponding clauses, we can derive $\neg p$.

$$\frac{(\neg p \vee \ell_1) \quad (\neg \ell_1 \vee \ell_2) \quad \dots \quad (\neg \ell_{n-1} \vee \ell_n) \quad (\neg \ell_n \vee p)}{\neg p}$$

Similarly, we can derive p by the path from $\neg p$ to p .

Therefore, we can derive empty clause.

F is unsat.

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

We need to show that at step 3 we always find S .

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

We need to show that at step 3 we always find S .

claim: at step 3, there is a node whose all children are assigned.

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

We need to show that at step 3 we always find S .

claim: at step 3, there is a node whose all children are assigned.

Choose an unassigned node and descend down if there is unassigned child.

Since the DAG is finite, termination.

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

We need to show that at step 3 we always find S .

claim: at step 3, there is a node whose all children are assigned.

Choose an unassigned node and descend down if there is unassigned child.

Since the DAG is finite, termination.

claim: an unassigned node can not have a child that is assigned 0.

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

We need to show that at step 3 we always find S .

claim: at step 3, there is a node whose all children are assigned.

Choose an unassigned node and descend down if there is unassigned child.

Since the DAG is finite, termination.

claim: an unassigned node can not have a child that is assigned 0.

If S is assigned 1, all its children are already 1. Therefore, all the parents of S^c are already assigned 0(due to theorem 8.5). □

2-SAT satisfiability(contd.)

Proof(contd.)

Fwd direction: Let us assume there is no such S .

We will construct a model of F as follows.

1. Initially all literals are unassigned.
2. if(some scc in V^R is unassigned)
3. Let $S \in V^R$ be an unassigned scc whose all children are assigned 1.
4. Assign literals of S to 1. Consequently, S^c is assigned 0.
5. goto 2.

We need to show that at step 3 we always find S .

claim: at step 3, there is a node whose all children are assigned.

Choose an unassigned node and descend down if there is unassigned child.

Since the DAG is finite, termination.

claim: an unassigned node can not have a child that is assigned 0.

If S is assigned 1, all its children are already 1. Therefore, all the parents of S^c are already assigned 0(due to theorem 8.5). □

Exercise 8.5

Show the above procedure produces a satisfying model.

2-SAT is polynomial

Theorem 8.7

A 2-SAT satisfiability problem can be solved in linear time.

Proof.

Due to the previous theorem, 2-SAT satisfiability problem is polynomial. □

Practice 2-SAT solving

Exercise 8.6

Find a satisfying assignment of the following formula

1. $(\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (\neg z \vee \neg x) \wedge (x \vee \neg w) \wedge (y \vee \neg w) \wedge (z \vee \neg w)$
2. $(p_0 \vee p_2) \wedge (p_0 \vee \neg p_3) \wedge (p_1 \vee \neg p_3) \wedge (p_1 \vee \neg p_4) \wedge (p_2 \vee \neg p_4) \wedge$
 $(p_0 \vee \neg p_5) \wedge (p_1 \vee \neg p_5) \wedge (p_2 \vee \neg p_5) \wedge (p_3 \vee p_6) \wedge (p_4 \vee p_6) \wedge (p_5 \vee p_6)$

Topic 8.2

Horn Clauses

Horn clauses

Definition 8.4

A *Horn clause* is a clause that has the following form

$$\neg p_1 \vee \cdots \vee \neg p_n \vee q,$$

where $p_1, \dots, p_n \in \mathbf{Vars}$, and $q \in \mathbf{Vars} \cup \{\perp\}$.

A *Horn formula* is a set of Horn clauses, which is interpreted as conjunction of the Horn clauses.

The clauses with $\perp$ literals are called *goal clauses* and others are called *implication clauses*.

Horn clauses

Definition 8.4

A *Horn clause* is a clause that has the following form

$$\neg p_1 \vee \cdots \vee \neg p_n \vee q,$$

where $p_1, \dots, p_n \in \mathbf{Vars}$, and $q \in \mathbf{Vars} \cup \{\perp\}$.

A *Horn formula* is a set of Horn clauses, which is interpreted as conjunction of the Horn clauses.

The clauses with $\perp$ literals are called *goal clauses* and others are called *implication clauses*.

