Overview

- Last time
  - Logic for KR in general; Propositional Logic; Natural Deduction

- Today
  - Entailment, satisfiability and validity
  - Normal forms
    - Negation normal form
    - Conjunctive normal form
  - Satisfiability as a search problem
    - DPLL algorithm

- Learning outcomes covered today:
  Distinguish the characteristics, and advantages and disadvantages, of the major knowledge representation paradigms that have been used in AI, such as production rules, semantic networks, propositional logic and first-order logic;
  Solve simple knowledge-based problems using the AI representations studied;

Propositional Logic for KR

- Given a knowledge base $KB$ and a property $\alpha$, check if $KB \models \alpha$
  - Model checking (e.g., use truth tables)
  - Theorem proving / inference: Prove $\alpha$ from $KB$
    - Natural deduction (last week)
    - Reduce to problems of validity or unsatisfiability
      - Davis-Putnam algorithm

Validity and Satisfiability

- A formula is said to be valid (or a tautology)
  - iff it is true under every interpretation.

- A formula is said to be satisfiable (or consistent)
  - iff it is true under at least one interpretation.

- A formula is said to be unsatisfiable (or inconsistent or contradictory)
  - iff it is not made true under any interpretation.

- $\varphi$ is a valid $\iff \neg\varphi$ is unsatisfiable.
Validity, Satisfiability and Entailment

Implications for Knowledge Representation...

• **Deduction Theorem:**
  \[ \text{KB} \models \alpha \text{ if and only if } (\text{KB} \Rightarrow \alpha) \text{ is valid} \]

Or...

• **Reductio ad absurdum:**
  \[ \text{KB} \models \alpha \text{ if and only if } (\text{KB} \land \neg \alpha) \text{ is unsatisfiable} \]

Satisfiability Checking

• Given a knowledge base \( \text{KB} \) and a property \( \alpha \), check if \( (\text{KB} \land \neg \alpha) \) is satisfiable
  - If not satisfiable, \( \alpha \) is implied by \( \text{KB} \)
  - Otherwise, an interpretation of propositions would give us a countermodel – shows how \( (\text{KB} \land \neg \alpha) \) can be made true

• Compare Prolog... “?- goal(x).” ↔ “\( \text{KB} \models \text{goal(X)} \)”
  - Prolog attempts to satisfy
    \[ \text{KB} \land \text{goal(X)} = \text{KB} \land \neg (\neg \text{goal(X)}) \]
    - not satisfiable: \( \text{KB} \models \neg \text{goal(X)} \)
    - satisfiable: countermodels to \( \neg \text{goal(X)} \), which are models of the goal.

Countermodel (I)

To check if
\[
(\text{hot} \land \text{smoky} \Rightarrow \text{fire})
\land (\text{alarm_beeps} \Rightarrow \text{smoky})
\land (\text{fire} \Rightarrow \text{switch_on_sprinklers})
\] is true under the interpretation \( I \):
\[
I(\text{hot}) = F
\]
\[
I(\text{smoky}) = T
\]
\[
I(\text{fire}) = F
\]
\[
I(\text{alarm_beeps}) = T
\]
\[
I(\text{switch_on_sprinklers}) = F
\]

Countermodel (II)

But
\[
(\text{hot} \land \text{smoky} \Rightarrow \text{fire})
\land (\text{alarm_beeps} \Rightarrow \text{smoky})
\land (\text{fire} \Rightarrow \text{switch_on_sprinklers})
\land \neg (\text{alarm_beeps} \Rightarrow \text{switch_on_sprinklers})
\] is true under the interpretation \( I \):
\[
I(\text{hot}) = F
\]
\[
I(\text{smoky}) = T
\]
\[
I(\text{fire}) = F
\]
\[
I(\text{alarm_beeps}) = T
\]
\[
I(\text{switch_on_sprinklers}) = F
\]
Example: Truth Tables and Satisfiability

Using a truth table show whether

\[(p \Rightarrow q) \lor (q \Rightarrow p)\]

is a tautology, consistent or inconsistent.

