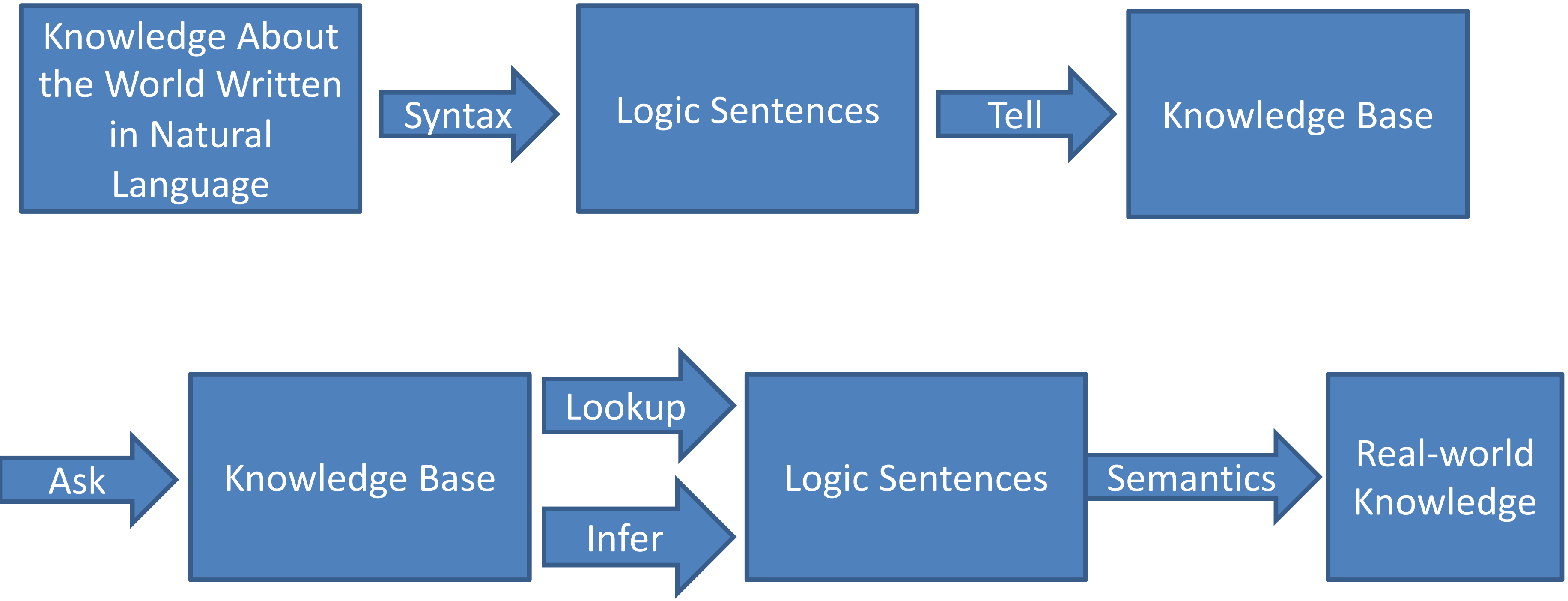
COSC76/276 Artificial Intelligence
Fall 2022
Logical Inference

Soroush Vosoughi
Computer Science
Dartmouth College
Soroush@Dartmouth.edu

Reminders

- PA4 due tonight!
- X-hours today!

Logic Flow



Knowledge About the World Written in Natural Language
Syntax
Logic Sentences
Tell
Knowledge Base
Ask
Knowledge Base
Lookup
Infer
Logic Sentences
Semantics
Real-world Knowledge

Logical inference

- $KB \mid\!-_i \alpha$ means *KB* derives a sentence α using inference procedure *i*

- *Sound* (or *truth preserving*):

  The algorithm **only** derives entailed sentences.

  i is sound iff whenever $KB \mid\!-_i \alpha$ *it is also true that* $KB \models \alpha$

- *Complete*:

  The algorithm can derive **every** entailed sentence.

  i is complete iff whenever $KB \models \alpha$ *it is also true that* $KB \mid\!-_i \alpha$

Validity and satisfiability

A sentence is valid if it is true in all models,

e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is false in all models

e.g., $A \wedge \neg A$

Example: College life

- $\neg$ Dinner $\Leftrightarrow$ FridgeEmpty $\vee$ (AssignmentDue $\wedge$ Procrastinated)
- Winter $\wedge$ NiceWeatherSunday $\Rightarrow$ Procrastinated
- FridgeEmpty $\Rightarrow$ HousemateMad

- ask: if it is not winter, and I did not eat dinner, does that imply that my housemate is mad?
- KnowledgeBase $\wedge \neg W \wedge \neg D \vDash H$?

