CS 4700: Foundations of Artificial Intelligence

Spring 2020

Prof. Haym Hirsh

Lecture 13

February 21, 2020

Resolution Theorem Proving

Algorithm:

1. Convert KB and $\neg$ (β) 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 $\neg\alpha \vee \beta$ everywhere [no more "$\Rightarrow$"]
2. Push in $\neg$
   Replace $\neg(\alpha \vee \beta)$ with $\neg\alpha \wedge \neg\beta$
   Replace $\neg(\alpha \wedge \beta)$ with $\neg\alpha \vee \neg\beta$
   ["$\neg$" only before propositional symbols]
3. Remove double negations:
   Replace $\neg\neg\alpha$ with α ["$\neg$" only singletons]
4. Distribute $\vee$ over $\wedge$:
   Rewrite $\alpha \vee (\beta \wedge \gamma)$ as $(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Resolution Inference Rule

$$\frac{(\boxed{\boldsymbol{p}} \vee \alpha_1 \vee \ldots \vee \alpha_a) \qquad (\neg \boldsymbol{p} \vee \beta_1 \vee \ldots \vee \beta_b)}{(\alpha_1 \vee \ldots \vee \alpha_a \vee \beta_1 \vee \ldots \vee \beta_b)}$$

where each α_i and β_j are literals (symbols or their negations) and $\boldsymbol{p}$ is a propositional symbol

Resolution Inference Rule

$$\frac{(\boxed{\boldsymbol{p}} \vee \alpha_1 \vee \ldots \vee \alpha_a) \qquad (\neg \boldsymbol{p} \vee \beta_1 \vee \ldots \vee \beta_b)}{(\alpha_1 \vee \ldots \vee \alpha_a \vee \beta_1 \vee \ldots \vee \beta_b)}$$

where each α_i and β_j are literals (symbols or their negations) and $\boldsymbol{p}$ is a propositional symbol

$$\frac{(P \vee Q \vee \neg R) \qquad (\neg S \vee T \vee \neg Q \vee V)}{(P \vee \neg R \vee \neg S \vee T \vee V)}$$

Resolution as Search

States: The collection (conjunction) of what you know (clauses)

Operators: Take two clauses and resolve them

Initial State: CNF(φ) $\wedge$ CNF($\neg\psi$)

Goal State: State includes ()

Resolution Proof Example

$\boxed{(\neg\, my \vee i)}$ $\boxed{(my \vee mo)}$ $(my \vee ml)$ $(\neg i \vee h)$ $(\neg ml \vee h)$ $(\neg\, h \vee ma)$ $(\neg\, ma)$

Resolution Proof Example

$\boxed{(\neg my \vee i)}$ $(my \vee mo)$ $\boxed{(my \vee ml)}$ $(\neg i \vee h)$ $(\neg ml \vee h)$ $(\neg h \vee ma)$ $(\neg ma)$

Resolution Proof Example

$\boxed{(\neg\, my \vee i)}$ $(my \vee mo)$ $(my \vee ml)$ $\boxed{(\neg i \vee h)}$ $(\neg ml \vee h)$ $(\neg\, h \vee ma)$ $(\neg\, ma)$

Resolution Proof Example

$(\neg my \vee i)$ $\boxed{(my \vee mo)}$ $(my \vee ml)$ $(\neg i \vee h)$ $(\neg ml \vee h)$ $(\neg h \vee ma)$ $(\neg ma)$

Resolution Proof Example

$(\neg my \vee i)$ $\boxed{(my \vee mo)}$ $(my \vee ml)$ $(\neg i \vee h)$ $(\neg ml \vee h)$ $(\neg h \vee ma)$ $(\neg ma)$

Resolution Proof Example

$(\neg my \vee i)$ $(my \vee mo)$ $\boxed{(my \vee ml)}$ $(\neg i \vee h)$ $\boxed{(\neg ml \vee h)}$ $(\neg h \vee ma)$ $(\neg ma)$

Resolution Proof Example

$(\neg my \vee i)$ $(my \vee mo)$ $(my \vee ml)$ $\boxed{(\neg i \vee h)}$ $(\neg ml \vee h)$ $\boxed{(\neg h \vee ma)}$ $(\neg ma)$

Resolution Proof Example

$(\neg my \vee i)$ $(my \vee mo)$ $(my \vee ml)$ $(\neg i \vee h)$ $\boxed{(\neg ml \vee h)}$ $\boxed{(\neg h \vee ma)}$ $(\neg ma)$

