CS 228 : Logic in Computer Science

S. Krishna

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable.
- Let C_1, C_2 be two clauses. We use set notation for the literals in a clause. If $C_1 = A \vee \neg B$, then $C_1 = \{A, \neg B\}$.
- Assume $A \in C_1$ and $\neg A \in C_2$ for some atomic formula A. Then the clause $R = (C_1 - \{A\}) \cup (C_2 - \{\neg A\})$ is a resolvent of C_1 and C_2.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable.
- Let C_1, C_2 be two clauses. We use set notation for the literals in a clause. If $C_1 = A \vee \neg B$, then $C_1 = \{A, \neg B\}$.
- Assume $A \in C_1$ and $\neg A \in C_2$ for some atomic formula A. Then the clause $R = (C_1 - \{A\}) \cup (C_2 - \{\neg A\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{A_1, \neg A_2, A_3\}$ and $C_2 = \{A_2, \neg A_3, A_4\}$. As $A_3 \in C_1$ and $\neg A_3 \in C_2$, we can find the resolvent. The resolvent is $\{A_1, A_2, \neg A_2, A_4\}$.

Resolution

- Resolution is a technique used to check if a formula in CNF is satisfiable.
- Let C_1, C_2 be two clauses. We use set notation for the literals in a clause. If $C_1 = A \vee \neg B$, then $C_1 = \{A, \neg B\}$.
- Assume $A \in C_1$ and $\neg A \in C_2$ for some atomic formula A. Then the clause $R = (C_1 - \{A\}) \cup (C_2 - \{\neg A\})$ is a resolvent of C_1 and C_2.
- Let $C_1 = \{A_1, \neg A_2, A_3\}$ and $C_2 = \{A_2, \neg A_3, A_4\}$. As $A_3 \in C_1$ and $\neg A_3 \in C_2$, we can find the resolvent. The resolvent is $\{A_1, A_2, \neg A_2, A_4\}$.
- Resolvent not unique : $\{A_1, A_3, \neg A_3, A_4\}$ is also a resolvent.

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)
- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)

3 rules in Resolution

- Let G be any formula. Let F be the CNF formula resulting from the CNF algorithm applied to G. Then $G \vdash F$ (Prove!)
- Let F be a formula in CNF, and let C be a clause in F. Then $F \vdash C$ (Prove!)
- Let F be a formula in CNF. Let R be a resolvent of two clauses of F. Then $F \vdash R$ (Prove!)
  - As a simplest case of this, prove that $A \vee B, A \vee \neg B \vdash A$ (Hint : use LEM)

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F) =$clauses in F, and there are finitely many clauses that can be derived from F.

Completeness of Resolution

Show that resolution can be used to determine whether any given formula is satisfiable.

- Given F in CNF, let $Res^0(F) = \{C \mid C \text{ is a clause in } F\}$.
- $Res^n(F) = Res^{n-1}(F) \cup \{R \mid R \text{ is a resolvent of two clauses in } Res^{n-1}(F)\}$
- $Res^0(F) =$clauses in F, and there are finitely many clauses that can be derived from F.
- Hence, there is some m such that $Res^m(F) = Res^{m+1}(F)$. Denote it by $Res^*(F)$.

Example

Let $F = \{\{A_1, A_2, \neg A_3\}, \{\neg A_2, A_3\}\}$.

- $Res^0(F) = F$

Example

Let $F = \{\{A_1, A_2, \neg A_3\}, \{\neg A_2, A_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{A_1, A_2, \neg A_2\} \cup \{A_1, \neg A_3, A_3\}$.

Example

Let $F = \{\{A_1, A_2, \neg A_3\}, \{\neg A_2, A_3\}\}$.

- $Res^0(F) = F$
- $Res^1(F) = F \cup \{A_1, A_2, \neg A_2\} \cup \{A_1, \neg A_3, A_3\}$.
- $Res^2(F) = Res^1(F) \cup \{A_1, A_2, \neg A_3\} \cup \{A_1, A_3, \neg A_2\}$

Resolution

Let F be a formula in CNF. If $\emptyset \in Res^*(F)$, then F is unsatisfiable.

- If $\emptyset \in Res^*(F)$. Then $\emptyset \in Res^n(F)$ for some n.

Resolution

Let F be a formula in CNF. If $\emptyset \in Res^*(F)$, then F is unsatisfiable.

- If $\emptyset \in Res^*(F)$. Then $\emptyset \in Res^n(F)$ for some n.
- Since $\emptyset \notin Res^0(F)$ ($\emptyset$ is not a clause), there is an $m > 0$ such that $\emptyset \notin Res^m(F)$ and $\emptyset \in Res^{m+1}(F)$.

