Problem Solving

Two methods for solving problems:

- **Procedural**
  - devise an algorithm
  - program the algorithm
  - execute the program

- **Declarative**
  - identify the knowledge needed
  - encode the knowledge in a representation (knowledge base - KB)
  - use logical consequences of KB to solve the problem

Proof Procedures

A logic consists of

- syntax: what is an acceptable sentence?
- semantics: what do the sentences and symbols mean?
- proof procedure: how do we construct valid proofs?

A proof: a sequence of sentences derivable using an inference rule

Logical Connectives

- **and** (conjunction) \( \land \)
- **or** (disjunction) \( \lor \)
- **not** (negation) \( \neg \)
- **if ... then ...** (implication) \( \rightarrow \)
- ... if and only if ... \( \leftrightarrow \)

Implication Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ( \rightarrow ) B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(A) If it rains, then I will carry an umbrella
**Implication Truth Table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

(A) If it rains, then I will carry an umbrella
(B) If you don’t study, then you will fail

**De Morgan’s Laws**

\[ A \lor B \equiv \neg(\neg A \land \neg B) \]

it rains OR I play football
not true that ( it doesn’t rain AND I don’t play football )

\[ A \land B \equiv \neg(\neg A \lor \neg B) \]

I’m a politician AND I lie
not true that ( I’m not a politician OR I tell the truth)

**If and only if Truth Table**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A ↔ B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

\[ A \leftrightarrow B \equiv (A \rightarrow B) \land (B \rightarrow A) \]

**Modus Ponens**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
<th>(A → B) \land A</th>
<th>((A → B) \land A) → B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</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>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Modus Ponens is a Tautology
If it’s raining then the grass is wet
it’s raining
therefore the grass is wet

**Modus Tolens**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
<th>(A → B) \land \neg B</th>
<th>((A → B) \land \neg B) → \neg A</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</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>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

Modus Tolens is a Tautology
If it’s raining then the grass is wet
the grass is not wet
therefore it’s not raining
Modus Bogus

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A → B</th>
<th>(A → B) ∧ B</th>
<th>((A → B) ∧ B) → A</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Modus Bogus is not a Tautology

If it’s raining then the grass is wet
the grass is wet
therefore it’s raining

A statement, A, is a logical consequence of a set of statements \{X\}, if A is true in every model of \{X\}.

If, for every set of truth assignments that hold for \{X\} (for every model of \{X\}), some other statement (A) is always true, then this other statement is a logical consequence of \{X\}.

Arguments and Models

P1: If I play hockey, then I’ll score a goal if the goalie is not good
P2: If I play hockey, the goalie is not good
D: Therefore, if I play hockey, I’ll score a goal

P: I play hockey
C: I’ll score a goal
H: the goalie is good

P1: P → (¬H → C)   P2: P → ¬H
D: P → C

Arguments and Models

P1: If I play hockey, then I’ll score a goal if the goalie is not good
P2: If I play hockey, the goalie is not good
D: Therefore, if I play hockey, I’ll score a goal

P: I play hockey
C: I’ll score a goal
H: the goalie is good

P1: P → (¬H → C)   P2: P → ¬H
D: P → C
Arguments and Models

<table>
<thead>
<tr>
<th>P</th>
<th>C</th>
<th>H</th>
<th>¬H → C</th>
<th>P1</th>
<th>P2</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Each row is an interpretation: an assignment of T/F to each proposition. In all the green lines, the premises are true: these interpretations are models of P1 and P2.

Every model of P1 and P2 is a model of D. Therefore, D is a logical consequence of P1 and P2: $P_1, P_2 \models D$.

Logical Consequence

P1: Elvis is Dead
P2: Elvis is Not Dead
D: Therefore, Gerry is Alive

Is this argument valid?
Yes!
E: Elvis is Alive
G: Gerry is Alive

An argument is valid if there is no situation in which the premises are all true, but the conclusions are false. But here, there is no model of the premises, so the argument is valid.