Resolution Proof Example

$(\neg my \vee i)$ $(my \vee mo)$ $(my \vee ml)$ $(\neg i \vee h)$ $(\neg ml \vee h)$ $\boxed{(\neg h \vee ma)}$ $\boxed{(\neg ma)}$

Resolution Proof Example

$(\neg my \vee i)$ $(my \vee mo)$ $(my \vee ml)$ $(\neg i \vee h)$ $(\neg ml \vee h)$ $(\neg h \vee ma)$ $(\neg ma)$

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 $\neg$ β 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 |CNF(φ)| is polynomial in |φ|
- Propositional Horn Clause Logic:

$$(\boldsymbol{p}_{1,1} \wedge \ldots \wedge \boldsymbol{p}_{1,k1}) \Rightarrow \boldsymbol{q}_1$$

$$\ldots$$

$$(\boldsymbol{p}_{n,1} \wedge \ldots \wedge \boldsymbol{p}_{n,kn}) \Rightarrow \boldsymbol{q}_n$$

$$\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots.$$

  - CNF(|φ|) is linear in |φ| for Horn Clause Logic
  - There are specialized inference methods for Horn Clause Logic that run in time linear in KB and β

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 φ in CNF is unsatisfiable

Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence φ in CNF is unsatisfiable

Let's try to show that φ in CNF is *satisfiable*

Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence φ in CNF is unsatisfiable

Let's try to show that φ in CNF is *satisfiable*

φ is satisfiable if there is an assignment of truth values to each of the propositional symbols in φ that makes φ true

Other Inference Approaches for Propositional Logic

Resolution tries to show that a sentence φ in CNF is unsatisfiable

Let's try to show that φ in CNF is *satisfiable*

φ is satisfiable if there is an assignment of truth values to each of the propositional symbols in φ that makes φ 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) [Queens(8): generates 64 symbols]
  - One queen in each row and column: [Queens(8): generates 2 disjunctions]
    - $(Qi1 \vee Qi2 \vee \ldots \vee QiN)$ for $i \in \{1,\ldots,N\}$
    - $(Q1j \vee Q2j \vee \ldots \vee QNj)$ for $j \in \{1,\ldots,N\}$
  - No two attacking queens:
    - $\neg (Qab \wedge Qac)$ for all $b \neq c$ [Queens(8): generates 224 conjunctions]
    - $\neg (Qab \wedge Qdb)$ for all $a \neq d$ [Queens(8): generates 224 conjunctions]
    - $\neg (Qij \wedge Qkl)$ for all $k=i+m$, $l=j+m$, $m \neq 0$ [Queens(8): generates 224 conjunctions]
    - $\neg (Qij \wedge Qkl)$ for all $i+j = k+l$, $i \neq k$ [Queens(8): generates 224 conjunctions]
  - 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) [Queens(8): generates 64 symbols]
  - One queen in each row and column: [Queens(8): generates 2 disjunctions]
    - $(Qi1 \vee Qi2 \vee \ldots \vee QiN)$ for $i \in \{1,\ldots,N\}$
    - $(Q1j \vee Q2j \vee \ldots \vee QNj)$ for $j \in \{1,\ldots,N\}$
  - No two attacking queens:
    - $\neg (Qab \wedge Qac)$ for all $b \neq c$ [Queens(8): generates 224 conjunctions]
    - $\neg (Qab \wedge Qdb)$ for all $a \neq d$ [Queens(8): generates 224 conjunctions]
    - $\neg (Qij \wedge Qkl)$ for all $k=i+m$, $l=j+m$, $m \neq 0$ [Queens(8): generates 224 conjunctions]
    - $\neg (Qij \wedge Qkl)$ for all $i+j = k+l$, $i \neq k$ [Queens(8): generates 224 conjunctions]
  - 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 $P_1 = P_2 = \ldots = P_n$ = True
  - Is sentence satisfied?
  - If yes, done
  - If not, assign P_n = False
  - Is sentence satisfied?
  - If not, assign P_{n-1} = False, P_n = 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 φ has been converted to CNF before we start)

Searching for Truth Assignments

- Stupid brute-force search ("backtracking") :
  - Assign $P_1 = P_2 = \ldots = P_n$ = True
  - Is sentence satisfied?
  - If yes, done
  - If not, assign P_n = False
  - Is sentence satisfied?
  - If not, assign P_{n-1} = False, P_n = 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
    - $\neg P_1 \wedge \ldots$ means set P_1 = 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