Example 8.3

The following set is a Horn formula

$$\begin{aligned} \{ & p, \quad \neg q \vee \neg r \vee \neg t \vee p, \\ & \neg p \vee q, \quad \neg p \vee \neg r \vee t, \\ & \neg p \vee \neg q \vee t, \quad \neg r \vee \perp, \\ & \neg p \vee \neg q \vee \neg t \vee \perp \} \end{aligned}$$

Implication view of the horn clauses

We may view a Horn clause

$$\neg p_1 \vee \cdots \vee \neg p_n \vee q$$

as

$$p_1 \wedge \cdots \wedge p_n \Rightarrow q.$$

Example 8.4

The following is an implication view of a Horn formula

$$\begin{aligned} \{ & \top \Rightarrow p, \quad q \wedge r \wedge t \Rightarrow p, \\ & p \Rightarrow q, \quad p \wedge r \Rightarrow t, \\ & \neg p \wedge q \Rightarrow t, \quad r \Rightarrow \perp, \\ & p \wedge q \wedge t \Rightarrow \perp \} \end{aligned}$$

Note $\top \Rightarrow p$ means p , which is a Horn clause without negative literals

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

$m = \lambda x.0;$

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

$m = \lambda x.0;$

while $m \not\models (p_1 \wedge \dots \wedge p_n \Rightarrow p) \in Hs$ **do**

|

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

$m = \lambda x.0$;

while $m \not\models (p_1 \wedge \dots \wedge p_n \Rightarrow p) \in Hs$ **do**

$m := m[p \mapsto 1]$;

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

$m = \lambda x.0$;

while $m \not\models (p_1 \wedge \dots \wedge p_n \Rightarrow p) \in Hs$ **do**

$m := m[p \mapsto 1]$;

if $m \not\models (q_1 \wedge \dots \wedge q_k \Rightarrow \perp) \in Gs$ **then return** *unsat* ;

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

$m = \lambda x.0$;

while $m \not\models (p_1 \wedge \dots \wedge p_n \Rightarrow p) \in Hs$ **do**

$m := m[p \mapsto 1]$;

if $m \not\models (q_1 \wedge \dots \wedge q_k \Rightarrow \perp) \in Gs$ **then return** *unsat* ;

return m

Horn Satisfiability

Algorithm 8.1: HORNSAT(Hs, Gs)

Input: Hs : implication clauses, Gs : goal clauses

Output: model/unsat

$m = \lambda x.0$;

while $m \not\models (p_1 \wedge \dots \wedge p_n \Rightarrow p) \in Hs$ **do**

$m := m[p \mapsto 1]$;

if $m \not\models (q_1 \wedge \dots \wedge q_k \Rightarrow \perp) \in Gs$ **then return** *unsat* ;

return m

Exercise 8.7

Solve

$$\{\top \Rightarrow p, \quad q \wedge r \wedge t \Rightarrow p, \quad p \Rightarrow q, \quad p \wedge r \Rightarrow t, \\ \neg p \wedge q \Rightarrow t, \quad r \Rightarrow \perp, \quad p \wedge q \wedge t \Rightarrow \perp\}$$

Recognizing Horn clauses

Sometimes a set of clauses are not immediately recognizable as Horn clause.

Recognizing Horn clauses

Sometimes a set of clauses are not immediately recognizable as Horn clause.

We may convert a CNF into a Horn formula by flipping the negation sign for some variables. Such CNF are called Horn clause renameable.

Recognizing Horn clauses

Sometimes a set of clauses are not immediately recognizable as Horn clause.

We may convert a CNF into a Horn formula by flipping the negation sign for some variables. Such CNF are called Horn clause renameable.

Definition 8.5

Let F be a CNF formula and m be a model. Let $\text{flip}(F, m)$ denote the formula obtained by flipping the variables that are assigned 1 in m .

Recognizing Horn clauses

Sometimes a set of clauses are not immediately recognizable as Horn clause.

We may convert a CNF into a Horn formula by flipping the negation sign for some variables. Such CNF are called Horn clause renameable.

Definition 8.5

Let F be a CNF formula and m be a model. Let $\text{flip}(F, m)$ denote the formula obtained by flipping the variables that are assigned 1 in m .

Example 8.5

$$\text{flip}((p \vee \neg q \vee \neg s), [p \mapsto 1, q \mapsto 0, s \mapsto 1, \dots]) = (\neg p \vee \neg q \vee s)$$

Renaming Horn clauses

Theorem 8.8

*A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.**

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978.

Renaming Horn clauses

Theorem 8.8

A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.*

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

Proof.

Forward direction: there is a model m such that $\text{flip}(F, m)$ is a Horn formula

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978.

Renaming Horn clauses

Theorem 8.8

A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.*

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

Proof.

Forward direction: there is a model m such that $\text{flip}(F, m)$ is a Horn formula

claim: $m \models G$

consider a clause $\ell_{ij} \vee \ell_{ik} \in G$

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978.

Renaming Horn clauses

Theorem 8.8

A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.*

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

Proof.

