Resolution Theorem Proving

Algorithm:

1. Convert KB and $\neg (\beta)$ to CNF, conjoin them to get KB’
2. Apply resolution to KB’ until you get an empty conjunction
   (or run out of things to resolve without getting an empty conjunction)
Conversion to CNF

Summary

1. Remove implications:
   Replace $\alpha \Rightarrow \beta$ with $\neg \alpha \lor \beta$ everywhere  
   [no more “$\Rightarrow$”]

2. Push in $\neg$
   Replace $\neg (\alpha \lor \beta)$ with $\neg \alpha \land \neg \beta$  
   [“$\neg$” only before propositional symbols]
   Replace $\neg (\alpha \land \beta)$ with $\neg \alpha \lor \neg \beta$

3. Remove double negations:
   Replace $\neg \neg \alpha$ with $\alpha$  
   [“$\neg$” only singletons]

4. Distribute $\lor$ over $\land$:
   Rewrite $\alpha \lor (\beta \land \gamma)$ as $(\alpha \lor \beta) \land (\alpha \lor \gamma)$
Resolution Inference Rule

\[
\begin{align*}
(p \lor \alpha_1 \lor ... \lor \alpha_a) \\
(\neg p \lor \beta_1 \lor ... \lor \beta_b) \quad \text{where each } \alpha_i \text{ and } \beta_j \text{ are literals (symbols or their negations) and } p \text{ is a propositional symbol}
\end{align*}
\]
Resolution Inference Rule

\[(p \lor \alpha_1 \lor ... \lor \alpha_a) \quad \text{where each } \alpha_i \text{ and } \beta_j \text{ are literals (symbols or their negations) and}
\]
\[\neg p \lor \beta_1 \lor ... \lor \beta_b\]
\[\alpha_1 \lor ... \lor \alpha_a \lor \beta_1 \lor ... \lor \beta_b\]

\[(P \lor Q \lor \neg R)\]
\[\neg S \lor T \lor \neg Q \lor V\]
\[(P \lor \neg R \lor \neg S \lor T \lor V)\]
Resolution as Search

States: The collection (conjunction) of what you know (clauses)
Operators: Take two clauses and resolve them
Initial State: $\text{CNF}(\varphi) \land \text{CNF}(\neg \psi)$
Goal State: State includes ()
Resolution Proof Example

(¬ my ∨ i) (my ∨ mo) (my ∨ ml) (¬ i ∨ h) (¬ ml ∨ h) (¬ h ∨ ma) (¬ ma)
Resolution Proof Example

\[ (¬ \text{my} ∨ \text{i}) \quad (\text{my} ∨ \text{mo}) \quad (\text{my} ∨ \text{ml}) \quad (¬ \text{i} ∨ \text{h}) \quad (¬ \text{ml} ∨ \text{h}) \quad (¬ \text{h} ∨ \text{ma}) \quad (¬ \text{ma}) \]
Resolution Proof Example

\((-my \lor i)\) \hspace{1cm} (my \lor mo) \hspace{1cm} (my \lor ml) \hspace{1cm} (-i \lor h) \hspace{1cm} (-ml \lor h) \hspace{1cm} (-h \lor ma) \hspace{1cm} (-ma)
Resolution Proof Example

$$\neg \text{my} \lor \text{i} \quad \boxed{\text{my} \lor \text{mo}} \quad \text{(my} \lor \text{ml}) \quad \neg \text{i} \lor \text{h} \quad \neg \text{ml} \lor \text{h} \quad \neg \text{h} \lor \text{ma} \quad \neg \text{ma}$$
Resolution Proof Example

(¬ my ∨ i) (my ∨ mo) (my ∨ ml) (¬i ∨ h) (¬ml ∨ h) (¬ h ∨ ma) (¬ ma)
Resolution Proof Example

$\neg my \lor i \quad my \lor mo \quad (my \lor ml) \quad \neg i \lor h \quad (\neg ml \lor h) \quad \neg h \lor ma \quad \neg ma$
Resolution Proof Example

\[\neg my \lor i \quad \neg mo \lor ml \quad \neg i \lor h \quad \neg ml \lor h \quad \neg h \lor ma \quad \neg ma\]
Resolution Proof Example

