Lecture 5 - Resolution

Vaishnavi Sundararajan

COL703 - Logic for Computer Science

CNF: Deleting "unnecessary" clauses

- We would like to show that $\{\varphi_0, \dots, \varphi_n\} \vDash \psi$
- Needs us to show that $(\bigwedge_{0 \leqslant i \leqslant n} \varphi_i) \wedge \neg\psi$ is unsatisfiable
- Convert $(\bigwedge_{0 \leqslant i \leqslant n} \varphi_i) \wedge \neg\psi$ into CNF
- This yields a set of clauses; each clause a set of literals
- Systematically delete "unnecessary" clauses from this set of clauses
- If we are left with $\{\emptyset\}$ at the end, the expression is unsatisfiable; therefore ψ is a logical consequence of $\{\varphi_0, \dots, \varphi_n\}$
- **Note**: $\{\emptyset\}$ is not satisfiable, but $\{\}$ is vacuously satisfiable! Pay attention to what set you get.

How to delete “unnecessary” clauses

- Consider a CNF expression φ which is a set of clauses $\delta_1, \ldots, \delta_n$
- If $\delta_i \subseteq \delta_j$ for some $1 \leqslant i, j \leqslant n$, delete δ_j
- If $\{p, \neg p\} \subseteq \delta_i$ for $p \in AP$ and some $1 \leqslant i \leqslant n$, delete δ_i
- Call a CNF expression “clean” if no further deletion can be performed
- **Theorem**: Any CNF expression φ is logically equivalent to its clean version φ^*
- **Proof idea**: A clause containing $\{p, \neg p\}$ as a subset evaluates to T under any valuation. By Absorption, a clause in a CNF expression is logically equivalent to its subset.

Propositional resolution

- For a clean set φ^* and $p \in AP$, define

$$\Delta_p = \{\delta \in \varphi^* \mid p \in \delta\} \text{ and } \overline{\Delta}_p = \{\delta' \in \varphi^* \mid \neg p \in \delta'\}$$

- Since φ^* is clean, $\Delta_p \cap \overline{\Delta}_p = \emptyset$ for any $p \in AP$
- We **resolve** a clean set φ^* of clauses by
  - Removing both Δ_p and $\overline{\Delta}_p$ from φ^*,
  - removing p and $\neg p$ from each pair δ, δ' such that $\delta \in \Delta_p$ and $\delta' \in \overline{\Delta}_p$, and
  - adding the resultant clause back to φ^*

$$\begin{aligned} \text{resolve}(\varphi^*, p) \triangleq {} & (\varphi^* \setminus (\Delta_p \cup \overline{\Delta}_p)) \\ & \cup \left\{ (\delta \cup \delta') \setminus \{p, \neg p\} \;\middle|\; \delta \in \Delta_p,\ \delta' \in \overline{\Delta}_p \right\} \end{aligned}$$

About resolve

Theorem: Suppose δ_1 and δ_2 are clauses such that $p \in (\delta_1 \setminus \delta_2)$ and $\neg p \in (\delta_2 \setminus \delta_1)$ for some $p \in AP$. If a valuation satisfies δ_1 and δ_2, then it satisfies $\delta = (\delta_1 \cup \delta_2) \setminus \{p, \neg p\}$.

Proof: Let $\delta_1 = \ell_{11} \vee \ell_{12} \vee \dots \ell_{1r} \vee p$ and $\delta_2 = \ell_{21} \vee \ell_{22} \vee \dots \ell_{2s} \vee \neg p$
Suppose there is a valuation τ that satisfies δ_1 and δ_2.
Any valuation will make exactly one of p or $\neg p$ true, not both.
So at least one of r or s is greater than 0, and at least one of $\{\ell_{1i}, \ell_{2j}\}$ for some i and j is made true by τ. This literal is retained in δ, so τ satisfies δ also.

About resolve

Theorem: Suppose δ_1 and δ_2 are such that $p \in (\delta_1 \setminus \delta_2)$ and $\neg p \in (\delta_2 \setminus \delta_1)$ for some $p \in AP$. If $\delta = (\delta_1 \cup \delta_2) \setminus \{p, \neg p\}$ is satisfiable, then so are δ_1 and δ_2.

