CS 228 : Logic in Computer Science

Krishna. S

Summary

- Proof Rules
- Soundness : If $\varphi \vdash \psi$, then $\varphi \models \psi$.
- Completeness : If $\varphi \models \psi$, then $\varphi \vdash \psi$.
- Normal Forms

Satisfiability Checking : Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.

Satisfiability Checking : Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- A formula F is a Horn formula if it is in CNF and every disjunction contains atmost one positive literal.

Satisfiability Checking : Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- 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.

Satisfiability Checking : Horn Formulae

- A Horn Formula is a particularly nice kind of CNF formula, which can be quickly checked for satisfiability.
- 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.

Satisfiability Checking : Horn Formulae

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

Satisfiability Checking : 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.

Satisfiability Checking : 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 $\top \rightarrow p$.

Satisfiability Checking : 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 $p \wedge q \wedge \cdots \wedge r \to s$ involving only positive literals.
- Basic Horn with no negative literals are of the form p and are written as $\top \to p$.
- Basic Horn with no positive literals are written as $p \wedge q \wedge \cdots \wedge r \to \bot$.

Satisfiability Checking : 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 $p \wedge q \wedge \cdots \wedge r \rightarrow \bot$.
- Thus, a Horn formula is written as a conjunction of implications.

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

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

An Example

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

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

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.

The Horn Algorithm

The Horn algorithm concludes Sat iff H is satisfiable.

- Let $\mathcal{S} = \{p_1, \ldots, p_n\}$ be the set of propositions occurring in H. At the end of the algorithm, some of these are marked.

The Horn Algorithm

The Horn algorithm concludes Sat iff H is satisfiable.

- Let $\mathcal{S} = \{p_1, \ldots, p_n\}$ be the set of propositions occurring in H. At the end of the algorithm, some of these are marked.
- Assume H is satisfiable. Then there is an assignment α of $\mathcal{S}$ such that $\alpha \models H$. For each basic Horn formula B of H, $\alpha(B) = 1$. Also, $\alpha(\bot) = 0$ and $\alpha(\top) = 1$.

The Horn Algorithm

The Horn algorithm concludes Sat iff H is satisfiable.

- Let $\mathcal{S} = \{p_1, \ldots, p_n\}$ be the set of propositions occurring in H. At the end of the algorithm, some of these are marked.
- Assume H is satisfiable. Then there is an assignment α of $\mathcal{S}$ such that $\alpha \models H$. For each basic Horn formula B of H, $\alpha(B) = 1$. Also, $\alpha(\bot) = 0$ and $\alpha(\top) = 1$.
- If B has the form $\top \to p_i$, then $\alpha(p_i) = 1$. If B has the form $(p_1 \wedge \cdots \wedge p_n) \to q$, where each $\alpha(p_i) = 1$, then $\alpha(q) = 1$. Hence, $\alpha(p_i)$ agrees with the marking of the algo.

The Horn Algorithm

- Will the algo say unsat?

The Horn Algorithm

- Will the algo say unsat?
- If it does, there is some $(p_1 \wedge \ldots p_m) \rightarrow \perp$, where all p_i's are marked.

The Horn Algorithm

- Will the algo say unsat?
- If it does, there is some $(p_1 \wedge \ldots p_m) \rightarrow \bot$, where all p_i's are marked.
- Then $\alpha(p_i) = 1$ for all i and $\alpha((p_1 \wedge \ldots p_m) \rightarrow \bot) = 0$, a contradiction to the assumption about α.

The Horn Algorithm

- Will the algo say unsat?
- If it does, there is some $(p_1 \wedge \ldots p_m) \rightarrow \perp$, where all p_i's are marked.
- Then $\alpha(p_i) = 1$ for all i and $\alpha((p_1 \wedge \ldots p_m) \rightarrow \perp) = 0$, a contradiction to the assumption about α.
- Conversely, assume that the algo says *Sat*. Show that there exists a satisfying assignment α, using the markings made by the algo.