Naïve Logical Inference

Model checking

Method #1 for inference

- Seven symbols: D, F, P, A, W, H, N.
- Each symbol can take the value true or false.
- Consider all assignments of true/false values.

- If H is true for all models in which all sentences in (KB and ¬W¬D) are true, then H is entailed.

Sophisticated Logical Inference

First convert logic sentences into conjunctive normal form (CNF)

Logical equivalence

- To manipulate logical sentences we need some rewrite rules.
- Two sentences are logically equivalent iff they are true in same models: α ≡ ß iff α ╞ β and β ╞ α

These are important to know

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

CNF rules

1. Eliminate $\Leftrightarrow$, replace with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
2. Eliminate $\Rightarrow$, replace with $\neg\alpha \vee \beta$
3. Move $\neg$ inwards:
   - $\neg(\neg\alpha)$:α
   - $\neg(\alpha \wedge \beta)$:$\neg\alpha \vee \neg\beta$ (DeMorgan)
   - $\neg(\alpha \vee \beta)$:$\neg\alpha \wedge \neg\beta$ (DeMorgan)
4. Distribute $\vee$ over $\wedge$:
   - $\alpha \vee (\beta \wedge \gamma) \rightarrow (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

Now we have a conjunction of disjunctions of literals:

- $(A \vee \neg B \vee C) \wedge$
- $(C \vee \neg D) \wedge \ldots$

The deduction theorem

For any sentences α,β,

$$\alpha \models \beta \text{ iff } (\alpha \Rightarrow \beta) \text{ is valid.}$$

- Entailment is a concept, and involves two sentences.
- $\alpha \Rightarrow \beta$ is a single sentence in PL.

How do we prove that $\alpha \Rightarrow \beta$ is valid?

Model checking? What would happen if we added hundreds of symbols, none of which appear in the KB or in α or β? Model checking would look silly

Proof by contradiction

Goal: prove that $\alpha \Rightarrow \beta$ is valid.

Maybe easy to prove a sentence unsatisfiable:

$A \land \neg A$ is unsatisfiable, even if we add new sentences.

So prove $\neg(\alpha \Rightarrow \beta)$ is unsatisfiable. In CNF:

- $\neg(\neg\alpha \lor \beta)$ unsatisfiable
- $\alpha \land \neg\beta$ unsatisfiable

So, add $\neg\beta$ to KB and show there are no models for which this is true. (Proof by contradiction.)

Inference rules: AND elimination

- If in the KB there are $\alpha \wedge \beta$, then any of the conjuncts can be inferred

$$\frac{\alpha \wedge \beta}{\alpha}$$

Inference rules

- All logical equivalences can be applied as inference rules
  - Example:

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)} \quad \text{and} \quad \frac{(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)}{\alpha \Leftrightarrow \beta}$$

Resolution rule

- One specific resolution rule will be the base of our procedure for inference

$$\frac{l_1 \vee l_2 \ldots \vee l_k, \quad m_1 \vee \ldots m_n}{l_1 \ldots l_{i-1} \vee l_{i+1} \vee \ldots l_k \vee m_1 \ldots m_{j-1} \vee m_{j+1} \vee \ldots m_n},$$

where l_i and m_j are complementary literals

Resolution Examples

$(A \vee B \vee C)$

$(\neg A)$ “If A or B or C is true, but not A, then B or C must be true.”

_ _ _ _ _ _ _ _ _ _ _ _ _ _

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

“If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true.”

_ _ _ _ _ _ _ _ _ _ _ _ _

“If A or B is true, and not A or B is true, then B must be true.”

$(A \vee B)$

$(\neg A \vee B)$

_ _ _ _ _ _ _ _ _

$\longleftarrow$ Factoring

Resolution Examples

$(A \vee B \vee C)$

$(\neg A)$

"If A or B or C is true, but not A, then B or C must be true."

$-----------$

$\therefore (B \vee C)$

$(A \vee B \vee C)$

$(\neg A \vee D \vee E)$

"If A is false then B or C must be true, or if A is true then D or E must be true, hence since A is either true or false, B or C or D or E must be true."

$-----------$

$\therefore (B \vee C \vee D \vee E)$

$(A \vee B)$

$(\neg A \vee B)$

"If A or B is true, and not A or B is true, then B must be true."

$--------$

$\therefore (B \vee B) \equiv B \longleftarrow$ Factoring

Resolution rule typical error

The resolution rule resolves only ONE Literal Pair.
If more than one pair, result always = TRUE.