Deduction and Proof

Given a knowledge base, we want to prove things that are true. We can use
- Truth Table
- Natural Deduction
- Semantic Tableaux
- Axiomatic Logic (Modus Ponens)
  $((A \rightarrow B) \land A) \rightarrow B$
- Resolution Refutation (Reductio Ad Absurdum)
  $((-A) \land \ldots \land \ldots \rightarrow \bot) \rightarrow A$

Complete Knowledge

- A KB is a set of axioms
- A proof procedure is a way of proving theorems
  - KB ⊢ g means g can be derived from KB using the proof procedure
  - If KB ⊢ g, then g is a Theorem
  - A proof procedure is sound: if KB ⊢ g then KB |= g.
  - A proof procedure is complete: if KB |= g then KB ⊢ g.
- Two types of proof procedures: bottom up and top down
- We assume a closed world
  - the agent knows everything (or can prove everything)
  - if it can’t prove something, must be false
    - negation as failure
- Other option is an open world:
  - the agent doesn’t know everything
  - can’t conclude anything from a lack of knowledge
Bottom-up proof also known as forward chaining - start from facts and use rules to generate all possible atoms

\[
\begin{align*}
\text{rain} & \leftarrow \text{clouds} \land \text{wind}. \\
\text{clouds} & \leftarrow \text{humid} \land \text{cyclone}. \\
\text{clouds} & \leftarrow \text{near sea} \land \text{cyclone}. \\
\text{wind} & \leftarrow \text{cyclone}. \\
\text{near sea}. \\
\text{cyclone}. \\
\{\text{near sea}, \text{cyclone}\} \\
\{\text{near sea}, \text{cyclone}, \text{wind}\} \\
\end{align*}
\]

\[
\begin{align*}
C & := \{\}; \\
\text{repeat} & \\
& \quad \text{select } r \in KB \text{ such that} \\
& \quad \quad \cdot r \text{ is } h \leftarrow b_1 \land \ldots \land b_m \\
& \quad \quad \cdot b_i \in C \quad \forall i \\
& \quad \quad \cdot h \notin C \\
& \quad \quad C := C \cup \{h\} \\
& \text{until no more clauses can be selected}
\end{align*}
\]

Sound and Complete
Top-Down Proof

start from query and work backwards

rain $\leftarrow$ clouds $\land$ wind.
clouds $\leftarrow$ humid $\land$ cyclone.
clouds $\leftarrow near\_sea \land cyclone.$
wind $\leftarrow$ cyclone.
near\_sea.
cyclone.

yes $\leftarrow$ rain.
yes $\leftarrow$ clouds $\land$ wind
yes $\leftarrow near\_sea \land cyclone \land$ wind
yes $\leftarrow near\_sea \land cyclone \land cyclone

start from query and work backwards

rain $\leftarrow$ clouds $\land$ wind.
clouds $\leftarrow$ humid $\land$ cyclone.
clouds $\leftarrow near\_sea \land cyclone.$
wind $\leftarrow$ cyclone.
near\_sea.
cyclone.

yes $\leftarrow$ rain.
yes $\leftarrow$ clouds $\land$ wind
yes $\leftarrow near\_sea \land cyclone \land$ wind
yes $\leftarrow near\_sea \land cyclone \land cyclone
Top-Down Proof

start from query and work backwards

\[
\begin{align*}
\text{rain} & \leftarrow \text{clouds} \land \text{wind}. \\
\text{clouds} & \leftarrow \text{humid} \land \text{cyclone}. \\
\text{clouds} & \leftarrow \text{near\_sea} \land \text{cyclone}. \\
\text{wind} & \leftarrow \text{cyclone}. \\
\text{near\_sea}. \\
\text{cyclone}. \\
\text{yes} & \leftarrow \text{rain}. \\
\text{yes} & \leftarrow \text{clouds} \land \text{wind} \\
\text{yes} & \leftarrow \text{near\_sea} \land \text{cyclone} \land \text{wind} \\
\text{yes} & \leftarrow \text{near\_sea} \land \text{cyclone} \land \text{cyclone} \\
\text{yes} & \leftarrow \text{near\_sea} \land \text{cyclone} \\
\text{yes} & \leftarrow \text{cyclone} \\
\text{yes} & \leftarrow
\end{align*}
\]

Top-Down Interpreter

\[
\text{solve}(q_1 \land \ldots \land q_k):
\begin{align*}
\text{ac} & := \text{"yes} \leftarrow q_1 \land \ldots \land q_k" \\
\text{repeat}
\begin{align*}
\text{select} & \text{ a conjunct } q_i \text{ from body of ac} \\
\text{choose} & \text{ a clause } C \text{ from KB with } q_i \text{ as head} \\
& \text{ replace } q_i \text{ in body of ac by body of } C
\end{align*}
\text{until ac} \text{ is an answer}
\end{align*}
\]

select: “don’t care nondeterminism”
If one doesn’t give a solution, no point trying others!
any one will do, but be careful: some selections will lead more
quickly to solutions!

choose: “don’t know nondeterminism”
if one doesn’t give a solution, others may
have to do them all: can determine complexity of the problem.

