CS 228 : Logic in Computer Science

Krishna. S

Checking Satisfiability

- The SAT problem : Given a formula φ, is it satisfiable?
- Given a formula φ, is it valid?
- SAT is NP-complete. NP represents non-deterministic polynomial time.
- Given a witness, it is easy to check if the witness is a valid witness in polynomial time, Finding a witness is not as easy.
  - Given a valuation α for the variables $p_1, \ldots, p_n$ of a formula φ, we can check if φ evaluates to true under α in time polynomial in n.
  - Finding whether such a valuation α exists is not as easy.
- SAT solvers are tools which implement heuristics to check satisfiability.
- The input to a SAT solver is a formula in some specified form.

Normal Forms

- A literal is a propositional variable p or its negation $\neg p$. These are referred to as positive and negative literals respectively.

Normal Forms

- A literal is a propositional variable p or its negation $\neg p$. These are referred to as positive and negative literals respectively.
- A formula F is in Conjunctive Normal Form (CNF) if it is a conjunction of a disjunction of literals.

$$F = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} L_{i,j}$$

each $L_{i,j}$ is a literal.

Normal Forms

- A literal is a propositional variable p or its negation $\neg p$. These are referred to as positive and negative literals respectively.
- A formula F is in Conjunctive Normal Form (CNF) if it is a conjunction of a disjunction of literals.

$$F = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} L_{i,j}$$

  each $L_{i,j}$ is a literal.
- A formula F is in DNF if it is a disjunction of a conjunction of literals.

$$F = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} L_{i,j}$$

  each $L_{i,j}$ is a literal.

Normal Forms

In the following, equivalent stands for semantically equivalent

Let F be a formula in CNF and let G be a formula in DNF. Then $\neg F$ is equivalent to a formula in DNF and $\neg G$ is equivalent to a formula in CNF.

Normal Forms

In the following, equivalent stands for semantically equivalent

Let F be a formula in CNF and let G be a formula in DNF. Then $\neg F$ is equivalent to a formula in DNF and $\neg G$ is equivalent to a formula in CNF.

Every formula F is equivalent to some formula F_1 in CNF and some formula F_2 in DNF.

CNF Algorithm

Given a formula F, $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

CNF Algorithm

Given a formula F, $(x \to [\neg(y \vee z) \wedge \neg(y \to x)])$

- Replace all subformulae of the form $F \to G$ with $\neg F \vee G$, and all subformulae of the form $F \leftrightarrow G$ with $(\neg F \vee G) \wedge (\neg G \vee F)$. When there are no more occurrences of $\to, \leftrightarrow$, proceed to the next step.

CNF Algorithm

Given a formula F, $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

- Replace all subformulae of the form $F \rightarrow G$ with $\neg F \vee G$, and all subformulae of the form $F \leftrightarrow G$ with $(\neg F \vee G) \wedge (\neg G \vee F)$. When there are no more occurrences of $\rightarrow, \leftrightarrow$, proceed to the next step.
- Get rid of all double negations, and push all negations inside to the level of literals : Replace all subformulae
  - $\neg\neg G$ with G,
  - $\neg(G \wedge H)$ with $\neg G \vee \neg H$
  - $\neg(G \vee H)$ with $\neg G \wedge \neg H$

  When there are no more such subformulae, proceed to the next step.

CNF Algorithm

Given a formula F, $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

- Replace all subformulae of the form $F \rightarrow G$ with $\neg F \vee G$, and all subformulae of the form $F \leftrightarrow G$ with $(\neg F \vee G) \wedge (\neg G \vee F)$. When there are no more occurrences of $\rightarrow, \leftrightarrow$, proceed to the next step.
- Get rid of all double negations, and push all negations inside to the level of literals : Replace all subformulae
  - $\neg\neg G$ with G,
  - $\neg(G \wedge H)$ with $\neg G \vee \neg H$
  - $\neg(G \vee H)$ with $\neg G \wedge \neg H$

  When there are no more such subformulae, proceed to the next step.
- Distribute $\vee$ wherever possible : that is, replace all $(F \vee (G \wedge H))$ or $((G \wedge H) \vee F)$ with $(F \vee G) \wedge (F \vee H)$.

