Propositional Logic

Chapter 7

Outline

• Review
  – Knowledge-based agents
  – Logic in general
  – Propositional logic in particular – syntax and semantics

• Wumpus world

• Inference rules and theorem proving
  – Resolution
  – forward chaining
  – backward chaining
A simple knowledge-based agent

```plaintext
function KB-Agент( percept ) returns an action
    static: KB, a knowledge base
    t, a counter, initially 0, indicating time
    TELL( KB, MAKE-PERCEPT-SENTENCE( percept, t ) )
    action ← ASK( KB, MAKE-ACTION-QUERY( t ) )
    TELL( KB, MAKE-ACTION-SENTENCE( action, t ) )
    t ← t + 1
    return action
```

• The agent must be able to:
  – Represent states, actions, etc.
  – Incorporate new percepts
  – Update internal representations of the world
  – Deduce hidden properties of the world
  – Deduce appropriate actions

Logic in general

• **Logics** are formal languages for representing information such that conclusions can be drawn
• **Syntax** defines the sentences in the language
• **Semantics** define the "meaning" of sentences;
  – i.e., define truth of a sentence in a world

• E.g., the language of arithmetic
  – \( x+2 \geq y \) is a sentence; \( x2+y > \{ \} \) is not a sentence
  – \( x+2 \geq y \) is true iff the number \( x+2 \) is no less than the number \( y \)
  – \( x+2 \geq y \) is true in a world where \( x=7, y=1 \)
  – \( x+2 \geq y \) is false in a world where \( x=0, y=6 \)
Entailment

- **Entailment** means that one thing follows from another:
  \[ KB \models \alpha \]

- Knowledge base \( KB \) entails sentence \( \alpha \) if and only if \( \alpha \) is true in all worlds where \( KB \) is true
  - E.g., the KB containing “the Steelers won” and “the Bengals won” entails “Either the Steelers won or the Bengals won”
  - E.g., \( x+y = 4 \) entails \( 4 = x+y \)
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A \land B</th>
<th>A \land C</th>
<th>B \land C</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<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>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</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>T</td>
</tr>
</tbody>
</table>

- \( A\land C, C \) does not entail \( B\land C \)
- \( A,B, \text{ Entails } A\land B \)
Inference

- $KB \vdash \alpha = \text{sentence } \alpha \text{ can be derived from } KB \text{ by procedure } i$
- **Soundness**: $i$ is sound if whenever $KB \vdash \alpha$, it is also true that $KB \models \alpha$
- **Completeness**: $i$ is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash \alpha$
- Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.
- That is, the procedure will answer any question whose answer follows from what is known by the $KB$.

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas
- The proposition symbols $P_1, P_2$ etc are sentences
  - If $S$ is a sentence, $\neg S$ is a sentence (negation)
  - If $S_1$ and $S_2$ are sentences, $S_1 \land S_2$ is a sentence (conjunction)
  - If $S_1$ and $S_2$ are sentences, $S_1 \lor S_2$ is a sentence (disjunction)
  - If $S_1$ and $S_2$ are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)
  - If $S_1$ and $S_2$ are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)
Propositional Logic: Semantics
(truth tables for connectives)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( \neg P )</th>
<th>( P \land Q )</th>
<th>( P \lor Q )</th>
<th>( P \Rightarrow Q )</th>
<th>( P \Leftrightarrow Q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<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>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Wumpus World PEAS
description

- **Performance measure**
  - gold +1000, death -1000
  - -1 per step, -10 for using the arrow

- **Environment**
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter iff gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square

- **Sensors:** Stench, Breeze, Glitter, Bump, Scream
- **Actuators:** Left turn, Right turn, Forward, Grab, Release, Shoot
<table>
<thead>
<tr>
<th>Characterization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully Observable</td>
<td>No – only local perception</td>
</tr>
<tr>
<td>Deterministic</td>
<td>Yes – outcomes exactly specified</td>
</tr>
<tr>
<td>Episodic</td>
<td>No – sequential at the level of actions</td>
</tr>
<tr>
<td>Static</td>
<td>Yes – Wumpus and Pits do not move</td>
</tr>
<tr>
<td>Discrete</td>
<td>Yes</td>
</tr>
<tr>
<td>Single-agent?</td>
<td>Yes – Wumpus is essentially a natural feature</td>
</tr>
</tbody>
</table>
Wumpus World continued

- Main difficulty: Agent doesn’t know the configuration
- Reason about configuration
- Knowledge evolves as new percepts arrive and actions are taken.