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>(p \Rightarrow q)</th>
<th>(q \Rightarrow p)</th>
<th>((p \Rightarrow q) \lor (q \Rightarrow p))</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Equivalent Sentences

- Two sentences A and B are equivalent, written \(A \equiv B\) iff A and B have the same truth values for every interpretation.
- Show

\[(p \Rightarrow q) \equiv (\neg p \lor q)\]

- Draw up a truth table for \((p \Rightarrow q)\) and \((\neg p \lor q)\) and check their truth values are the same.

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>(p \Rightarrow q)</th>
<th>(\neg p)</th>
<th>(\neg p \lor q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Efficiency

- So we are back to model checking...
  - A truth table contains \(2^n\) rows, construction requires \(2^n\) steps. . .
  - Can we do better than that?
    - Not in general: theory says that this is a very hard problem (Satisfiability checking is NP-complete)
    - In practice, not so bad, if we can use heuristics to identify lines we don’t need to check
    - But we have to transform \((KB \land \neg \alpha)\) into a normal form

Equivalent Sentences

- Two sentences A and B are equivalent, written \(A \equiv B\) iff A and B have the same truth values for every interpretation.
- Show

\[(p \Rightarrow q) \equiv (\neg p \lor q)\]

\(\equiv\) is not a 'logical operator' (as \(\equiv\)) but similar to entailment...

In fact: two sentences are equivalent \(\alpha \equiv \beta\) if and only if \(\alpha \models \beta\) and \(\beta \models \alpha\)
Equivalent Transformations (I)

• Where $A$, $B$ and $C$ are propositions or propositional formulae and $T$ and $F$ are true and false respectively

• Idempotent laws
  
  $A \land A \equiv A$
  $A \lor A \equiv A$

• Associative laws
  
  $(A \land B) \land C \equiv A \land (B \land C)$
  $(A \lor B) \lor C \equiv A \lor (B \lor C)$

• Commutative laws
  
  $A \land B \equiv B \land A$
  $A \lor B \equiv B \lor A$

• Distributive laws
  
  $A \land (B \lor C) \equiv (A \land B) \lor (A \land C)$
  $A \lor (B \land C) \equiv (A \lor B) \land (A \lor C)$

Examples

• Simplify the following expression: $\neg (\neg P \land \neg Q)$

  $\neg (\neg P \land \neg Q)$ given
  $\equiv (\neg \neg P \lor \neg \neg Q)$ de Morgan's laws
  $\equiv (P \lor Q)$ Complement laws

• Prove the following equivalence: $\neg (\neg (P \land Q) \lor P) = F$

  $\neg (\neg (P \land Q) \lor P)$ given
  $\equiv \neg (\neg P \lor \neg Q) \lor P$ de Morgan's laws
  $\equiv (P \land Q) \lor P$ Complement laws
  $\equiv (\neg Q \lor (\neg P \land P))$ Associative laws
  $\equiv (\neg Q \lor \neg P) \lor P$ Commutative laws
  $\equiv \neg (\neg P \lor \neg Q) \lor P$ de Morgan's laws
  $\equiv (P \land Q) \lor P$ Complement laws
  $\equiv \neg P \lor \neg Q \lor P$ Complement laws
  $\equiv \neg P \lor T$ Complement laws
  $\equiv F$ Complement laws

Equivalent Transformations (II)

• Identity laws
  
  $A \land T \equiv A$
  $A \lor F \equiv A$
  $A \land F \equiv F$
  $A \lor T \equiv T$

• Complement laws
  
  $A \land \neg A \equiv F$
  $A \lor \neg A \equiv T$
  $\neg (\neg A) \equiv A$
  $\neg T \equiv F$
  $\neg F \equiv T$

• de Morgan's laws
  
  $\neg (A \land B) \lor \neg (A \lor B) \equiv (\neg A \land \neg B) \lor \neg (\neg A \lor \neg B)$

• laws for $\Rightarrow$ and $\Leftrightarrow$
  
  $A \Rightarrow B \equiv \neg A \lor B$
  $A \Leftrightarrow B \equiv (A \Rightarrow B) \land (B \Rightarrow A)$

• We can use these laws to simplify expressions and to prove equivalences.

