CS 228 : Logic in Computer Science

S. Krishna

Completeness : Recap

- Show that if $\models \psi$ (ψ is valid), then $\vdash \psi$ (we can prove ψ)

Completeness : Recap

- Show that if $\models \psi$ (ψ is valid), then $\vdash \psi$ (we can prove ψ)
- Assume $p_1, \ldots, p_n$ as the variables in ψ, and let $\models \psi$.

Completeness : Recap

- Show that if $\models \psi$ (ψ is valid), then $\vdash \psi$ (we can prove ψ)
- Assume $p_1, \dots, p_n$ as the variables in ψ, and let $\models \psi$.
- Let $\hat{p}_1, \dots, \hat{p}_n$ be the assignment of $p_1, \dots, p_n$ for any line l in the truth table

Completeness : Recap

- Show that if $\models \psi$ (ψ is valid), then $\vdash \psi$ (we can prove ψ)
- Assume $p_1, \ldots, p_n$ as the variables in ψ, and let $\models \psi$.
- Let $\hat{p}_1, \ldots, \hat{p}_n$ be the assignment of $p_1, \ldots, p_n$ for any line l in the truth table
- Show that $\hat{p}_1, \ldots, \hat{p}_n \vdash \psi$

Completeness : Recap

- Show that if $\models \psi$ (ψ is valid), then $\vdash \psi$ (we can prove ψ)
- Assume $p_1, \ldots, p_n$ as the variables in ψ, and let $\models \psi$.
- Let $\hat{p}_1, \ldots, \hat{p}_n$ be the assignment of $p_1, \ldots, p_n$ for any line l in the truth table
- Show that $\hat{p}_1, \ldots, \hat{p}_n \vdash \psi$
- The above step gives 2^n proofs for ψ, starting from 2^n different premises

Completeness : Recap

- Show that if $\models \psi$ (ψ is valid), then $\vdash \psi$ (we can prove ψ)
- Assume $p_1, \dots, p_n$ as the variables in ψ, and let $\models \psi$.
- Let $\hat{p}_1, \dots, \hat{p}_n$ be the assignment of $p_1, \dots, p_n$ for any line l in the truth table
- Show that $\hat{p}_1, \dots, \hat{p}_n \vdash \psi$
- The above step gives 2^n proofs for ψ, starting from 2^n different premises
- Combine all these proofs, and give a proof for ψ starting with no premises

Completeness : Recap

- To combine the proofs, use LEM. That is, use $p_1 \vee \neg p_1$.

Completeness : Recap

- To combine the proofs, use LEM. That is, use $p_1 \vee \neg p_1$.
- You need to prove ψ individually from p_1 and from $\neg p_1$ (why?)

Completeness : Recap

- To combine the proofs, use LEM. That is, use $p_1 \vee \neg p_1$.
- You need to prove ψ individually from p_1 and from $\neg p_1$ (why?)
- Within p_1, use $p_2 \vee \neg p_2$. This opens up two proof obligations, one where you have p_1, p_2, and other where you have $p_1, \neg p_2$.

Completeness : Recap

- To combine the proofs, use LEM. That is, use $p_1 \vee \neg p_1$.
- You need to prove ψ individually from p_1 and from $\neg p_1$ (why?)
- Within p_1, use $p_2 \vee \neg p_2$. This opens up two proof obligations, one where you have p_1, p_2, and other where you have $p_1, \neg p_2$.
- The same can be done within $\neg p_1$, and in fact inside each $p_i, \neg p_i$.

Completeness : Recap

- To combine the proofs, use LEM. That is, use $p_1 \vee \neg p_1$.
- You need to prove ψ individually from p_1 and from $\neg p_1$ (why?)
- Within p_1, use $p_2 \vee \neg p_2$. This opens up two proof obligations, one where you have p_1, p_2, and other where you have $p_1, \neg p_2$.
- The same can be done within $\neg p_1$, and in fact inside each $p_i, \neg p_i$.
- This gives rise to discharging 2^n proofs for ψ, which is what you had.

Completeness : Recap

- To combine the proofs, use LEM. That is, use $p_1 \vee \neg p_1$.
- You need to prove ψ individually from p_1 and from $\neg p_1$ (why?)
- Within p_1, use $p_2 \vee \neg p_2$. This opens up two proof obligations, one where you have p_1, p_2, and other where you have $p_1, \neg p_2$.
- The same can be done within $\neg p_1$, and in fact inside each $p_i, \neg p_i$.
- This gives rise to discharging 2^n proofs for ψ, which is what you had.
- This gives a proof of ψ with no premises.

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 CNF if it is a conjunction of a disjunction of literals.

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

  each C_i is a clause and each $L_{i,j}$ is a literal.
  - $(x_1 \vee \neg x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_2)$

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 CNF if it is a conjunction of a disjunction of literals.