Resolution

Let F be a formula in CNF. If $\emptyset \in Res^*(F)$, then F is unsatisfiable.

- If $\emptyset \in Res^*(F)$. Then $\emptyset \in Res^n(F)$ for some n.
- Since $\emptyset \notin Res^0(F)$ ($\emptyset$ is not a clause), there is an $m > 0$ such that $\emptyset \notin Res^m(F)$ and $\emptyset \in Res^{m+1}(F)$.
- Then $\{A\}, \{\neg A\} \in Res^m(F)$. By the rules of resolution, we have $F \vdash A, \neg A$, and thus $F \vdash \bot$. Hence, F is unsatisfiable.

Resolution

Prove the converse: If F is unsatisfiable, then $\emptyset \in Res^*(F)$.

Resolution : Converse

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.

Resolution : Converse

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$
  ($C_i = p \vee \neg p \vee \cdots \equiv$ *true*)

Resolution : Converse

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$ ($C_i = p \vee \neg p \vee \cdots \equiv$ *true*)
- Induct on the number n of propositional variables that occur in F.

Resolution : Converse

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$ ($C_i = p \vee \neg p \vee \cdots \equiv \; true$)
- Induct on the number n of propositional variables that occur in F.
- If $n = 1$, then the possible clauses are p, $\neg p$ ($p \vee \neg p$ is ruled out, by assumption).

Resolution : Converse

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \ldots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$ ($C_i = p \vee \neg p \vee \cdots \equiv \ true$)
- Induct on the number n of propositional variables that occur in F.
- If $n = 1$, then the possible clauses are p, $\neg p$ ($p \vee \neg p$ is ruled out, by assumption).
- If $F = \{\{p\}\}$ or $F = \{\{\neg p\}\}$, F is satisfiable.

Resolution : Converse

If F in CNF is unsatisfiable, then $\emptyset \in Res^*(F)$.

- Let F have k clauses $C_1, \dots, C_k$.
- wlg, assume that no C_i has both p and $\neg p$ ($C_i = p \vee \neg p \vee \dots \equiv \textit{true}$)
- Induct on the number n of propositional variables that occur in F.
- If $n = 1$, then the possible clauses are p, $\neg p$ ($p \vee \neg p$ is ruled out, by assumption).
- If $F = \{\{p\}\}$ or $F = \{\{\neg p\}\}$, F is satisfiable.
- Hence, $F = \{\{p\}, \{\neg p\}\}$. Clearly, $\emptyset \in Res^1(F)$.

Resolution : Converse

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.

Resolution : Converse

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

Resolution : Converse

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \dots, p_{n+1}$.

  - Let G_0 be the conjunction of all C_i in F such that $\neg p_{n+1} \notin C_i$.
  - Let G_1 be the conjunction of all C_i in F such that $p_{n+1} \notin C_i$.

Resolution : Converse

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

  - Let G_0 be the conjunction of all C_i in F such that $\neg p_{n+1} \notin C_i$.
  - Let G_1 be the conjunction of all C_i in F such that $p_{n+1} \notin C_i$.

- Clauses in F= Clauses in G_0 $\cup$ Clauses in G_1

Resolution : Converse

- Inductive hypothesis : If F has $\leqslant n$ variables and is unsat, then $\emptyset \in Res^*(F)$.
- Let F have $n+1$ variables $p_1, \ldots, p_{n+1}$.

  - Let G_0 be the conjunction of all C_i in F such that $\neg p_{n+1} \notin C_i$.
  - Let G_1 be the conjunction of all C_i in F such that $p_{n+1} \notin C_i$.

- Clauses in F= Clauses in $G_0 \cup$ Clauses in G_1

- Let $F_0 = \{C_i - \{p_{n+1}\} \mid C_i \in G_0\}$
- Let $F_1 = \{C_i - \{\neg p_{n+1}\} \mid C_i \in G_1\}$

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$

- If p_{n+1} is assigned *false* in F, then F is equivalent to F_0

$$F = (p_1 \vee \mathit{false}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \mathit{false}) \wedge (\neg p_2 \vee \neg \mathit{false})$$

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$

- If p_{n+1} is assigned *false* in F, then F is equivalent to F_0

$$F = (p_1 \vee \ \mathit{false}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \ \mathit{false}) \wedge (\neg p_2 \vee \neg \mathit{false})$$

- If p_{n+1} is assigned *true* in F, then F is equivalent to F_1

$$F = (p_1 \vee \ \mathit{true}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \ \mathit{true}) \wedge (\neg p_2 \vee \ \neg \mathit{true})$$

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$
- If p_{n+1} is assigned *false* in F, then F is equivalent to F_0

$$F = (p_1 \vee \mathit{false}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \mathit{false}) \wedge (\neg p_2 \vee \neg \mathit{false})$$

- If p_{n+1} is assigned *true* in F, then F is equivalent to F_1

$$F = (p_1 \vee \mathit{true}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \mathit{true}) \wedge (\neg p_2 \vee \neg \mathit{true})$$

- Hence $F \equiv F_0 \vee F_1$.