Negation Normal Form

• It is often useful to transform formulae into normal forms. These are logically equivalent formulae but have syntactically different forms that may be more suitable for reasoning with.

• There are several normal forms: Negation Normal Form, Clausal Form, Disjunctive Normal Form and Conjunctive Normal Form. We are most interested in Conjunctive Normal Form (CNF).

• but first: a formula is in negation normal form if negations appear only in front of propositions and the only operators are $\land$, $\lor$ and $\neg$.

  - First remove the $\Rightarrow$ and $\Leftrightarrow$ operators. Then apply de Morgan’s laws and remove double negations (complement laws), until in the correct form.

  - $\land$, $\lor$ can still be nested
Conjunctive Normal Form

- A formula is in *Conjunctive Normal Form* if it is of the form $A_1 \land A_2 \land \ldots A_k$ where each $A_i$ is a disjunction of propositions or their negations.

- Example
  
  \[(p \lor q) \land r \land (\neg p \lor \neg r \lor s)\] is in CNF.
  
  \[\neg(p \lor q) \land r \land (\neg p \lor \neg r \lor s)\] is not in CNF.
  
  \[(p \lor q) \land r \land (p \Rightarrow (\neg r \lor s))\] is not in CNF.

### Example

- Translate $\neg(p \lor q) \lor r \lor p \Rightarrow (q \lor r)$ into CNF.

- We first translate into NNF and obtain

  \[\neg(\neg(p \lor q) \lor r) \lor p \lor (q \lor r) \lor p\]

- Then transform into CNF

  \[(p \lor q) \land \neg r \land p \Rightarrow (\neg r \lor p)\]

### Exercise

- Convert the following into *Conjunctive Normal Form*, using the appropriate equivalence laws:

  \[p \leftrightarrow (q \lor r)\]
Exercise

• Convert the following into Conjunctive Normal Form, using the appropriate equivalence laws:

\[ p \iff (q \lor r) \]

\[
(p \implies (q \lor r)) \land ((q \lor r) \implies p) \quad \text{law for } \iff
\]

\[
(\neg p \lor q \lor r) \land (\neg(q \lor r) \lor p) \quad \text{law for } \implies
\]

\[
(\neg p \lor q \lor r) \land ((\neg q \land \neg r) \lor p) \quad \text{de Morgan's law}
\]

\[
(\neg p \lor q \lor r) \land (\neg q \lor p) \land (\neg r \lor p) \quad \text{distributive law}
\]

Satisfiability as a Search Problem

• Given a formula in CNF, can we find an assignment of truth values to propositions that satisfies it?

• States are partial assignments - some propositions get values, some are possibly unassigned.

• Actions are deciding whether a (yet unassigned) proposition is true or false.

• Initial state: empty partial assignment.

• Goal state: an assignment making the formula true.

Algorithm as Satisfiability for CNF

• A complete backtracking algorithm – Davis-Putnam paper (1960)

• The version presented (DPLL) is described in a paper by Davis, Logemann, and Loveland (1962).

• Naive Approach: every proposition is either true or false in any interpretation.

Search Space

• Consider \((\neg p \lor \neg r) \land (p \lor q) \land (r \lor \neg q)\)

Every path in the tree represents a (partial) assignment.
Partial Assignment

- Another idea: Simplify formula with a (previously guessed) partial assignment
  \((\neg p \lor \neg r) \land (p \lor q) \land (r \lor \neg q)\)
  
  { if we tried \(p = True\) we can now simplify:}
  \((\neg True \lor \neg r) \land (True \lor q) \land (r \lor \neg q)\)
  
  \((\neg False \lor \neg r) \land True \land (r \lor \neg q)\)
  
  {can only be true if \(r = False\)}
  \(\neg False \lor (False \lor \neg q)\)
  
  \(\neg q\)
  
  {can only be true if \(\neg q = True\) (so \(q = False\))}
  
  True

- Given a good order for partial assignments this can be very helpful. DPLL provides heuristics for choosing partial assignments. Only in the worst case we will need to try everything.