The resultant formula F_1 is in CNF and is provably equivalent to F.
$[(\neg x \vee \neg y) \wedge (\neg x \vee \neg z)] \wedge [(\neg x \vee y) \wedge (\neg x \vee \neg x)]$

Polynomial Time Formula Classes

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.
- $p \wedge (\neg p \vee \neg q \vee r) \wedge (\neg a \vee \neg b)$ is Horn, but $a \vee b$ is not Horn.

Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- Programming languages Prolog and Datalog are based on Horn clauses in first order logic
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.
- $p \wedge (\neg p \vee \neg q \vee r) \wedge (\neg a \vee \neg b)$ is Horn, but $a \vee b$ is not Horn.
- A basic Horn formula is one which has no $\wedge$. Every Horn formula is a conjunction of basic Horn formulae.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.
- Basic Horn with no positive literals are written as $p \wedge q \wedge \cdots \wedge r \rightarrow \bot$.

Horn Formulae

- Three types of basic Horn : no positive literals, no negative literals, have both positive and negative literals.
- Basic Horn with both positive and negative literals are written as an implication $p \wedge q \wedge \cdots \wedge r \rightarrow s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \rightarrow p$.
- Basic Horn with no positive literals are written as $p \wedge q \wedge \cdots \wedge r \rightarrow \perp$.
- Thus, a Horn formula is written as a conjunction of implications.

A Decision Problem

Horn SAT : The Horn Satisfiability Problem

Given a Horn formula, is it satisfiable?

A Decision Problem

Horn SAT : The Horn Satisfiability Problem

Given a Horn formula, is it satisfiable?

The class **P**

An algorithm is polynomial time if there exists a polynomial $p(x)$ such that given the input size n, the algorithm terminates with the correct answer in $\leqslant p(n)$ steps. The class of all problems which can be solved by a polynomial time algorithm is denoted **P**.

Horn SAT is in **P**

The Horn Algorithm

Given a Horn formula H,

- Mark all occurrences of p, whenever $\top \to p$ is a subformula.

The Horn Algorithm

Given a Horn formula H,

- Mark all occurrences of p, whenever $\top \to p$ is a subformula.
- If there is a subformula of the form $(p_1 \wedge \cdots \wedge p_m) \to q$, where each p_i is marked, and q is not marked, mark q. Repeat this until there are no subformulae of this form and proceed to the next step.

The Horn Algorithm

Given a Horn formula H,

- Mark all occurrences of p, whenever $\top \to p$ is a subformula.
- If there is a subformula of the form $(p_1 \wedge \cdots \wedge p_m) \to q$, where each p_i is marked, and q is not marked, mark q. Repeat this until there are no subformulae of this form and proceed to the next step.
- Consider subformulae of the form $(p_1 \wedge \cdots \wedge p_m) \to \bot$. If there is one such subformula with all p_i marked, then say Unsat, otherwise say Sat.

An Example

$(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$

An Example

$(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$

- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$

An Example

$(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$

- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B).$

An Example

$(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.

- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.
- $(\top \to A) \wedge (C \to D) \wedge ((A \wedge B) \to C) \wedge ((C \wedge D) \to \bot) \wedge (\top \to B)$.

An Example

$(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$

- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$
- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$
- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$
- $(\top \to A) \land (C \to D) \land ((A \land B) \to C) \land ((C \land D) \to \bot) \land (\top \to B).$

The Horn Algorithm

The Horn algorithm concludes Sat iff H is satisfiable.

Complexity of Horn

- Given a Horn formula of length n, Horn SAT takes at most n^2 steps to conclude.
- Read once marking all $\top \to p$ clauses.
- Inspect and mark all clauses $(p_1 \wedge \ldots p_j) \to Q$ (at most n times)
- Check one final time

2-CNF

- 2-CNF : CNF where each clause has at most 2 literals.

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

Discussion on Consistency