LOGICAL AGENTS.
INFERENCE RULES AND PROOFS
References

• We will use here a number of slides designed by:
  
  • Dr. Rina Dechter - University of California at Irvine
  
  • Dr. Sofus A. Macskassy – University of Southern California
    http://www-bcf.usc.edu/~macskass/CS561/Spring2011/
Inference Rules (Proofs)

• Proof methods divide (roughly) into two kinds:
  1) Model checking – truth table enumeration (the more facts we have, the bigger table should be built, and this becomes computationally inefficient – exponential with respect to the number of facts)
  2) Application of inference rules – legitimate generation of new sentences from the given sentences. In such a case, proof is a sequence of inference rule applications
Inference Rules

◊ Modus Ponens or Implication-Elimination: (From an implication and the premise of the implication, you can infer the conclusion.)

\[ \alpha \Rightarrow \beta, \alpha \quad \frac{}{\beta} \]

If \( \alpha \Rightarrow \beta \) is true and \( \alpha \) is true, then \( \beta \) is also true

◊ And-Elimination: (From a conjunction, you can infer any of the conjuncts.)

\[ \alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \quad \frac{}{\alpha_i} \]

If a conjunction is true, then all conjuncts are also true

◊ And-Introduction: (From a list of sentences, you can infer their conjunction.)

\[ \alpha_1, \alpha_2, \ldots, \alpha_n \quad \frac{}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n} \]

If some sentences are true, then their conjunction is also true

◊ Or-Introduction: (From a sentence, you can infer its disjunction with anything else at all.)

\[ \alpha_i \quad \frac{}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n} \]

If some sentence is true, then its disjunction with anything else is also true
InferenceRules

◊ Double-Negation Elimination: (From a doubly negated sentence, you can infer a positive sentence.)

\[ \frac{-\neg\alpha}{\alpha} \]

If any doubly negated sentence is true, then this sentence is also true

◊ Unit Resolution: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

\[ \frac{\alpha \lor \beta, \neg \beta}{\alpha} \]

If the disjunction of two sentences is true, but one of these sentences is false, then another sentence is true

◊ Resolution: (This is the most difficult. Because \( \beta \) cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

\[ \frac{\alpha \lor \beta, \neg \beta \lor \gamma}{\alpha \lor \gamma} \quad \text{or equivalently} \quad \frac{-\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{-\alpha \Rightarrow \gamma} \]

If the disjunction of two sentences is true, and disjunction of the negation of the second sentence with the third one is also true, then the disjunction of the first and third sentences is true either
Wumpus world: example

- **Facts:** Percepts TELL facts to the KB
  - smell at (1,1) and (2,1) \(\rightarrow S_{1,1} ; S_{2,1}\)

- **Rules:** if square not smelly then neither the square or adjacent square contain the wumpus
  - R1: \(S_{1,1} \Rightarrow W_{1,1} \wedge W_{1,2} \wedge W_{2,1}\)
  - R2: \(S_{2,1} \Rightarrow W_{1,1} \wedge W_{2,1} \wedge W_{2,2} \wedge W_{3,1}\)
  - ...

- **Inference:**
  - KB contains \(S_{1,1}\) then using **Modus Ponens** we infer
    \(W_{1,1} \wedge W_{1,2} \wedge W_{2,1}\)
  - Using **And-Elimination** we get: \(W_{1,1} \quad W_{1,2} \quad W_{2,1}\)
  - ...

Propositional inference: normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms.

**Conjunctive Normal Form (CNF—universal)**

Conjunction of disjunctions of literals clauses

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

**Disjunctive Normal Form (DNF—universal)**

Disjunction of conjunctions of literals terms

E.g., \((A \land B) \lor (A \land \neg C) \lor (A \land \neg D) \lor (\neg B \land \neg C) \lor (\neg B \land \neg D)\)
Disjunctive normal form

- A **Disjunctive Normal Form (DNF)** (a sum of products form) is a representation of a logical formula through a disjunction of elementary conjunctions.

- An **elementary conjunction** is a conjunction of propositions or their negations (no other logical operations are involved in the elementary conjunction). For example,

\[
\begin{align*}
\neg x_1 \land x_2 \land \neg x_3; & \quad x_1 \land x_2 \land x_3; & \quad \neg x_1 \land x_2 \land \neg x_3; & \quad x_1 \land \neg x_2 \land x_3
\end{align*}
\]
Full disjunctive normal form (FDNF)

- A Full Disjunctive Normal Form (FDNF) is such a DNF where each of its variables appears exactly once in every elementary conjunction.
- Any FDNF of $n$ variables contains the same amount of elementary conjunctions that the number of True values of the corresponding logical formula, and each variable appears exactly once in every elementary conjunction, either without negation or with negation.
Derivation of FDNF

• Example:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>RESULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

P \land (\neg Q)

(\neg P) \land Q

FDNF = (P \land (\neg Q)) \lor ((\neg P) \land Q)
Conjunctive normal form (CNF)

- A Conjunctive Normal Form (CNF) (a product of sums form) is a representation of a logical formula through a conjunction of elementary disjunctions.