Wumpus Example

<table>
<thead>
<tr>
<th></th>
<th>stench</th>
<th>[Wumpus]</th>
<th>stench</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glitter [gold]</td>
<td>stench, breeze</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[start]</td>
<td>breeze</td>
<td>[Pit]</td>
<td>breeze</td>
</tr>
</tbody>
</table>

0 0
Examples of reasoning

• If the player is in square (1, 0) and the percept is breeze, then there must be a pit in (0,0) or a pit in (2,0) or a pit in (1,1).
• If the player is now in (0,0) [and still alive], there is not a pit in (0,0).
• If there is no breeze percept in (0,0), there is no pit in (0,1)
• If there is also no breeze in (0,1) then there is no pit in (1,1).
• Therefore, there must be a pit in (2,0)

Exploring a wumpus world
Exploring a wumpus world

Exploring a wumpus world
Exploring a wumpus world

Exploring a wumpus world
Exploring a wumpus world

Exploring a wumpus world
Exploring a wumpus world

Entailment in the wumpus world

Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for $KB$ assuming only pits

3 Boolean choices $\Rightarrow$ 8 possible models
Wumpus models

\[ KB = \text{wumpus-world rules} + \text{observations} \]
Wumpus models

- $KB = wumpus$-world rules + observations
- $\alpha_1 = "[1,2] is safe"$, $KB \models \alpha_1$, proved by model checking

Wumpus models

- $KB = wumpus$-world rules + observations
Wumpus models

• $KB = \text{wumpus-world rules + observations}$
• $\alpha_2 = \text{"[2,2] is safe"}$, $KB \not\vdash \alpha_2$

Logical Representation of Wumpus

Is there a pit in $[i, j]$?
Is there a breeze in $[i, j]$?

Pits cause breezes in adjacent squares.
Some Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

...

"Pits cause breezes in adjacent squares"

$B_{1,1} \iff (P_{1,2} \lor P_{2,1})$

$B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

...

Inference-based agents in the wumpus world

A wumpus-world agent using propositional logic:

$\neg P_{1,1}$

$\neg W_{1,1}$

$B_{x,y} \iff (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y})$

$S_{x,y} \iff (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y})$

$W_{1,1} \lor W_{1,2} \lor ... \lor W_{4,4}$

$\neg W_{1,1} \lor \neg W_{1,2}$

$\neg W_{1,1} \lor \neg W_{1,3}$

...

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences
Truth tables for inference

<table>
<thead>
<tr>
<th>$B_{1,1}$</th>
<th>$B_{2,1}$</th>
<th>$P_{1,1}$</th>
<th>$P_{1,2}$</th>
<th>$P_{2,1}$</th>
<th>$P_{2,2}$</th>
<th>$P_{3,1}$</th>
<th>$KB$</th>
<th>$\alpha_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</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>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>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>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Inference by enumeration