Resolution

Let $F = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}, \{\neg p_2, \neg p_3\}\}$ and $n = 2$.

- $G_0 = \{\{p_1, p_3\}, \{p_2\}, \{\neg p_1, \neg p_2, p_3\}\}$, $G_1 = \{\{p_2\}, \{\neg p_2, \neg p_3\}\}$.
- $F_0 = \{\{p_1\}, \{p_2\}, \{\neg p_1, \neg p_2\}\}$ and $F_1 = \{\{p_2\}, \{\neg p_2\}\}$
- If p_{n+1} is assigned *false* in F, then F is equivalent to F_0

$$F = (p_1 \vee \ \mathit{false}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \ \mathit{false}) \wedge (\neg p_2 \vee \neg \mathit{false})$$

- If p_{n+1} is assigned *true* in F, then F is equivalent to F_1

$$F = (p_1 \vee \ \mathit{true}) \wedge p_2 \wedge (\neg p_1 \vee \neg p_2 \vee \ \mathit{true}) \wedge (\neg p_2 \vee \ \neg \mathit{true})$$

- Hence $F \equiv F_0 \vee F_1$.
- As F is unsatisfiable, F_0 and F_1 are both unsatisfiable.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.
- Hence, $\emptyset \in Res^*(G_0)$ or $\{p_{n+1}\} \in Res^*(G_0)$, and $\emptyset \in Res^*(G_1)$ or $\{\neg p_{n+1}\} \in Res^*(G_1)$.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.
- Hence, $\emptyset \in Res^*(G_0)$ or $\{p_{n+1}\} \in Res^*(G_0)$, and $\emptyset \in Res^*(G_1)$ or $\{\neg p_{n+1}\} \in Res^*(G_1)$.
- If $\emptyset \in Res^*(G_0)$ or $\emptyset \in Res^*(G_1)$, then $\emptyset \in Res^*(F)$.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.
- Hence, $\emptyset \in Res^*(G_0)$ or $\{p_{n+1}\} \in Res^*(G_0)$, and $\emptyset \in Res^*(G_1)$ or $\{\neg p_{n+1}\} \in Res^*(G_1)$.
- If $\emptyset \in Res^*(G_0)$ or $\emptyset \in Res^*(G_1)$, then $\emptyset \in Res^*(F)$.
- Else, $\{p_{n+1}\} \in Res^*(G_0)$ and $\{\neg p_{n+1}\} \in Res^*(G_1)$.

Resolution

- By induction hypothesis, $\emptyset \in Res^*(F_0)$ and $\emptyset \in Res^*(F_1)$.
- Hence, $\emptyset \in Res^*(G_0)$ or $\{p_{n+1}\} \in Res^*(G_0)$, and $\emptyset \in Res^*(G_1)$ or $\{\neg p_{n+1}\} \in Res^*(G_1)$.
- If $\emptyset \in Res^*(G_0)$ or $\emptyset \in Res^*(G_1)$, then $\emptyset \in Res^*(F)$.
- Else, $\{p_{n+1}\} \in Res^*(G_0)$ and $\{\neg p_{n+1}\} \in Res^*(G_1)$.
- Hence $\emptyset \in Res^*(F)$.

Resolution Summary

Given a formula ψ, convert it into CNF, say ζ. ψ is satisfiable iff $\emptyset \notin \mathit{Res}^*(\zeta)$.

- If ψ is unsat, we might get $\emptyset$ before reaching $\mathit{Res}^*(\zeta)$.
- If ψ is sat, then truth tables might be faster : stop when some row evaluates to 1.

Propositional Logic : Summary

- Syntax, Semantics
- Encoding problems into logic
- Sound and Complete Proof Engine
- Semantic/Provable equivalence of formulae
- Normal forms, satisfiability, hardness
- Resolution for SAT checking

Moving On

Propositional Logic

SAT solvers, heuristics, competitions for SAT solvers and so on. In many cases, parts of a complex problem reduced to SAT solving.

What we propose to do now

Move on to other logics, and their applications in CS.