Proof: Let $\delta_1 = \ell_{11} \vee \ell_{12} \vee \ldots \ell_{1r} \vee p$ and $\delta_2 = \ell_{21} \vee \ell_{22} \vee \ldots \ell_{2s} \vee \neg p$. Suppose $\delta = \ell_1 \vee \ell_2 \vee \ldots \ell_n$ is satisfied by τ. Thus, δ is not empty (**why?**), and $\ell_i \notin \{p, \neg p\}$ for every i. So τ does not enforce any valuation on p. The following possibilities arise.

- δ contains an ℓ_{1i} and an ℓ_{2j} for some i, j. In that case, τ satisfies δ_1 and δ_2
- δ contains no ℓ_{1i} but contains an ℓ_{2j}. Set $\tau'(p)$ to T, preserve the behaviour of τ on others. δ_1 is satisfied by τ' (since it makes p true) and δ_2 is satisfied (since it makes ℓ_{2j} true).
- δ contains an ℓ_{1i} but contains no ℓ_{2j}. Similar to above, set $\tau'(p)$ to F, preserve the behaviour of τ on others.

Resolution: Algorithm

- Input: A clean set Δ of clauses
- Loop while $\emptyset \notin \Delta$ and there is at least one pair of clauses δ_1 and δ_2 which contain p and $\neg p$. If not, return Δ.
- In the loop body,
  - Compute $\Delta' = \text{resolve}(\Delta, p)$
  - Clean Δ' by deleting unnecessary clauses
  - Set Δ to be the clean version of Δ'
- The input expression is unsat iff $\emptyset$ is in the set of clauses under examination

Resolution: Algorithm analysis

- **Theorem (Termination)**: Given a clean set Δ as input, the algorithm terminates and returns a clean set.
  - If Δ already contains $\emptyset$, the algorithm terminates immediately
  - Otherwise, it might be the case that each atom and its negated form present in some pair of clauses
  - Each step eliminates one such pair.
  - Each clause only contains one occurrence (positive or negative) of each propositional atom
  - Terminates in at most $|atoms(\Delta)|$ steps.

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \lor q, p \supset r, q \supset r\}$?
- First, we convert each expression in Γ into CNF

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?
- First, we convert each expression in Γ into CNF
- $\Gamma = \{p \vee q, \neg p \vee r, \neg q \vee r\}$

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?
- First, we convert each expression in Γ into CNF
- $\Gamma = \{p \vee q, \neg p \vee r, \neg q \vee r\}$
- CNF set of clauses is $\Delta = \{\{p, q\}, \{\neg p, r\}, \{\neg q, r\}, \{\neg r\}\}$

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?
- First, we convert each expression in Γ into CNF
- $\Gamma = \{p \vee q, \neg p \vee r, \neg q \vee r\}$
- CNF set of clauses is $\Delta = \{\{p, q\}, \{\neg p, r\}, \{\neg q, r\}, \{\neg r\}\}$
- This set is clean, so we can feed it to the algorithm as input

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?
- First, we convert each expression in Γ into CNF
- $\Gamma = \{p \vee q, \neg p \vee r, \neg q \vee r\}$
- CNF set of clauses is $\Delta = \{\{p, q\}, \{\neg p, r\}, \{\neg q, r\}, \{\neg r\}\}$
- This set is clean, so we can feed it to the algorithm as input
- δ_1 and δ_2 contain $p \in AP$ and $\neg p$ respectively

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?
- First, we convert each expression in Γ into CNF
- $\Gamma = \{p \vee q, \neg p \vee r, \neg q \vee r\}$
- CNF set of clauses is $\Delta = \{\{p, q\}, \{\neg p, r\}, \{\neg q, r\}, \{\neg r\}\}$
- This set is clean, so we can feed it to the algorithm as input
- δ_1 and δ_2 contain $p \in AP$ and $\neg p$ respectively
- $\Delta' = \{\{q, r\}, \{\neg q, r\}, \{\neg r\}\}$ is clean

Resolution: Example

- Is r a logical consequence of $\Gamma = \{p \vee q, p \supset r, q \supset r\}$?
- CNF set of clauses is $\Delta = \{\{p, q\}, \{\neg p, r\}, \{\neg q, r\}, \{\neg r\}\}$
- After eliminating p and $\neg p$, we get $\Delta' = \{\{q, r\}, \{\neg q, r\}, \{\neg r\}\}$
- δ'_1 and δ'_2 contain $q \in AP$ and $\neg q$ respectively
- $\Delta'' = \{\{r\}, \{\neg r\}\}$
- Similarly eliminate r and $\neg r$ to get $\Delta''' = \{\emptyset\}$
- Algorithm terminates here, since $\emptyset \in \Delta'''$
- Δ is unsat, so $\Gamma \vDash r$

Resolution: Example

Is r a logical consequence of $\Gamma = \{(p \vee q) \vee \neg r, p \supset r, q \supset r\}$?

