innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: PROPOSITIONAL LOGIC

Validity and Conjunctive Normal Form

Conjunctive Normal Form (CNF)

RECALL

- A propositional logic formula φ is said to be in **CNF** if the formula is *a conjunction of sub-formulas* (or ***clauses***):

  i.e. it is of the form $\mathbf{C_1 \wedge C_2 \wedge \ldots \wedge C_n}$

  where each clause $\mathbf{C_i}$ is a *disjunction of literals*:

  i.e. it is of the form $\mathbf{L_{i1} \vee L_{i2} \vee \ldots \vee L_{im}}$
- where each literal $\mathbf{L_{ij}}$ is
  - either *an atomic proposition* (**p**) or the negation of *an atomic proposition* (**¬p**) .

- In Boolean logic, the CNF is referred to as the ***Product-of-Sums*** (***POS***) form.

Validity and CNF

- Consider a formula ϕ in CNF:
  - Let ϕ be $C_1 \wedge C_2 \wedge \ldots \wedge C_n$
    - Then ϕ is valid *if and only if* C_i is valid for all i
  - Let a given clause C_i be $L_{i1} \vee L_{i2} \vee \ldots \vee L_{im}$
    - Then, under what conditions will C_i be valid?

Validity and CNF

- Consider a formula ϕ in CNF:
  - Let $\boldsymbol{\phi}$ be $\mathbf{C_1 \wedge C_2 \wedge \ldots \wedge C_n}$
    - Then $\boldsymbol{\phi}$ is valid *if and only if* $\mathbf{C_i}$ is valid for all **i**
- Let a given clause $\mathbf{C_i}$ be $\mathbf{L_{i1} \vee L_{i2} \vee \ldots \vee L_{im}}$
- Question:
  - *Under what conditions will C_i be valid*?
  - Answer:
    - *C_i will be valid only if it includes a proposition **p** and its negation* i.e.:
      - there exist **k** and **l** such that $\mathbf{L_{ik}}$ is **p** and $\mathbf{L_{il}}$ is $\neg$**p** for some propositional atom **p**

CNF – Algorithm for checking Validity

- **isValidCNF(ϕ)**
  1. Let **ϕ** be $\mathbf{C_1 \wedge C_2 \wedge \ldots \wedge C_n}$
  2. *for each* i:
     1. if not(isValidDisClause($\mathbf{C_i}$)) then return FALSE;
  3. return TRUE;
- **isValidDisClause(C)**
  1. Let **C** be $\mathbf{L_1 \vee L_2 \vee \ldots \vee L_m}$
  2. *for each* **i**:
     1. if $\mathbf{L_i}$ is **p** { *for each* **j>i**: if $\mathbf{L_j}$ is $\neg$**p** then return TRUE }
        else { *for each* **j>i**: if $\mathbf{L_j}$ is **p** then return TRUE }
  3. return FALSE;

CNF – Algorithm for checking Validity

isValidCNF(ϕ)

Let **ϕ** be $C_1 \wedge C_2 \wedge \ldots \wedge C_n$

for each i:

if not(isValidDisClause(C_i))
then return FALSE;

return TRUE;

isValidDisClause(C)

Let **C** be $L_1 \vee L_2 \vee \ldots \vee L_m$

for each **i**:

if L_i is **p** { *for each* **j>i**: if L_j is $\neg$**p** then return TRUE }

else { *for each* **j>i**: if L_j is **p** then return TRUE }

return FALSE;

CNF – Cost of checking Validity

Time taken by **isValidCNF**:

- **ϕ** has **n** clauses:
  - **n** * (1 + (*time taken by* ***isValidDisClause***))
    - in the worst case

Time taken by **isValidDisClause**:

- The input clause has **m** literals:
  - **1 + m** * (1 + (**m** * 1))
    - in the worst case

Cost of checking Validity

- Given a formula in CNF with **n** clauses, and each clause with at most m literals:
  - the cost of checking whether the formula is valid:
    - **c*(n + *m*n + m*m*n)** in the worst case for some constant **c**
- Thus the cost of checking whether a formula in CNF is valid is at most:
  - **$c * m^2 * n$** for a suitable constant c

Cost of checking Validity

- Compare the cost of checking validity of a formula in CNF with
  - the cost of checking validity of a formula *using the truth table method* (or equivalently, *by testing a circuit*):
    - (# rows in truth table) * (cost of evaluating the formula)
    - **2^k * (n*m)**
      - where **k** different propositional atoms are used in the formula.
- Is there a better way for checking validity of a propositional logic formula, if it is not known to be in CNF?