Likewise

- (-p \lor -r) \land (p \lor q) \land (r \lor -q)
  
  { let us now consider \(p = False\)}
  
  (-False \lor -r) \land (False \lor q) \land (r \lor -q)
  
  (True \lor -r) \land q \land (r \lor -q)
  
  True \land q \land (r \lor -q)
  
  q \land (r \lor -q)
  
  { can only be true if \(q = True\)}
  
  True \land (r \lor -True)
  
  (r \lor False)
  
  {r can only be true if \(r = True\)}
  
  True

Reduced Search Space

- Consider \((\neg p \lor \neg r) \land (p \lor q) \land (r \lor \neg q)\)

Algorithms Structure

- Search through possible assignments of propositions
- Simplify formulae with partial assignments
  - Unit clause propagation
  - Pure literal elimination
Unit Clause

- A clause with just one literal is called a **unit clause**
  - e.g. \([q] \land (\neg r \lor \neg q) \land (r \lor s)\)
- Literal's value can be **uniquely** assigned
  - \(q\) must be set to True

Unit clause propagation:
- Check if a formula in CNF has a unit clause, \(C\).
- Set the value of the literal in \(C\) such that \(C\) is True
  - Notice that some other clauses may become unit
  - e.g. \((p \lor q) \land (\neg q \lor \neg r) \land (\neg q \lor r)\) reduces to \((\neg q \lor \neg r) \land (\neg q \lor r)\)
- \(r\) must be set to False
  - \(\neg r \land (r \lor s)\) reduces to \(s\)

Pure Literal

- A **pure literal** is a literal that always appears with the same “sign” in all clauses.
  - e.g. \((p \lor q) \land (\neg q \lor \neg r) \land (\neg q \lor r)\)
- Making a pure literal True makes **some** clauses True, but no clause False
  - e.g. \((p \lor q) \land (\neg q \lor \neg r) \land (\neg q \lor r)\) reduces to \((\neg q \lor \neg r) \land (\neg q \lor r)\)
  - \((\neg q \lor \neg r) \land (\neg q \lor r)\) reduces to True
- Pure literal elimination:
  - Check if a formula in CNF has a pure literal, \(l\)
  - Set the value of \(l\) to True

DPPL(\(\varphi\))

- Given a propositional formula \(\varphi\), DPLL(\(\varphi\)) does the following:
  - If \(\varphi\) is True then return True;
  - If \(\varphi\) is False then return False;
  - Pick a symbol \(p\) in \(\varphi\)
    - If DPLL(Simplify(\(\varphi\), \(p\))) is True then return True;
    - If DPLL(Simplify(\(\varphi\), \(\neg p\))) is True then return True;
  - Return False;

Simplify(\(\varphi\), \(L\))

- Given a propositional formula \(\varphi\) and a literal \(l\), Simplify(\(\varphi\), \(L\)) does the following:
  - Delete every clause that contains \(l\) from \(\varphi\);
  - Delete \(\neg l\) from all clauses in \(\varphi\)
  - Apply exhaustively unit clause propagation and pure literal elimination;
  - Fail (return False) if positive and negative unit clauses for the **same** literal.
Applications

• There are now several efficient SAT solvers based on DPLL available on the web. They are used in a number of applications.
  – Hardware and software verification
    • IBM, Intel, . . .
  – Planning
  – Scheduling
  – . . .

Summary

• We have considered issues concerning efficiency in propositional reasoning
• This has covered equivalent transformations and normal forms
  – Negation normal form
  – Conjunctive normal form
• We have considered how satisfiability can be viewed as a search problem
  – Looked at an algorithm for satisfiability in CNF

• Next time
  – Proof method resolution for propositional logic