- An elementary disjunction is a disjunction of propositions or their negations (no other logical operations are involved in the elementary disjunction). For example,

\[ x_1 \lor x_2 \lor \overline{x}_3; \quad x_1 \lor x_2 \lor x_3; \quad \overline{x}_1 \lor x_2 \lor \overline{x}_3; \quad x_1 \lor \overline{x}_2 \lor x_3 \]
Full conjunctive normal form (FCNF)

• A Full Conjunctive Normal Form (FDNF) is such a CNF where each of its variables appears exactly once in every elementary conjunction.
• An FCNF can be obtained from FDNF by applying distributive law to FDNF.
Derivation of FCNF

• Example:

\[
\begin{array}{ccc}
P & Q & \text{RESULT} \\
T & T & F \\
T & F & T \\
F & T & T \\
F & F & F \\
\end{array}
\]

FDNF = \((p \land \overline{q}) \lor (\overline{p} \land q)\)

FCNF = \((p \land q) \lor (p \land q) \equiv ((p \land q) \lor \overline{p}) \land ((p \land q) \lor q) \equiv ((p \lor \overline{p}) \land (q \lor \overline{p})) \land ((p \lor q) \land (q \lor q)) \equiv (q \lor \overline{p}) \land (p \lor q)\)
Conversion to CNF
(more examples)

1. \((P \rightarrow Q) \rightarrow \neg P \lor Q\)

• 1. Eliminate implications

1. \(\neg(\neg P \lor Q) \lor (\neg P \lor Q)\)

2. \(\neg(P \rightarrow Q) \lor (R \rightarrow P)\)

• 2. Reduce the scope of negation sign

1. \((P \land \neg Q) \lor (\neg P \lor Q) \equiv (\neg P \lor Q) \lor (P \land \neg Q)\)

2. \((P \land \neg Q) \lor (\neg R \lor P)\)

• 3. Convert to cnfs using the associative and distributive laws

1. \((\neg P \lor Q \lor P) \land (\neg P \lor Q \lor \neg Q)\)

1. \(Q \land \neg P\)

2. \((P \lor \neg R \lor P) \land (\neg Q \lor \neg R \lor P)\)

2. \((P \lor \neg R) \land (\neg Q \lor \neg R \lor P)\)
Resolution in Propositional Calculus

- **Resolution rule:**
- **Unit rule:** Resolving resolvent $P$ from $(P \lor Q) \land (P \lor \neg Q)$:
  $$(P \lor Q) \land (P \lor \neg Q) \vdash \neg P$$

- Resolving resolvents $(P \lor R)$ from $(P \lor Q) \land (R \lor \neg Q)$:
  $$(P \lor Q) \land (R \lor \neg Q) \vdash \neg (P \lor R)$$
  .... up to full resolution rule

- **Resolution is sound** (whenever $KB \vdash_r \alpha$, then $KB \models \alpha$)
  and **complete** (whenever $KB \models \alpha$, then $KB \vdash_r \alpha$)
Soundness of resolution

\[ KB = (\alpha \lor \beta) \land (\neg \beta \lor \gamma) \]

<table>
<thead>
<tr>
<th>(\alpha)</th>
<th>(\beta)</th>
<th>(\gamma)</th>
<th>(\alpha \lor \beta)</th>
<th>(\neg \beta \lor \gamma)</th>
<th>(\alpha \lor \gamma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

*Figure 6.14* A truth table demonstrating the soundness of the resolution inference rule. We have underlined the rows where both premises are true.

\[ KB \models \alpha \lor \gamma \]
Resolution

Conjunctive Normal Form (CNF—universal)

\[ \text{conjunction of } \text{disjunctions of literals} \]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

Resolution inference rule (for CNF): complete for propositional logic

\[
\frac{\ell_1 \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_n}{\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n}
\]

where \(\ell_i\) and \(m_j\) are complementary literals. E.g.,

\[
\frac{P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}}{P_{1,3}}
\]

Resolution is sound and complete for propositional logic
Resolution example: Wumpus World

- $KB = (B_{1,1} \iff (P_{1,2} \lor P_{2,1})) \land \neg B_{1,1} ; \quad \alpha = \neg P_{1,2}$
- $KB \models \alpha$

\[
(B_{11} \iff (P_{12} \lor P_{21})) \land \neg B_{11} \equiv (B_{11} \rightarrow (P_{12} \lor P_{21})) \land ((P_{12} \lor P_{21}) \rightarrow B_{11}) \land \neg B_{11} \equiv \equiv (\neg B_{11} \lor (P_{12} \lor P_{21})) \land (\neg (P_{12} \lor P_{21}) \lor B_{11}) \land \neg B_{11} \equiv \equiv (\neg B_{11} \lor (P_{12} \lor P_{21})) \land (\neg P_{12} \land \neg P_{21} \lor B_{11}) \land \neg B_{11} \equiv \equiv (\neg B_{11} \lor P_{12} \lor P_{21}) \land (\neg P_{12} \lor B_{11}) \land (\neg P_{21} \lor B_{11}) \land \neg B_{11}
\]
function PL-RESOLUTION(KB, α) returns true or false

    clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
    new ← {}
    loop do
        for each 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
Limitations of Propositional Logic

1. It is too weak, i.e., has very limited expressiveness:
   • Each rule has to be represented for each situation:
     e.g., “don’t go forward if the wumpus is in front of you” takes 64 rules

2. It cannot keep track of changes:
   • If one needs to track changes, e.g., where the agent has been before then we need a timed-version of each rule. To track 100 steps we’ll then need 6400 rules for the previous example.

   It's **hard to write and maintain** such a huge rule-base
   **Inference becomes intractable**

➢ To overcome these limitations, we should use **first-order logic**