The Horn Algorithm

- Will the algo say unsat?
- If it does, there is some $(p_1 \wedge \ldots p_m) \rightarrow \perp$, where all p_i's are marked.
- Then $\alpha(p_i) = 1$ for all i and $\alpha((p_1 \wedge \ldots p_m) \rightarrow \perp) = 0$, a contradiction to the assumption about α.
- Conversely, assume that the algo says *Sat*. Show that there exists a satisfying assignment α, using the markings made by the algo. Let α be the assignment of $\mathcal{S}$ defined by $\alpha(p_i) = 1$ iff p_i is marked. We claim that $\alpha \models H$.

The Horn Algorithm

- Will the algo say unsat?
- If it does, there is some $(p_1 \wedge \ldots p_m) \to \bot$, where all p_i's are marked.
- Then $\alpha(p_i) = 1$ for all i and $\alpha((p_1 \wedge \ldots p_m) \to \bot) = 0$, a contradiction to the assumption about α.
- Conversely, assume that the algo says *Sat*. Show that there exists a satisfying assignment α, using the markings made by the algo. Let α be the assignment of $\mathcal{S}$ defined by $\alpha(p_i) = 1$ iff p_i is marked. We claim that $\alpha \models H$.
- Show that $\alpha \models B$ for each basic Horn formula B of H.

The Horn Algorithm

- If B has the form $\top \to p$, then p is marked in step 1 of the algo, and so $\alpha(B) = 1$.

The Horn Algorithm

- If B has the form $\top \to p$, then p is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $p_1 \wedge \ldots p_m \to q$, then q is either $\bot$ or an atomic formula.

The Horn Algorithm

- If B has the form $\top \to p$, then p is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $p_1 \wedge \ldots p_m \to q$, then q is either $\perp$ or an atomic formula.
- If some p_i was not marked, then $\alpha(p_i) = 0$, and hence $\alpha(B) = 1$.

The Horn Algorithm

- If B has the form $\top \to p$, then p is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $p_1 \wedge \dots p_m \to q$, then q is either $\perp$ or an atomic formula.
- If some p_i was not marked, then $\alpha(p_i) = 0$, and hence $\alpha(B) = 1$.
- Assume all the p_i's were marked. Then $\alpha(p_i) = 1$ for all i. Since the algo said *Sat*, $q \neq \perp$. Then q is also marked (step 2 of algo). Hence, $\alpha(q) = 1$, and we have $\alpha(B) = 1$.

The Horn Algorithm

- If B has the form $\top \to p$, then p is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $p_1 \wedge \dots p_m \to q$, then q is either $\perp$ or an atomic formula.
- If some p_i was not marked, then $\alpha(p_i) = 0$, and hence $\alpha(B) = 1$.
- Assume all the p_i's were marked. Then $\alpha(p_i) = 1$ for all i. Since the algo said *Sat*, $q \neq \perp$. Then q is also marked (step 2 of algo). Hence, $\alpha(q) = 1$, and we have $\alpha(B) = 1$.
- Thus, the markings of the algorithm gives rise to a satisfying assignment α if the algorithm said *Sat*.

Complexity of Horn

- Given a Horn formula ψ with *n* propositions, how many times do you have to read ψ?
- Step 1: Read once
- Step 2: Read atmost *n* times
- Step 3: Read once