\((\neg \text{my} \lor i) \ (\text{my} \lor \text{mo}) \ (\text{my} \lor \text{ml}) \ (\neg i \lor h) \ (\neg \text{ml} \lor h) \ (\neg h \lor \text{ma}) \ (\neg \text{ma})\)
Resolution Proof Example

$$\neg (\neg \text{my} \lor \text{i}) \quad (\text{my} \lor \text{mo}) \quad (\text{my} \lor \text{ml}) \quad (\neg \text{i} \lor \text{h}) \quad (\neg \text{ml} \lor \text{h}) \quad (\neg \text{h} \lor \text{ma}) \quad (\neg \text{ma})$$
Resolution Proof Example

\((\neg \text{my } \lor \text{i}) \ (\text{my } \lor \text{mo}) \ (\text{my } \lor \text{ml}) \ (\neg \text{i } \lor \text{h}) \ (\neg \text{ml } \lor \text{h}) \ (\neg \text{h } \lor \text{ma}) \ (\neg \text{ma})\)

7 successor states
Each is a copy of the state above with a new clause added to it
Tractability of Resolution

1. Convert KB and $\neg \beta$ to CNF, conjoin them to get KB’
2. Apply resolution to KB’ until you get an empty clause

Worst case for both steps is exponential
Tractability of Resolution

What to do?
Tractability of Resolution

What to do?

Hope for the best
Tractability of Resolution

What to do?

Hope for the best
Consider restricted languages
Horn Clause Logic

• There are restricted forms of propositional logic for which $|\text{CNF}(\varphi)|$ is polynomial in $|\varphi|$

• Propositional Horn Clause Logic:

\[ (p_{1,1} \land \ldots \land p_{1,k_1}) \Rightarrow q_1 \]
\[ \vdots \]
\[ (p_{n,1} \land \ldots \land p_{n,k_n}) \Rightarrow q_n \]
\[ r_1, r_2, \ldots \]

• $\text{CNF}(|\varphi|)$ is linear in $|\varphi|$ for Horn Clause Logic

• There are specialized inference methods for Horn Clause Logic that run in time linear in KB and $\beta$
Tractability of Resolution

What to do?

Hope for the best
Consider restricted languages
Tractability of Resolution

What to do?

Hope for the best
Consider restricted languages
Consider other inference operations
Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence $\varphi$ in CNF is unsatisfiable
Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence $\varphi$ in CNF is unsatisfiable

Let’s try to show that $\varphi$ in CNF is *satisfiable*
Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence $\phi$ in CNF is unsatisfiable

Let’s try to show that $\phi$ in CNF is *satisfiable*

$\phi$ is satisfiable if there is an assignment of truth values to each of the propositional symbols in $\phi$ that makes $\phi$ true
Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence $\phi$ in CNF is unsatisfiable.

Let’s try to show that $\phi$ in CNF is *satisfiable*.

$\phi$ is satisfiable if there is an assignment of truth values to each of the propositional symbols in $\phi$ that makes $\phi$ true.

Let’s search for a satisfying assignment directly.
Other Inference Approaches for Propositional Logic

Let’s search for a satisfying assignment directly

Why?
Many problems can be reduced to a satisfiability question
The satisfying assignment can be converted back into a solution to the original problem
Example: N-Queens

• Encode in propositional logic ("Queens(N)")
  • Qij: There is a queen in position (i,j)  
  • One queen in each row and column:  
  • (Qi1 ∨ Qi2 ∨ ... ∨ QiN) for i ∈ {1,...,N}  
  • (Q1j ∨ Q2j ∨ ... ∨ QNj) for j ∈ {1,...,N}  
  • No two attacking queens:
    • ¬ (Qab ∧ Qac) for all b ≠ c  
    • ¬ (Qab ∧ Qdb) for all a ≠ d  
    • ¬ (Qij ∧ Qkl) for all k=i+m, l=j+m, m ≠ 0  
    • ¬ (Qij ∧ Qkl) for all i+j = k+l, i ≠ k  
  • Conjoin everything together to get a sentence φ
Example: N-Queens

• Encode in propositional logic (“Queens(N)”)  
  • Qij: There is a queen in position (i,j)  
  • One queen in each row and column:  
    • (Qi1 ∨ Qi2 ∨ ... ∨ QiN) for i ∈ {1,...,N}  
    • (Q1j ∨ Q2j ∨ ... ∨ QNj) for j ∈ {1,...,N}  
  • No two attacking queens:  
    • ¬(Qab ∧ Qac) for all b ≠ c  
    • ¬(Qab ∧ Qdb) for all a ≠ d  
    • ¬(Qij ∧ Qkl) for all k=i+m, l=j+m, m ≠ 0  
    • ¬(Qij ∧ Qkl) for all i+j = k+l, i ≠ k  
  • Conjoin everything together to get a sentence φ  