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

  each C_i is a clause and each $L_{i,j}$ is a literal.
    - $(x_1 \vee \neg x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_1 \vee \neg x_2)$
- 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.
    - $(x_1 \wedge \neg x_2) \vee (x_3 \wedge x_4 \wedge x_5) \vee (\neg x_1 \wedge \neg x_2)$

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, $(x \rightarrow [\neg(y \vee z) \wedge \neg(y \rightarrow x)])$

- 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.

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.
- Get rid of all double negations, and replace all subformulae
  - $\neg(G \wedge H)$ 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 replace all subformulae
  - $\neg(G \wedge H)$ 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.

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

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.
- How hard is checking satisfiability, in general?

Satisfiability Checking : Horn Formulae

- 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 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 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 $p \wedge q \wedge \cdots \wedge r \rightarrow s$ 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 $p \wedge q \wedge \cdots \wedge r \rightarrow \perp$.

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 $\top \rightarrow p$ is a subformula.
- If there is a subformula of the form $(p_1 \wedge \cdots \wedge p_m) \rightarrow q$, 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 $\top \rightarrow p$ is a subformula.
- If there is a subformula of the form $(p_1 \wedge \cdots \wedge p_m) \rightarrow q$, 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 $(p_1 \wedge \cdots \wedge p_m) \rightarrow \bot$. 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

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

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

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

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

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

An Example

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

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

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 $\mathcal{S} = \{C_1, \ldots, C_n\}$ 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 $\mathcal{S} = \{C_1, \ldots, C_n\}$ 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 $\mathcal{S} = \{C_1, \ldots, C_n\}$ 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 $\top \to C_i$, then $\alpha(C_i) = 1$. If B has the form $(C_1 \wedge \cdots \wedge C_n) \to D$, where each $\alpha(C_i) = 1$, then $\alpha(D) = 1$. Hence, $\alpha(C_i)$ agrees with the marking of the algo.

The Horn Algorithm

- Assume the algo says H is unsat. Then there is a subformula B of the form $(A_1 \wedge \cdots \wedge A_m) \rightarrow \perp$, where each A_i is marked. Hence, $\alpha(A_i) = 1$ for each A_i. Then $\alpha(B) = 0$, a contradiction to our assumption that $\alpha(B) = 1$ for each B.

The Horn Algorithm

- Assume the algo says H is unsat. Then there is a subformula B of the form $(A_1 \wedge \cdots \wedge A_m) \rightarrow \perp$, where each A_i is marked. Hence, $\alpha(A_i) = 1$ for each A_i. Then $\alpha(B) = 0$, a contradiction to our assumption that $\alpha(B) = 1$ for each B.
- 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

- Assume the algo says H is unsat. Then there is a subformula B of the form $(A_1 \wedge \cdots \wedge A_m) \to \perp$, where each A_i is marked. Hence, $\alpha(A_i) = 1$ for each A_i. Then $\alpha(B) = 0$, a contradiction to our assumption that $\alpha(B) = 1$ for each B.
- 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 $\alpha(C_i) = 1$ iff C_i is marked. We claim that $\alpha \models H$.

The Horn Algorithm

- Assume the algo says H is unsat. Then there is a subformula B of the form $(A_1 \wedge \cdots \wedge A_m) \to \perp$, where each A_i is marked. Hence, $\alpha(A_i) = 1$ for each A_i. Then $\alpha(B) = 0$, a contradiction to our assumption that $\alpha(B) = 1$ for each B.
- 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 $\alpha(C_i) = 1$ iff C_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 $\top \rightarrow A$, then A is marked in step 1 of the algo, and so $\alpha(B) = 1$.

The Horn Algorithm

- If B has the form $\top \rightarrow A$, then A is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $A_1 \wedge \ldots A_m \rightarrow G$, then G is either $\perp$ or an atomic formula.

The Horn Algorithm

- If B has the form $\top \to A$, then A is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $A_1 \wedge \dots A_m \to G$, then G is either $\perp$ or an atomic formula.
- If some A_i was not marked, then $\alpha(A_i) = 0$, and hence $\alpha(B) = 1$.

The Horn Algorithm

- If B has the form $\top \to A$, then A is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $A_1 \wedge \dots A_m \to G$, then G is either $\bot$ or an atomic formula.
- If some A_i was not marked, then $\alpha(A_i) = 0$, and hence $\alpha(B) = 1$.
- Assume all the A_i's were marked. Then $\alpha(A_i) = 1$ for all i. Since the algo said *Sat*, $G \neq \bot$. Then G is also marked (step 2 of algo). Hence, $\alpha(G) = 1$, and we have $\alpha(B) = 1$.

The Horn Algorithm

- If B has the form $\top \to A$, then A is marked in step 1 of the algo, and so $\alpha(B) = 1$.
- If B has the form $A_1 \wedge \dots A_m \to G$, then G is either $\perp$ or an atomic formula.
- If some A_i was not marked, then $\alpha(A_i) = 0$, and hence $\alpha(B) = 1$.
- Assume all the A_i's were marked. Then $\alpha(A_i) = 1$ for all i. Since the algo said *Sat*, $G \neq \perp$. Then G is also marked (step 2 of algo). Hence, $\alpha(G) = 1$, 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