(OR A B C D)
(OR ¬A ¬B F G)

(OR C D F G)

This is wrong

(OR A B C D)
(OR ¬A ¬B ¬C)

(OR D)

This is wrong

Resolution Algorithm

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- Generate all new sentences from KB and the (negated) query.
- One of two things can happen:

1. We find $P \wedge \neg P$ which is unsatisfiable. i.e.* we **can** entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \wedge \neg\alpha$ and hence we **cannot** entail the query.

Inference by Resolution

- KB is represented in CNF
  - KB = AND of all the sentences in KB
  - KB sentence = clause = OR of literals
  - Literal = propositional symbol or its negation

- Find two clauses in KB, one of which contains a literal and the other its negation
  - Cancel the literal and its negation
  - Bundle everything else into a new clause
  - Add the new clause to KB
  - Repeat

Pseudo-code for resolution

function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

```
clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
new ← { }
loop do
    for each pair of clauses Ci, Cj in clauses do
        resolvents ← PL-RESOLVE(Ci, Cj)
        if resolvents contains the empty clause then return true
        new ← new ∪ resolvents
    if new ⊆ clauses then return false
    clauses ← clauses ∪ new
```

Proof by contradiction, i.e., show KB ^ ~α: unsatisfiable

Resolution example

Stated in Propositional Logic

- “Laws of Physics” in the Wumpus World:
  - “A breeze in B11 is equivalent to a pit in P12 or a pit in P21.”

    $(B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1}))$
- Particular facts about a specific instance:
  - “There is no breeze in B11.”

    $(\neg B_{1,1})$
- Goal or query sentence:
  - “Is it true that P12 does not have a pit?”

    $(\neg P_{1,2})$

Example: Conversion to CNF

Example: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replace with $(\alpha\Rightarrow\beta)\wedge(\beta\Rightarrow\alpha)$.
2. Eliminate $\Rightarrow$, replace with $\neg\alpha\vee\beta$
3. Move $\neg$ inwards:
   - $\neg(\neg\alpha)$:α
   - $\neg(\alpha\wedge\beta)$:$\neg\alpha\vee\neg\beta$ (DeMorgan)
   - $\neg(\alpha\vee\beta)$:$\neg\alpha\wedge\neg\beta$ (DeMorgan)
4. Distribute $\vee$ over $\wedge$:
   - $\alpha\vee(\beta\wedge\gamma)\rightarrow(\alpha\vee\beta)\wedge(\alpha\vee\gamma)$

Write together

Example: Conversion to CNF

Example: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

1. Eliminate $\Leftrightarrow$ by replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)$.
   $= (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$ by replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \lor \beta$ and simplify.
   $= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg(P_{1,2} \lor P_{2,1}) \lor B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and simplify.
   $\neg(\alpha \lor \beta) \equiv (\neg\alpha \land \neg\beta)$, $\neg(\alpha \land \beta) \equiv (\neg\alpha \lor \neg\beta)$
   $= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})$

4. Apply distributive law ($\land$ over $\lor$) and simplify.
   $= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

Example: Conversion to CNF

Example: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$

From the previous slide we had:

$= (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})$

5. KB is the conjunction of all of its sentences (all are true), so write each clause (disjunct) as a sentence in KB:

KB =

…

$(\neg B_{1,1} \lor P_{1,2} \lor P_{2,1})$ → $(\neg B_{1,1} \quad P_{1,2} \quad P_{2,1})$

$(\neg P_{1,2} \lor B_{1,1})$ → $(\neg P_{1,2} \quad B_{1,1})$

$(\neg P_{2,1} \lor B_{1,1})$ → $(\neg P_{2,1} \quad B_{1,1})$

(same)

…

Often, Won't Write "$\lor$" or "$\land$"

(we know they are there)

Resolution example

Resulting Knowledge Base stated in CNF

- "Laws of Physics" in the Wumpus World:

  $(\neg B_{1,1} \quad P_{1,2} \quad P_{2,1})$
  $(\neg P_{1,2} \quad B_{1,1})$
  $(\neg P_{2,1} \quad B_{1,1})$

- Particular facts about a specific instance:

  $(\neg B_{1,1})$

- Negated goal or query sentence:

  $(P_{1,2})$

Resolution example

A Resolution proof ending in ( )

- Knowledge Base at start of proof:

$(\neg B_{1,1} \quad P_{1,2} \quad P_{2,1})$

$(\neg P_{1,2} \quad B_{1,1})$

$(\neg P_{2,1} \quad B_{1,1})$

$(\neg\ B_{1,1})$

$(P_{1,2})$

A resolution proof ending in ( ):