• If φ is unsatisfiable, there is no solution  
• If φ is satisfiable, the truth assignment gives a solution
Example: N-Queens

- If $N = 2$ or $3$
Example: N-Queens

• If N = 2 or 3 there is no legal placement of questions
Example: N-Queens

• If $N = 2$ or $3$ there is no legal placement of questions
  • Queens(2) and Queens(3) are unsatisfiable
Example: N-Queens

• If N = 2 or 3 there is no legal placement of questions
  • Queens(2) and Queens(3) are unsatisfiable
  • (Could use resolution to prove it)
Example: N-Queens

• If N = 2 or 3 there is no legal placement of questions
  • Queens(2) and Queens(3) are unsatisfiable
  • (Could use resolution to prove it)

• For N > 3
Example: N-Queens

- If $N = 2$ or $3$ there is no legal placement of questions
  - Queens(2) and Queens(3) are unsatisfiable
  - (Could use resolution to prove it)

- For $N > 3$ there is always a legal placement of queens
Example: N-Queens

- If N = 2 or 3 there is no legal placement of questions
  - Queens(2) and Queens(3) are unsatisfiable
  - (Could use resolution to prove it)

- For N > 3 there is always a legal placement of queens
  - Queens(N) is satisfiable for N > 3
Example: N-Queens

• If N = 2 or 3 there is no legal placement of questions
  • Queens(2) and Queens(3) are unsatisfiable
  • (Could use resolution to prove it)

• For N > 3 there is always a legal placement of queens
  • Queens(N) is satisfiable for N>3
  • Find a satisfying truth assignment for Queens(N), then read off the solution (the values of the Qij)
Searching for Truth Assignments

• Stupid brute-force search (“backtracking”):
  • Assign $P_1 = P_2 = \ldots = P_n = \text{True}$
  • Is sentence satisfied?
  • If yes, done
  • If not, assign $P_n = \text{False}$
  • Is sentence satisfied?
  • If not, assign $P_{n-1} = \text{False}$, $P_n = \text{True}$

  • Etc.

• Steps through all $2^n$ assignments
Searching for Truth Assignments

Can do better
Searching for Truth Assignments

Can do better

(Let’s assume $\varphi$ has been converted to CNF before we start)
Searching for Truth Assignments

• Stupid brute-force search (“backtracking”):
  • Assign $P_1 = P_2 = \ldots = P_n = \text{True}$
  • Is sentence satisfied?
  • If yes, done
  • If not, assign $P_n = \text{False}$
  • Is sentence satisfied?
  • If not, assign $P_{n-1} = \text{False}$, $P_n = \text{True}$

• Etc.

• Steps through all $2^n$ assignments

• Problem what if sentence is $\neg P_1 \wedge (\ldots)$?
Searching for Truth Assignments

• Brute-force search ("backtracking") for CNF:
  • Assign a variable True or False
  • Is any clause violated?
    • If not, go on to another variable
    • If yes, backtrack to most recent variable assignment and flip it
    • If both options have been tried and failed, backtrack to the next most recent assignment
Searching for Truth Assignments

• Brute-force search ("backtracking") for CNF:
  • Assign a variable True or False
  • Is any clause violated?
    • If not, go on to another variable
    • If yes, backtrack to most recent variable assignment and flip it
      • If both options have been tried and failed, backtrack to the next most recent assignment
  • Complete
Searching for Truth Assignments

• Brute-force search (“backtracking”) for CNF:
  • Assign a variable True or False
  • Is any clause violated?
    • If not, go on to another variable
    • If yes, backtrack to most recent variable assignment and flip it
    • If both options have been tried and failed, backtrack to the next most recent assignment
  • Complete

• Made more efficient with lots of clever observations gathered over the years
  • \( \neg P_1 \land \ldots \) means set \( P_1 = \text{False} \)
• Guaranteed to find a solution
  • Worst-case exponential
  • Surprisingly effective!
Searching for Truth Assignments

• Hill climbing approaches:
  • Pick random assignment
  • If it doesn’t satisfy the sentence, flip one of the variable’s values
    • GSAT:
      • Flip whichever variable most increases the number of satisfied terms, or
      • With some probability flip a random variable
    • WalkSAT
      • Pick an unsatisfied clause at random
      • Flip which variable violates the fewest previously satisfied clauses, or
      • With some probability flip a random variable in the clause

• Not complete
• Surprisingly effective