Towards Automated Methods

A proof procedure gives us a method for deriving theorems
Therefore, given a knowledge base of assumptions, we can
‘prove’ things and know they are tautologies (they are logical
consequences of our knowledge base)
but ....
The method is difficult and requires some know-how - how could
we make it work more automatically?

Conjunctive Normal Form

A well-formed formula is in conjunctive normal form (CNF) if it is a
conjunction of disjunctions of atoms.

\[
(p_1 \lor p_2) \land (p_3 \lor p_4 \lor p_5) \land (p_6 \lor p_7 \lor \ldots) \ldots \land (p_{n-1} \lor p_n)
\]

Convert a propositional formula to CNF:
1. Eliminate \( \leftrightarrow \) using \( A \leftrightarrow B \equiv (A \rightarrow B) \land (B \rightarrow A) \)
2. Eliminate \( \rightarrow \) using \( A \rightarrow B \equiv \neg A \lor B \)
3. Use deMorgan’s laws to push \( \neg \) into atoms
4. Use \( \neg \neg A \equiv A \) to eliminate double negatives
5. use distributive law to complete
   \( A \lor (B \land C) \equiv (A \lor B) \land (A \lor C) \)
write
\[
(p_1 \lor p_2) \land (p_3 \lor p_4 \lor p_5) \land (p_6 \lor p_7 \lor \ldots) \ldots \land (p_{n-1} \lor p_n)
\]
as
\[
\{\{p_1, p_2\}, \{p_3, p_4, p_5\}, \{p_6, p_7, \ldots \}, \ldots, \{p_{n-1}, p_n\}\}
\]

Conjunctive Normal Form - Example 1

Refutation of Modus Ponens

\( A \land (A \rightarrow B) \vdash B \)

show a contradiction \( \bot \): means “false”
If our refutation leads to a [contradiction], it must be “false”, so
the conclusion must be true
\( A \land (A \rightarrow B) \land \neg B \vdash \bot \)
1. \( A \land (\neg A \lor B) \land (\neg B) \)
2. \{\{A\}, \{\neg A, B\}, \{\neg B\}\}
can already tell this is false since \( A \) must be true, so \( B \) must be
true, but \( B \) must be false
We will demonstrate using [resolution] on slide 28
Conjunctive Normal Form - Example 2

Transitivity of Implication

\(((A \rightarrow B) \land (B \rightarrow C)) \rightarrow (A \rightarrow C)\)

try to show a contradiction

\((A \rightarrow B) \land (B \rightarrow C) \land \neg(A \rightarrow C) \equiv \bot\)

1. \((\neg A \lor B) \land (\neg B \lor C) \land (\neg A \lor C)\)
2. \((\neg A \lor B) \land (\neg B \lor C) \land (\neg A \land \neg C)\)
3. \((\neg A \lor B) \land (\neg B \lor C) \land A \land \neg C\)
4. \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}
5. \{\{\neg A, B\}, \{\neg B, C\}, \{\neg A, C\}, \{A\}, \{\neg C\}\}
6. \{\{\neg A, B\}, \{\neg B, C\}, \{\neg A, C\}, \{A\}, \{C\}, \{\neg C\}\}
7. \bot

Resolution - Example 1

Refutation of Modus Ponens

\(A \land (A \rightarrow B) \equiv B\)

show a contradiction

\((A \land (A \rightarrow B)) \land \neg B \equiv \bot\)

1. \((A \land (\neg A \lor B)) \land (\neg B)\)
2. \{\{A\}, \{\neg A, B\}, \{\neg B\}\}
3. \{\{A\}, \{\neg A, B\}, \{B\}, \{\neg B\}\}
4. \bot

Resolution - Example 2

Transitivity of Implication (again)