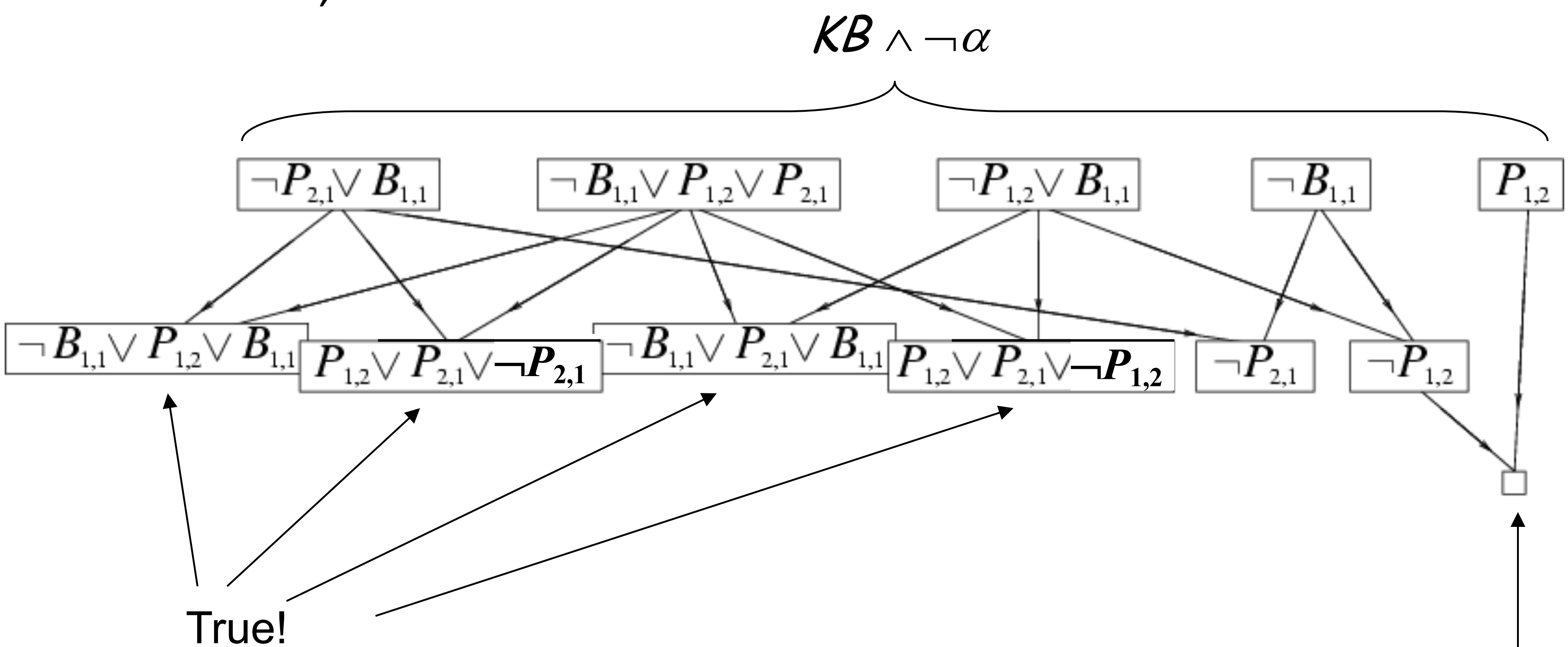
- Resolve $(\neg P_{1,2} \quad B_{1,1})$ and $(\neg\ B_{1,1})$ to give $(\neg P_{1,2})$
- Resolve $(\neg P_{1,2})$ and $(P_{1,2})$ to give ( )

- Consequently, the goal or query sentence is entailed by KB.

Resolution example

Graphical view of the proof

- $KB = (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1}$
- $\alpha = \neg P_{1,2}$



A sentence in KB is not "used up" when it is used in a resolution step. It is true, remains true, and is still in KB.

False in all worlds

Search

- States: current CNF KB + new clauses
- Operators: resolution
- Initial state: KB + negated goal
- Goal State: a database containing the empty clause
- Search using any search method

Exercise on applying resolution inference procedure

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

Prove:

1. $\Phi_1 \models \Phi_2$, by using the resolution inference procedure

$\Phi_1 \models \Phi_2$, by using the resolution inference procedure.