Forward direction: there is a model m such that $\text{flip}(F, m)$ is a Horn formula

claim: $m \models G$

consider a clause $\ell_{ij} \vee \ell_{ik} \in G$

case $\ell_{ij} = p, \ell_{ik} = q$: one of them must flip, i.e., $m(p) = 1$ or $m(q) = 1$

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978.

Renaming Horn clauses

Theorem 8.8

A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.*

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

Proof.

Forward direction: there is a model m such that $\text{flip}(F, m)$ is a Horn formula

claim: $m \models G$

consider a clause $\ell_{ij} \vee \ell_{ik} \in G$

case $\ell_{ij} = p, \ell_{ik} = q$: one of them must flip, i.e., $m(p) = 1$ or $m(q) = 1$

case $\ell_{ij} = \neg p, \ell_{ik} = \neg q$: at least one must not flip, i.e., not $m(p) = m(q) = 1$

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978. 19

Renaming Horn clauses

Theorem 8.8

A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.*

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

Proof.

Forward direction: there is a model m such that $\text{flip}(F, m)$ is a Horn formula

claim: $m \models G$

consider a clause $\ell_{ij} \vee \ell_{ik} \in G$

case $\ell_{ij} = p, \ell_{ik} = q$: one of them must flip, i.e., $m(p) = 1$ or $m(q) = 1$

case $\ell_{ij} = \neg p, \ell_{ik} = \neg q$: at least one must not flip, i.e., not $m(p) = m(q) = 1$

case $\ell_{ij} = \neg p, \ell_{ik} = q$: if p flips then q must, i.e., if $m(p) = 1$ then $m(q) = 1$

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978. 19

Renaming Horn clauses

Theorem 8.8

A CNF formula $F = \{C_1, \dots, C_n\}$, where $C_i = \{\ell_{i1}, \dots, \ell_{il_i}\}$ is Horn clause renameable iff the following 2-SAT formula is satisfiable.*

$$G = \{\ell_{ij} \vee \ell_{ik} \mid i \in 1..n \text{ and } 1 \leq j < k \leq l_i\}$$

Proof.

Forward direction: there is a model m such that $\text{flip}(F, m)$ is a Horn formula

claim: $m \models G$

consider a clause $\ell_{ij} \vee \ell_{ik} \in G$

case $\ell_{ij} = p, \ell_{ik} = q$: one of them must flip, i.e., $m(p) = 1$ or $m(q) = 1$

case $\ell_{ij} = \neg p, \ell_{ik} = \neg q$: at least one must not flip, i.e., not $m(p) = m(q) = 1$

case $\ell_{ij} = \neg p, \ell_{ik} = q$: if p flips then q must, i.e., if $m(p) = 1$ then $m(q) = 1$

In all the three cases $m \models \ell_{ij} \vee \ell_{ik}$.

*H. Lewis. Renaming a Set of Clauses as a Horn Set. J. of the ACM, 25:134-135, 1978. 19

Renaming Horn clauses(contd.)

Proof(contd.)

Reverse direction: Let $m \models G$. Let $F' = \text{flip}(F, m)$.

Renaming Horn clauses(contd.)

Proof(contd.)

Reverse direction: Let $m \models G$. Let $F' = \text{flip}(F, m)$.

claim: F' is a Horn formula

Suppose F' is not a Horn formula

Renaming Horn clauses(contd.)

Proof(contd.)

Reverse direction: Let $m \models G$. Let $F' = \text{flip}(F, m)$.

claim: F' is a Horn formula

Suppose F' is not a Horn formula

Then, there are positive literals ℓ'_{ij} and ℓ'_{ik} in F' .

Renaming Horn clauses(contd.)

Proof(contd.)

Reverse direction: Let $m \models G$. Let $F' = \text{flip}(F, m)$.

claim: F' is a Horn formula

Suppose F' is not a Horn formula

Then, there are positive literals ℓ'_{ij} and ℓ'_{ik} in F' .

Therefore, $m \not\models \ell_{ij} \vee \ell_{ik}$ (why?). **Contradiction.**



Exercise 8.8

What is the complexity of checking if a formula is Horn clause renameable?

Topic 8.3

XOR SAT

XOR-SAT

Definition 8.6

*A formula is **XOR-SAT** if it is a conjunction of xors of literals.*

XOR-SAT

Definition 8.6

A formula is **XOR-SAT** if it is a conjunction of xors of literals.

Example 8.6

$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$
is a XOR-SAT formula.

Solving XOR-SAT

Since xors are negation of equality, we may eliminate variables via substitution.

Solving XOR-SAT

Since xors are negation of equality, we may eliminate variables via substitution.

Theorem 8.9

*For a variable, p , xor formula G , and XOR-SAT formula F ,
 $(p \oplus G) \wedge F$ is sat iff $F[\neg G/p]$ is sat*

Solving XOR-SAT

Since xors are negation of equality, we may eliminate variables via substitution.