Resolution: Example

Is $(p \wedge q) \vee s$ a logical consequence of

$\Gamma = \{(p \vee q) \vee \neg r, p \supset q, p \supset \neg r, q \supset s, s \supset p, s \supset r\}$?

Resolution: Algorithm analysis

- **Theorem (Soundness)**: If the algorithm returns $\{\emptyset\}$, Δ is unsat
  - Often easier to prove this as *each step* of the algorithm being sound
  - If Δ' is the clean set obtained after one iteration of the algorithm on Δ, if Δ' is unsat, then Δ is unsat.
- **Proof sketch**: Suppose towards a contradiction there is some valuation τ that makes Δ true.
  Use the first theorem about resolve, which ensures that satisfiability is preserved. Any CNF expression is logically equivalent to its clean version, so Δ' is also satisfied by τ.

Resolution: Algorithm analysis

- **Theorem (Completeness)**: If Δ is unsat, the algorithm returns $\{\emptyset\}$
  - Needs termination; already shown
  - Once again, can prove for each step
  - If Δ' is the clean set obtained after one iteration of the algorithm on Δ, if Δ is unsat, then Δ' is also unsat.
- **Proof idea**: Prove by contradiction. Use the second theorem about resolve, and that a set is logically equivalent to its clean version.

Unit resolution

- Can we pick the eliminated proposition “more intelligently”?
- **Unit resolution**: Prefer a clause that only contains one literal
- Suppose I pick two clauses $\delta_1 = \{\ell\}$ and $\delta_2 = \{\ell_1, \neg\ell, \ell_2, \ell_3\}$
- The resolve step creates a clause of the form $\{\ell_1, \ell_2, \ell_3\}$
- Throw away δ_1 entirely, and reduces the size of δ_2 by one
- If multiple clauses contain $\neg\ell$, all their sizes reduce by one
- **Exercise**: How many steps does the algorithm take to terminate if the unit resolution strategy suffices to yield a result?

Falsum

- Is $(\neg q \vee r) \wedge \neg r$ a logical consequence of $\Gamma = \{p, \neg p\}$?
- Needs no relationship between p and the consequent formula!
- Anything is a logical consequence of a contradiction!
- Introduce a new wff $\bot$; always evaluates to F
- Grammar for propositional logic (PL) is now as follows

$$\varphi, \psi := p \mid \bot \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \vee \psi \qquad \text{where } p \in AP$$

- For any PL wff φ, $\varphi \wedge \neg\varphi \Leftrightarrow \bot$
- **Exercise**: Show that $\neg\varphi \Leftrightarrow \varphi \supset \bot$

Resolution: Bigger picture

- "If I see p and its negation, I throw both away"
- **Does not matter what truth value is assigned to p**
- All manipulation happens at the level of **syntax**
- Even though we are checking for logical consequence/validity
- Can write a **proof rule** to capture resolve

Proof rules

$$\frac{\ell_{11} \vee \ell_{12} \vee \ldots \vee \ell_{1m} \vee p \qquad\qquad \ell_{21} \vee \ell_{22} \vee \ldots \vee \ell_{2n} \vee \neg p}{\ell_{11} \vee \ell_{12} \vee \ldots \vee \ell_{1m} \vee \ell_{21} \vee \ell_{22} \vee \ldots \vee \ell_{2n}} \text{ res}$$

- The horizontal line indicates inference
- The name of the inference rule is given next to the line
- Every expression above the line is called a **premise**
- The expression below the line is called the **conclusion**
- "If all the premises hold, then the conclusion holds"
- Each ℓ_{ij} and p a variable; can substitute any literal and any atom
- Cannot change the "shape" of expressions though