Write together

Exercise on applying resolution inference procedure

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

Prove:

1. $\Phi_1 \models \Phi_2$, by using the resolution inference procedure

$\Phi_1 \models \Phi_2$, by using the resolution inference procedure.
First of all we have to write Φ_1 and $\neg\Phi_2$ in CNF.

$$\text{CNF}(\Phi_1) = A \wedge (B \vee Q)$$

$$\text{CNF}(\neg\Phi_2) = \neg((A \wedge B) \vee (A \wedge Q)) = \neg(A \wedge B) \wedge \neg(A \wedge Q) = (\neg A \vee \neg B) \wedge (\neg A \vee \neg Q)$$

Then we apply resolution inference:

1. A
2. $B \vee Q$
3. $\neg A \vee \neg B$
4. $\neg A \vee \neg Q$

---

5. $\neg B$ (1,3)
6. $\neg Q$ (1,4)
7. $\neg A \vee Q$ (2,3)
8. $\neg A \vee B$ (2,4)

---

9. Q (1,7)
10. B (1,8)
11. $\neg A$ (3,8)

---

12. $\{\}$ (1,11)

Algorithms to check satisfiability

What could work here from what we have learned so far?

Algorithms to check satisfiability

Two families of efficient algorithms:

- Complete backtracking search algorithms:
  - E.g., DPLL algorithm
- Incomplete local search algorithms
  - E.g., `WalkSAT` algorithm

The DPLL algorithm

Determine if an input propositional logic sentence (in CNF) is satisfiable.
This is just model checking using CSP with backtracking search.

Improvements:

1. Early termination

    A clause is true if any literal is true.
    A sentence is false if any clause is false.

2. Pure symbol heuristic

    Pure symbol: always appears with the same "sign" in all clauses.
    e.g., In the three clauses $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A and B are pure, C is impure.

3. Unit clause heuristic

    Unit clause: only one in the clause
    The only literal in a unit clause must be true.
    For example, if the model contains B =true,
    then $(\neg B \vee \neg C)$ simplifies to $\neg C$

DPLL algorithm

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(*clauses*, *symbols*, { })

---

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

  if every clause in *clauses* is true in *model* **then return** *true*
  if some clause in *clauses* is false in *model* **then return** *false*
  P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  return DPLL(*clauses*, *rest*, *model* ∪ {P=*true*}) **or**
      DPLL(*clauses*, *rest*, *model* ∪ {P=*false*}))

Same exercise solved with the resolution inference procedure with DPLL

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

$\Phi_1 \models \Phi_2$, by using the DPLL algorithm.

First of all we have to write Φ_1 and $\neg\Phi_2$ in CNF.

$$\mathrm{CNF}(\Phi_1) = A \wedge (B \vee Q)$$

$$\mathrm{CNF}(\neg\Phi_2) = \neg((A \wedge B) \vee (A \wedge Q)) = \neg(A \wedge B) \wedge \neg(A \wedge Q) = (\neg A \vee \neg B) \wedge (\neg A \vee \neg Q)$$

1. Early termination
   A clause is true if any literal is true.
   A sentence is false if any clause is false. E.g., (A ∨ B) ∧ (A ∨ C) is true if A is true

2. Pure symbol heuristic
   Pure symbol: always appears with the same "sign" in all clauses.
   e.g., In the three clauses (A ∨ ¬B), (¬B ∨ ¬C), (C ∨ A), A and B are pure, C is impure.

3. Unit clause heuristic
   Unit clause: only one literal in the clause
   The only literal in a unit clause must be true.
   For example, if the model contains B =true,
   then (¬B ∨ ¬C) simplifies to ¬C, which is a unit clause

Same exercise solved with the resolution inference procedure with DPLL

Consider the following formulas in propositional logic:

$$\Phi_1 = A \wedge (B \vee Q)$$

$$\Phi_2 = (A \wedge B) \vee (A \wedge Q)$$

$\Phi_1 \models \Phi_2$, by using the DPLL algorithm.

The WalkSAT algorithm

- Incomplete, local search algorithm
  - In many problems, like the CSP ones, just a consistent assignment is enough
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses
- Balance between greediness and randomness

Walksat Procedure

Start with random initial assignment.

Pick a random unsatisfied clause.

Select and flip a variable from that clause:

  With probability p, pick a random variable.

  With probability 1-p, pick greedily

    a variable that minimizes the number of unsatisfied clauses

Repeat to predefined maximum number flips;
 if no solution found, restart.

Summary

- Resolution inference procedure starting from CNF sentences and the negated query sentence to prove unsatisfiability
- Model checking
  - DPLL: deterministic algorithm based on backtracking search
  - WalkSAT: local search algorithm, to find one satisfiable assignment

Next

- How to make the logic capable of expressing more than facts as in the propositional logic? <u>First order logic</u>