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable. Call each conjunct in the CNF a clause.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable. Call each conjunct in the CNF a clause.
- If $\psi = (a \vee b \vee \neg d) \wedge (d \vee \neg a \vee b)$, then $C_1 = \{a, b, \neg d\}$ and $C_2 = \{d, \neg a, b\}$ are the two clauses.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable. Call each conjunct in the CNF a clause.
- If $\psi = (a \vee b \vee \neg d) \wedge (d \vee \neg a \vee b)$, then $C_1 = \{a, b, \neg d\}$ and $C_2 = \{d, \neg a, b\}$ are the two clauses.
- We also write ψ as $\{\{a, b, \neg d\}, \{d, \neg a, b\}\}$.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable. Call each conjunct in the CNF a clause.
- If $\psi = (a \vee b \vee \neg d) \wedge (d \vee \neg a \vee b)$, then $C_1 = \{a, b, \neg d\}$ and $C_2 = \{d, \neg a, b\}$ are the two clauses.
- We also write ψ as $\{\{a, b, \neg d\}, \{d, \neg a, b\}\}$.
- Let C_1, C_2 be two clauses. Assume $a \in C_1$ and $\neg a \in C_2$ for some atomic formula a. Then the clause $R = (C_1 - \{a\}) \cup (C_2 - \{\neg a\})$ is a resolvent of C_1 and C_2.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable. Call each conjunct in the CNF a clause.
- If $\psi = (a \vee b \vee \neg d) \wedge (d \vee \neg a \vee b)$, then $C_1 = \{a, b, \neg d\}$ and $C_2 = \{d, \neg a, b\}$ are the two clauses.
- We also write ψ as $\{\{a, b, \neg d\}, \{d, \neg a, b\}\}$.
- Let C_1, C_2 be two clauses. Assume $a \in C_1$ and $\neg a \in C_2$ for some atomic formula a. Then the clause $R = (C_1 - \{a\}) \cup (C_2 - \{\neg a\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{a_1, \neg a_2, a_3\}$ and $C_2 = \{a_2, \neg a_3, a_4\}$. As $a_3 \in C_1$ and $\neg a_3 \in C_2$, we can find the resolvent. A resolvent is $\{a_1, a_2, \neg a_2, a_4\}$.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable. Call each conjunct in the CNF a clause.
- If $\psi = (a \vee b \vee \neg d) \wedge (d \vee \neg a \vee b)$, then $C_1 = \{a, b, \neg d\}$ and $C_2 = \{d, \neg a, b\}$ are the two clauses.
- We also write ψ as $\{\{a, b, \neg d\}, \{d, \neg a, b\}\}$.
- Let C_1, C_2 be two clauses. Assume $a \in C_1$ and $\neg a \in C_2$ for some atomic formula a. Then the clause $R = (C_1 - \{a\}) \cup (C_2 - \{\neg a\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{a_1, \neg a_2, a_3\}$ and $C_2 = \{a_2, \neg a_3, a_4\}$. As $a_3 \in C_1$ and $\neg a_3 \in C_2$, we can find the resolvent. A resolvent is $\{a_1, a_2, \neg a_2, a_4\}$.
- Resolvent not unique : $\{a_1, a_3, \neg a_3, a_4\}$ is also a resolvent.

Resolution

- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)

Resolution

- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)
- Let F be a formula in CNF. Let R be a resolvent of two clauses of F. Then $F \vdash R$ (Prove!)

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res(F) = Res^0(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^0(F)\}$

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res(F) = Res^0(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^0(F)\}$
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res(F) = Res^0(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^0(F)\}$
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F)$ is finite : there are finitely many clauses that can be derived from F.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res(F) = Res^0(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^0(F)\}$
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F)$ is finite : there are finitely many clauses that can be derived from F.
- $Res^1(F)$ is finite : finitely many resolvents can be obtained from clauses of F

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C$ is a clause in $F\}$.
- $Res(F) = Res^0(F) \cup \{R \mid R$ is a resolvent of two clauses in $Res^0(F)\}$
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R$ is a resolvent of two clauses in $Res^{n-1}(F)\}$
- $Res^0(F)$ is finite : there are finitely many clauses that can be derived from F.
- $Res^1(F)$ is finite : finitely many resolvents can be obtained from clauses of F
- There is some m such that $Res^m(F) = Res^{m+1}(F)$. Denote it by $Res^*(F)$.

Example

Let $F = \{\{a_1, a_2, \neg a_3\}, \{\neg a_2, a_3\}\}$.

- $Res^0(F) = F$

Example

Let $F = \{\{a_1, a_2, \neg a_3\}, \{\neg a_2, a_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{a_1, a_2, \neg a_2\} \cup \{a_1, \neg a_3, a_3\}$.

Example

Let $F = \{\{a_1, a_2, \neg a_3\}, \{\neg a_2, a_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{a_1, a_2, \neg a_2\} \cup \{a_1, \neg a_3, a_3\}$.
- $Res^2(F) = Res^1(F) \cup \{a_1, a_2, \neg a_3\} \cup \{a_1, a_3, \neg a_2\}$