0 Announcements

- Assignment 1 is due on Friday midnight (29th Jan, 11:59). Dropbox link will be shared soon. Project topics are posted and new related topics can be proposed that utilize tool(s) like Isabelle, Alloy.

1 Review of Previous Lecture

- Definitions of Conjunctive Normal Form (CNF)
  A) BNF
  \[ \varphi ::= p \mid \neg \varphi \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \]

  B) NNF: In the parse tree, all “\(\neg\)” immediately precede propositional atom \(p\)
  \[ \varphi ::= p \mid \neg p \mid \varphi_1 \land \varphi_2 \mid \varphi_1 \lor \varphi_2 \]

- SAT solvers
  - are used for automating WFF satisfiability.
  - assume WFF to be in CNF

2 Efficient Transformation Into CNF

2.1 Proving efficiency:

Lemma: Every propositional WFF \(\varphi\) in the syntax of BNF definition (A) can be translated in linear time into an equivalent propositional WFF \(\psi\) in the syntax of BNF definition (B) such that,
\[ |\psi| = \left(\frac{3}{2}\right) \cdot |\varphi|. \]

Proof: By Structural Induction,

- Base case (trivial case):
  \(\varphi\) has 0 logical connectives. Which means \(\varphi\) is \(p\) (an atom). In this case, \(\psi\) is indeed \(\varphi\), and obviously the inequality in lemma holds.
  We can also state the base case using 1 logical connective. Here, \(\varphi\) can be \(\neg p\) or \(p \lor q\) or \(p \land q\). All the three WFFs are already defined in (B).

- Induction Hypothesis (IH):
  State lemma is true for WFF \(\varphi\) with total 0,1,\ldots,n logical connectives.

- Induction step: (To prove the lemma, we must prove validity for \(\varphi\) with total \((n+1)\) logical connectives)
  Assume a WFF \(\varphi\) having \((n+1)\) logical connectives.
  Here, \(\varphi\) can be \(\neg \varphi_0\) or \(\varphi_1 \land \varphi_2\) or \(\varphi_1 \lor \varphi_2\),
  such that \(\varphi_0, \varphi_1\) and \(\varphi_2\) can be translated into NNF \(\psi_0, \psi_1\) and \(\psi_2\) in linear time where,
  \[ |\psi_i| < \left(\frac{3}{2}\right) \cdot |\varphi_i|. \]
  To prove the induction step, consider all the three compositions of \(\varphi\):
i. $\varphi = \varphi_1 \land \varphi_2$
ii. $\varphi = \varphi_1 \lor \varphi_2$
iii. $\varphi = \neg \varphi_0$

(i) and (ii) are trivial from the definition of NNF. To prove (iii), we need to consider another lemma which can be stated as follows,

**Lemma A:** If there is a set of WFFs $\psi_0$ already in NNF, then $\neg \psi_0$ can be linearly transformed into NNF $\psi'$ such that, $|\psi'| < \left(\frac{3}{2}\right) \cdot |\neg \psi_0|$.

**Proof:** Lemma A can be proved using *structural induction* by applying Morgan’s laws where we can replace occurrences of $\land$ with $\lor$ and $p$ with $\neg p$ and vice-versa.

Continuing with the original proof for (iii), we can now convert $\neg \varphi_0$ to $\neg \psi_0$ in two steps,

- Convert $\varphi_0$ to $\psi_0$ in linear time. —– from IH
- Convert $\neg \psi_0$ to $\psi'$ by pushing down the ‘$\neg$’ in linear time. —– from Lemma A

Using (i), (ii) and (iii), we can prove the induction steps which proves the original lemma.

### 3 SAT solvers

SAT solvers work on WFFs in CNF and need some preprocessing to ensure that. Before looking at preprocessing steps, let us define the problem for SAT solvers.

#### 3.1 Defining Problems:

**SAT problem (Theoretical view)**

Input - A set of clauses $C$ built from a propositional language with $n$ variables.
Output - Is there an assignment of the $n$ variables that satisfies all those clauses?
For example,

$C_1 = \{\neg a \lor b, \neg b \lor c\} = (\neg a \lor b) \land (\neg b \lor c)$ —– *satisfiable*

$C_2 = C_1 \cup \{a, \neg c\} = C_1 \land a \land \neg c$ —– *not satisfiable*

**SAT solvers (Practical view)**

Input - A set of clauses $C$ built from a propositional language with $n$ variables.
Output - If there is a assignment of the $n$ variables that satisfies all those clauses, provide such assignment, else provide a subset of C which cannot be satisfied.

**k-SAT problem:** For $k=2$, satisfiable assignment can be found in deterministic polynomial time. But for $k>3$, no algorithm is known to find an assignment in polynomial time. In other words, $k$-SAT is NP-complete for $k>3$.

**Core Engines:**

- #SAT: If the set of clauses is satisfiable, then find out how many different configurations.
- MAXSAT: If there is no satisfying configuration, then find out the maximum subset of literals that satisfies the set of clauses.

#### 3.2 Preprocessing Rules:

Before passing the WFFs to SAT solvers, first we have to convert them into CNF. Following recursive rules can be applied for this step,

- $\text{CNF}(p, \triangle) := (p, \triangle)$
  Here := implies the function return tuple. In this case, for an atom $p$ there is no work done.
• CNF(¬ϕ, Δ) := (¬l, Δ′) where, CNF(ϕ, Δ) = ⟨l, Δ⟩.
  Note: The second CNF() call is a recursive call to calculate l and Δ'. More rules are listed here.

3.3 Equisatisfiability

Theorem: Let ϕ be an arbitrary propositional WFF and let CNF(ϕ, φ) = ⟨l, Δ⟩. Then ϕ is satisfiable iff
l ∪ Δ is satisfiable.

3.4 Approaches to SAT solvers

1) Stochastic search:
   • First, guess full assignment for all atoms.
   • If the assignment doesn’t satisfy F then the solver starts to flip the values of atoms based on certain
     heuristics.

2) Exhaustive search: (more common)
   • The solver traverses a binary tree in which internal nodes are partial valuations and leaves are full
     valuations.
   • Then it repeatedly backtracks to find a satisfying full valuation.

3.5 Universe of Propositional WFFs

• Valid WFFs/tautologies
  Warning: Negation of satisfiable WFFs is NOT an unsatisfiable WFF.

• Unsatisfiable WFFs/contradictions
  Warning: Negation of a falsifiable WFF is NOT a tautology.

• Contingent:
  WFFs that are neither tautologies nor contradictions are sometimes called contingent. This division
  can be seen in [HD07 page 18].

In the next class we will discuss SAT solvers and define resolution.