\(((A \rightarrow B) \land (B \rightarrow C)) \rightarrow (A \rightarrow C)\)

try to show a contradiction

\((A \rightarrow B) \land (B \rightarrow C) \land \neg(A \rightarrow C) \equiv \bot\)

1. \((\neg A \lor B) \land (\neg B \lor C) \land (\neg A \lor C)\)
2. \((\neg A \lor B) \land (\neg B \lor C) \land (\neg A \land \neg C)\)
3. \((\neg A \lor B) \land (\neg B \lor C) \land A \land \neg C\)
4. \{\{\neg A, B\}, \{\neg B, C\}, \{A\}, \{\neg C\}\}
5. \{\{\neg A, B\}, \{\neg B, C\}, \{\neg A, C\}, \{A\}, \{\neg C\}\}
6. \{\{\neg A, B\}, \{\neg B, C\}, \{\neg A, C\}, \{A\}, \{C\}, \{\neg C\}\}
7. \bot

Resolution - Example 3

P1: If I play hockey, then I'll score a goal if the goalie is not good
P2: If I play hockey, the goalie is not good
D: If I play hockey, I'll score a goal

P: I play hockey, C: I'll score a goal, H: the goalie is good

P1: \(P \rightarrow (\neg H \rightarrow C)\)

P2: \(P \rightarrow \neg H\)

D: \(P \rightarrow C\)

test (refutation of D): \(P \land P \land \neg D\)

\[(P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H) \land \neg(P \land \neg D)\]

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)

\(((P \rightarrow (\neg H \rightarrow C)) \land (P \rightarrow \neg H)) \land \neg(P \land \neg D)\)
Combinatorial Search Problems

Many problems can be formulated as a CNF
- Satisfiability
- Logic circuits
- Gene decoding
- Scheduling
- Air traffic control
- ...

Constraint Satisfaction as CNF

A CSP variable $Y$ with domain $\{v_1, \ldots, v_k\}$ can be converted into $k$ Boolean variables $\{Y_1, \ldots, Y_k\}$ where $Y_j$ is true when $Y$ has value $v_j$ and false otherwise.

Thus, $k$ atoms $y_1, \ldots, y_k$ are used to represent the CSP variable

Constraints:
- exactly one of $y_1, \ldots, y_k$ must be true:
  - $y_i$ and $y_j$ cannot both be true when $i \neq j$: $\neg y_i \lor \neg y_j$ for $i < j$
  - at least one of the $y_i$ must be true: $y_1 \lor \ldots \lor y_k$
- There is a clause for each false assignment in each constraint that specifies which assignments are not allowed.
- Thus, if there are two variables $Y$ and $Z$, and a constraint $Y \neq Z$, then we have clauses $\neg y_i \lor \neg z_i$ for all $i$ (Assuming $Y$ and $Z$ have the same domains).

Example Delivery robot: activities $a, b$, times 1, 2, 3, 4.

Constraints:
\[(A \neq 2) \land (B \neq 1) \land (A < B)\]

We have 8 variables in the CNF:
\[a_1, a_2, a_3, a_4, b_1, b_2, b_3, b_4\]

where $a_i$ means $A = i$ is true and $b_i$ means $B = i$ is true.

Constraints saying that $A$ (and $B$) must be exactly one value:
- $\neg a_i \lor \neg a_j$ for $i < j$
- $a_1 \lor a_2 \lor a_3 \lor a_4$
- $\neg b_i \lor \neg b_j$ for $i < j$
- $b_1 \lor b_2 \lor b_3 \lor b_4$

Domain constraints $\neg a_2$ and $\neg b_1$

The binary constraint $A < B$ has one $\neg (a_i \land b_j)$ for all $j \leq i$

Beyond propositions: Individuals and Relations

Symbols: $M, I, U$

Axiom: $MI$

Rules:
- if $xI$ is a theorem, so is $xIU$
- $Mx$ is a theorem, so is $Mxx$
- in any theorem, $III$ can be replaced by $U$
- $UU$ can be dropped from any string

Starting from $MI$, can you generate $MU$? (use top-down or bottom-up)

MIU Puzzle

Next:

Planning under certainty (Poole & Mackworth 2nd ed. Chapter 6.1-6.4)

Supervised Learning (Poole & Mackworth 2nd ed. Chapter 7.1-7.6)