Theorem 8.9

*For a variable, p , xor formula G , and XOR-SAT formula F ,
 $(p \oplus G) \wedge F$ is sat iff $F[\neg G/p]$ is sat*

Exercise 8.9

Prove the above theorem.

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p :

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p :

Due to the first xor: $p \Leftrightarrow \neg r \oplus s$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p :

Due to the first xor: $p \Leftrightarrow \neg r \oplus s$

After substitution: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p :

Due to the first xor: $p \Leftrightarrow \neg r \oplus s$

After substitution: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$

Simplification: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p :

Due to the first xor: $p \Leftrightarrow \neg r \oplus s$

After substitution: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$

Simplification: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$

Eliminate r :

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

Due to the first xor: $p \Leftrightarrow \neg r \oplus s$

After substitution: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$

Simplification: $(q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$

Eliminate r:

Due to the second xor: $r \Leftrightarrow \neg q$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

$$\text{Due to the first xor: } p \Leftrightarrow \neg r \oplus s$$

$$\text{After substitution: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$$

$$\text{Simplification: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$$

Eliminate r:

$$\text{Due to the second xor: } r \Leftrightarrow \neg q$$

$$\text{After substitution: } (q \oplus \neg \neg q \oplus s) \wedge (s \oplus \neg q)$$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

$$\text{Due to the first xor: } p \Leftrightarrow \neg r \oplus s$$

$$\text{After substitution: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$$

$$\text{Simplification: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$$

Eliminate r:

$$\text{Due to the second xor: } r \Leftrightarrow \neg q$$

$$\text{After substitution: } (q \oplus \neg \neg q \oplus s) \wedge (s \oplus \neg q)$$

$$\text{Simplification: } s \wedge (s \oplus \neg q)$$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

$$\text{Due to the first xor: } p \Leftrightarrow \neg r \oplus s$$

$$\text{After substitution: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$$

$$\text{Simplification: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$$

Eliminate r:

$$\text{Due to the second xor: } r \Leftrightarrow \neg q$$

$$\text{After substitution: } (q \oplus \neg \neg q \oplus s) \wedge (s \oplus \neg q)$$

$$\text{Simplification: } s \wedge (s \oplus \neg q)$$

Eliminate q:

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

$$\text{Due to the first xor: } p \Leftrightarrow \neg r \oplus s$$

$$\text{After substitution: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$$

$$\text{Simplification: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$$

Eliminate r:

$$\text{Due to the second xor: } r \Leftrightarrow \neg q$$

$$\text{After substitution: } (q \oplus \neg \neg q \oplus s) \wedge (s \oplus \neg q)$$

$$\text{Simplification: } s \wedge (s \oplus \neg q)$$

Eliminate q:

$$\text{Due to the second xor: } q \Leftrightarrow s$$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

$$\text{Due to the first xor: } p \Leftrightarrow \neg r \oplus s$$

$$\text{After substitution: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$$

$$\text{Simplification: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$$

Eliminate r:

$$\text{Due to the second xor: } r \Leftrightarrow \neg q$$

$$\text{After substitution: } (q \oplus \neg \neg q \oplus s) \wedge (s \oplus \neg q)$$

$$\text{Simplification: } s \wedge (s \oplus \neg q)$$

Eliminate q:

$$\text{Due to the second xor: } q \Leftrightarrow s$$

$$\text{After substitution: } s$$

Example : solving XOR-SAT

Example 8.7

$$(p \oplus r \oplus s) \wedge (q \oplus \neg r \oplus s) \wedge (p \oplus q \oplus \neg s) \wedge (p \oplus \neg q \oplus \neg r)$$

Eliminate p:

$$\text{Due to the first xor: } p \Leftrightarrow \neg r \oplus s$$

$$\text{After substitution: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus s \oplus q \oplus \neg s) \wedge (\neg r \oplus s \oplus \neg q \oplus \neg r)$$

$$\text{Simplification: } (q \oplus \neg r \oplus s) \wedge (\neg r \oplus \neg q) \wedge (s \oplus \neg q)$$

Eliminate r:

$$\text{Due to the second xor: } r \Leftrightarrow \neg q$$

$$\text{After substitution: } (q \oplus \neg \neg q \oplus s) \wedge (s \oplus \neg q)$$

$$\text{Simplification: } s \wedge (s \oplus \neg q)$$

Eliminate q:

$$\text{Due to the second xor: } q \Leftrightarrow s$$

After substitution: s

Solution:

$$m(s) = 1$$

$$m(q) = m(s) = 1$$

$$m(r) = m(\neg q) = 0$$

$$m(p) = m(\neg r \oplus s) = 0$$

End of Lecture 8