- Depth-first enumeration of all models is sound and complete

```plaintext
function TT-ENTAILS?(KB, α) returns true or false
    symbols ← a list of the proposition symbols in KB and α
    return TT-CHECK-ALL(KB, α, symbols, [])

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
    if EMPTY?(symbols) then
        if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
        else return true
    else do
        P ← FIRST(symbols), rest ← REST(symbols)
        return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model) and
                              TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

- For $n$ symbols, time complexity is $O(2^n)$, space complexity is $O(n)$
Logical equivalence

- Two sentences are logically equivalent iff true in same models: \( \alpha \equiv \beta \) iff \( \alpha \vdash \beta \) and \( \beta \vdash \alpha \)

\[
(\alpha \land \beta) \equiv (\beta \land \alpha) \quad \text{commutativity of} \land \\
(\alpha \lor \beta) \equiv (\beta \lor \alpha) \quad \text{commutativity of} \lor \\
((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of} \land \\
((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of} \lor \\
\neg(\neg \alpha) \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta) \quad \text{de Morgan} \\
\neg(\alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta) \quad \text{de Morgan} \\
(\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of} \land \text{over} \lor \\
(\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of} \lor \text{over} \land
\]

Example Proof by Deduction

- Knowledge
  \[
  S1: B_{22} \iff (P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \quad \text{rule} \\
  S2: \neg B_{22} \quad \text{observation}
  \]

- Inferences
  \[
  S3: (B_{22} \Rightarrow (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})) \land \\
  \quad \quad ((P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \Rightarrow B_{22}) \quad S1, bi elim
  \]

S4:
S5:
S6:
S7:
Example Proof by Deduction

- Knowledge
  S1: $B_{22} \iff (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})$  \textit{rule}
  S2: $\neg B_{22}$  \textit{observation}

- Inferences
  S3: $(B_{22} \Rightarrow (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})) \land$
    $( (P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \Rightarrow B_{22})$  \textit{S1, bi elim}
  S4: $((P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \Rightarrow B_{22})$  \textit{S3, and elim}
  S5: $(\neg B_{22} \Rightarrow \neg(P_{21} \lor P_{23} \lor P_{12} \lor P_{32}))$  \textit{contrapos}
  S6: $\neg(P_{21} \lor P_{23} \lor P_{12} \lor P_{32})$  \textit{S2, S5, MP}
  S7: $\neg P_{21} \land \neg P_{23} \land \neg P_{12} \land \neg P_{32}$  \textit{S6, DeMorg}

Proof methods

- Proof methods divide into (roughly) two kinds:
  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
      Can use inference rules as operators in a standard search
    - Typically require transformation of sentences into a normal form
  - Model checking
    - truth table enumeration (always exponential in $n$)
    - improved backtracking, e.g., Davis–Putnam–Logemann–Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms
Resolution

Conjunctive Normal Form (CNF)

- conjunction of disjunctions of literals clauses
- E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

- **Resolution** inference rule (for CNF):

  \[
  \begin{array}{c}
  \ell_1 \lor \ldots \lor \ell_k \\
  \ell_1 \lor \ldots \lor \ell_{k-1} \lor \ell_k, \quad m_1 \lor \ldots \lor m_n
  \end{array}
  \]

  \[
  \Rightarrow \quad \ell_1 \lor \ldots \lor \ell_{k-1} \lor \ell_k \lor m_1 \lor \ldots \lor m_{n-1} \lor \ldots \lor m_n
  \]

  where \(\ell_i\) and \(m_j\) are complementary literals.

  - E.g., \(P_{1,3} \lor P_{2,2}, \quad \neg P_{2,2}\)

- Resolution is sound and complete for propositional logic

Resolution in Wumpus World

- There is a pit at 2,1 or 2,3 or 1,2 or 3,2
  - \(P_{2,1} \lor P_{2,3} \lor P_{1,2} \lor P_{3,2}\)

- There is no pit at 2,1
  - \(\neg P_{2,1}\)

- Therefore (by resolution) the pit must be at 2,3 or 1,2 or 3,2
  - \(P_{2,3} \lor P_{1,2} \lor P_{3,2}\)
Conversion to CNF

\[ B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \]

1. Eliminate \( \iff \), replacing \( \alpha \iff \beta \) with \( (\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha) \).

2. Eliminate \( \Rightarrow \), replacing \( \alpha \Rightarrow \beta \) with \( \neg \alpha \lor \beta \).

3. Move \( \neg \) inwards using de Morgan’s rules and double-negation:

4. Apply distributivity law (\( \land \) over \( \lor \)) and flatten:

\[ (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land (\neg (P_{1,2} \lor P_{2,1}) \lor B_{1,1}) \]
\[ B_{22} \iff (P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \]

1. Eliminate \( \iff \), replacing with two implications
   \[ (B_{22} \implies (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})) \land ((P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \implies B_{22}) \]

2. Replace implication \((A \implies B)\) by \(\neg A \lor B\)
   \[ (\neg B_{22} \lor (P_{21} \lor P_{23} \lor P_{12} \lor P_{32})) \land ((\neg P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \lor B_{22}) \]

3. Move \(\neg\) “inwards” (unnecessary parens removed)
   \[ (\neg B_{22} \lor P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \land ((\neg P_{21} \land \neg P_{23} \land \neg P_{12} \land \neg P_{32}) \lor B_{22}) \]

4. Distributive Law
   \[ (\neg B_{22} \lor P_{21} \lor P_{23} \lor P_{12} \lor P_{32}) \land ((\neg P_{21} \land \neg P_{23} \land \neg P_{12}) \land \neg P_{32} \lor B_{22}) \]

**Last Step**

- Sentences are now in CNF:
  - \((P_{1} \lor P_{2} \lor \neg P_{3}) \land P_{4} \land \neg P_{5} \land (P_{2} \lor P_{3})\)
- Create a separate clause corresponding to each conjunct
  - \(P_{1} \lor P_{2} \lor \neg P_{3}\)
  - \(P_{4}\)
  - \(\neg P_{5}\)
  - \(P_{2} \lor P_{3}\)
Resolution algorithm

- Proof by contradiction, i.e., show $KB \land \neg \alpha$ unsatisfiable

