innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Horn Clauses and Horn Formulas:

- **Alternative Form**
- **Limitation**
- **Satisfiability**

Horn Clauses and Horn Formulas

- A propositional logic formula is said to be a ***Horn formula*** if it is a *conjunction of Horn clauses*
  - where ***a Horn clause*** is of the form:
    - $p_1 \wedge p_2 \wedge \ldots p_k \dashrightarrow q$
  - where p_i and q are:
    - either atomic propositions or
    - atomic values (i.e. TRUE or FALSE)

Horn Clauses: Alternative Form

- A ***Horn clause*** is of the form:
  - $\mathbf{p_1 \wedge p_2 \wedge \ldots \wedge p_k \dashrightarrow q}$

  where p_i and q are atomic propositions or atomic values (i.e. TRUE or FALSE)

- Alternative form of a Horn clause:
  - $\mathbf{p_1 \wedge p_2 \wedge \ldots \wedge p_k \dashrightarrow q}$ is equivalent to
  - $\mathbf{\neg(p_1 \wedge p_2 \wedge \ldots \wedge p_k) \vee q}$ which is equivalent to
  - $\mathbf{\neg p_1 \vee \neg p_2 \vee \ldots \vee \neg p_k \vee q}$
- Therefore an alternative description of a Horn clause is this:
  - *A Horn clause is a disjunction of literals in which at most one of them is positive.*

Horn Formulas – Satisfiability

- When is a Horn formula satisfiable?
  - When is a Horn clause satisfiable?
    - When is p-->q satisfiable for any atomic propositions p and q?
  - But a Horn clause may be formed out of atomic values (TRUE and FALSE) as well:
    - Is p-->FALSE satisfiable?
    - Is TRUE-->q satisfiable?
    - Is TRUE-->FALSE satisifiable?

Horn Formulas – Satisfiability

- When is a Horn formula *satisfiable*?
  - When is a <u>conjunction of Horn clauses</u> *satisfiable*?
    - Transitivity of Implication!
  - e.g. TRUE -->q $\wedge$ q -->FALSE:
    - Can you generalize this?

- Argue whether this example formula is *satisfiable* or not:
  - $(p \wedge q \text{ --> } r) \wedge (s \text{ --> } p) \wedge (t \text{ --> } q) \wedge (s \text{ --> } t) \wedge (\neg r) \wedge (s)$

innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Algorithm for Satisfiability of Horn Formulas

Satisfiability of Horn Formulas

- When is a Horn formula satisfiable?
  - When is a Horn clause satisfiable?
    - p --> q is satisfiable for any atomic propositions p and q
  - But a Horn clause may be formed out of atomic values (TRUE and FALSE) as well:
    - TRUE --> FALSE is not satisfiable.
  - When is a conjunction of Horn clauses not satisfiable?
    - Consider C_1 of the form p_1-->q_1 and C_2 of the form p_2-->q_2:
      - What if q_1 and p_2 are the same but p_1 is TRUE and q_2 is FALSE.
    - Can you generalize this?

Satisfiability of Horn Formulas

- A Horn formula is a conjunction of Horn clauses:
  - Consider these formulas – for each, identify whether it is satisfiable and if so, when is it satisfied?
    - (p-->q) $\wedge$ (q-->FALSE)
    - (p-->q)$\wedge$(s-->t)$\wedge$(q-->s)$\wedge$(q-->r)$\wedge$(TRUE-->p)$\wedge$(t-->u)$\wedge$(u-->FALSE)

Satisfiability of Horn Formulas

- HORN_SAT(ϕ)
  - pre-condition: ϕ is a Horn formula
  - returns : *yes* if ϕ is satisfiable, *no* otherwise
- Steps:

1. **mark** all occurrences of TRUE in ϕ
2. while

   (there is a clause $p_1 \wedge p_2 \wedge \ldots \wedge p_k$ --> q of ϕ such that all p_j are ***marked*** *but q is* ***not marked****)*

   do { mark q }
3. if FALSE is **marked** then return "no" else return "yes"

Satisfiability of Horn Formulas

- HORN_SAT(ϕ)
  - pre-condition: ϕ is a Horn formula
  - returns : *yes* if ϕ is satisfiable, *no* otherwise
- Steps:

1. **mark** all occurrences of TRUE in ϕ
2. while

   *(there is a clause $p_1 \wedge p_2 \wedge \ldots \wedge p_k$ --> q of ϕ such that all p_j are **marked** but q is **not marked**)*

   do { mark q }
3. if FALSE is **marked** then return "no" else return "yes"

Claim: This algorithm terminates for all correct inputs

Satisfiability of Horn Formulas

- HORN_SAT(ϕ)
  - pre-condition: ϕ is a Horn formula
  - returns : *yes* if ϕ is satisfiable, *no* otherwise
- Steps:

1. **mark** all occurrences of TRUE in ϕ
2. while

   *(there is a clause $p_1 \wedge p_2 \wedge \ldots \wedge p_k$ --> q of ϕ such that all p_j are **marked** but q is **not marked**)*

   do { mark q }
3. if FALSE is **marked** then return "no" else return "yes"

Claim: This algorithm terminates in at most n+1 iterations where n is the number of atomic propositions in ϕ

innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Horn Clauses vs. Propositional Formulas

Horn Formulas - Limitation

- Can any propositional formula be written as a Horn formula?
  - No, by Horn's Theorem.
- Examples?

Horn's Theorem

- A propositional formula φ over atoms $x_1,\ldots,x_n$ is expressible as a conjunction of Horn clauses <u>if and only if</u>
  - whenever φ evaluates to 1 for the assignments $\mathbf{z_1,\ldots, z_n}$ and $\mathbf{y_1,\ldots,y_n}$
  - it also evaluates to 1 for the assignment $\mathbf{z_1 \wedge y_1, \ldots, z_n \wedge y_n}$

  for all assignments $\vec{z}$ and $\vec{y}$

- Exercise:
  - *Find formulas that cannot be written in Horn form.*
    - Hint: <u>*Use a truth table.*</u> End of Hint.

innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Horn Clauses and Prolog Programming

Logic Programming - Prolog

- HORN clauses form the basis of the programming language Prolog
  - Of course Prolog uses predicates rather than propositions.
- A program in Prolog is a collection of rules
  - where each rule is a Horn clause.
    - e.g. grandparent(X,Y) :- parent(X,Z), parent(Z,Y).
  - is a Horn clause:
    - parent(X,Z) $\wedge$ parent(Z,Y) --> grandparent(X,Y).
- A query is answered (i.e. resolved) in Prolog by finding a proof for the query using the given rules.

Prolog Programming and Complexity

- The execution overhead in a Prolog program is ***polynomial-time.***
  - *i.e*. execution of a Prolog program proceeds by finding satisfiable assignments for Horn Clauses
    - which can be done in polynomial-time.