```plaintext
function PL-RESOLUTION(KB, \alpha) returns true or false
  clauses ← the set of clauses in the CNF representation of $KB \land \neg \alpha$
  new ← { }
  loop do
    for each $C_i, C_j$ in clauses do
      resolvents ← PL-RESOLVE($C_i, C_j$)
      if resolvents contains the empty clause then return true
      new ← new \cup resolvents
      if new ⊆ clauses then return false
      clauses ← clauses \cup new
  end loop
  return false
```

Simple Resolution Example

- When the agent is in 1,1, there is no breeze, so there can be no pits in neighboring squares
- Percept: $\neg B_{11}$
- Prove: $\neg P_{12}$. 
Resolution example

\[ KB = (B_{1,1} ⇔ (P_{1,2} ∨ P_{2,1})) ∧ ¬B_{1,1} \]

\[ α = ¬P_{1,2} \]
Forward and backward chaining

- **Horn Form** (restricted)
  - KB = conjunction of Horn clauses
  - Horn clause =
    - proposition symbol; or
    - (conjunction of symbols) ⇒ symbol
  - E.g., C ∧ (B ⇒ A) ∧ (C ∧ D ⇒ B)
- **Modus Ponens** (for Horn Form): complete for Horn KBs
  
  \[ \alpha_1, \ldots, \alpha_n, \alpha_1 \land \ldots \land \alpha_n \Rightarrow \beta \]

- Can be used with **forward chaining** or **backward chaining**.
- These algorithms are very natural and run in linear time

---

Forward chaining

- Idea: fire any rule whose premises are satisfied in the KB,
  - add its conclusion to the KB, until query is found

\[
\begin{align*}
P \Rightarrow Q \\
L \land M \Rightarrow P \\
B \land L \Rightarrow M \\
A \land P \Rightarrow L \\
A \land B \Rightarrow L \\
A \\
B
\end{align*}
\]
Forward chaining algorithm

```
function PL-FC-Entails?(KB, q) returns true or false
    local variables: count, a table, indexed by clause, initially the number of premises
    inferred, a table, indexed by symbol, each entry initially false
    agenda, a list of symbols, initially the symbols known to be true
    while agenda is not empty do
        p ← Pop(agenda)
        unless inferred[p] do
            inferred[p] ← true
            for each Horn clause c in whose premise p appears do
                decrement count[c]
                if count[c] = 0 then do
                    if Head[c] = q then return true
                    PUSH(Head[c], agenda)
        return false
```

- Forward chaining is sound and complete for Horn KB

Forward chaining example
Forward chaining example

Forward chaining example
Forward chaining example

Forward chaining example
Forward chaining example

Forward chaining example
Forward chaining example

Backward chaining

Idea: work backwards from the query $q$:

to prove $q$ by BC,
  check if $q$ is known already, or
  prove by BC all premises of some rule concluding $q$

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal
  1. has already been proved true, or
  2. has already failed
Backward chaining example

Backward chaining example
Backward chaining example

Backward chaining example
Backward chaining example

Backward chaining example
Backward chaining example

Backward chaining example
Backward chaining example

Backward chaining example
Forward vs. backward chaining

- FC is **data-driven**, automatic, unconscious processing,
  - e.g., object recognition, routine decisions
- May do lots of work that is irrelevant to the goal
- BC is **goal-driven**, appropriate for problem-solving,
  - e.g., Where are my keys? How do I get into a PhD program?
- Complexity of BC can be **much less** than linear in size of KB

Efficient propositional inference

Two families of efficient algorithms for propositional inference:

Complete backtracking search algorithms
- DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Incomplete local search algorithms
  - WalkSAT algorithm
Expressiveness limitation of propositional logic

• KB contains "physics" sentences for every single square

• For every time $t$ and every location $[x,y]$,  
  $L^t_{x,y} \land FacingRight^t \land Forward^t \Rightarrow L^t_{x+1,y}$

• Rapid proliferation of clauses

Summary

• Logical agents apply inference to a knowledge base to derive new information and make decisions

• Basic concepts of logic:
  – syntax: formal structure of sentences
  – semantics: truth of sentences wrt models
  – entailment: necessary truth of one sentence given another
  – inference: deriving sentences from other sentences
  – soundness: derivations produce only entailed sentences
  – completeness: derivations can produce all entailed sentences

• Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

• Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses

• Propositional logic lacks expressive power