7 LOGICAL AGENTS

- We now turn to knowledge-based agents that have a knowledge base KB at their disposal.
- With the help of the KB the agent aims at maintaining knowledge of its partially-observable environment and make inferences of the state of the world.
- Logic is used as the knowledge representation language.
- KB consists of a set of sentences.
- Initially the agent's KB contains the background knowledge given in advance.
- The agent TELLs the KB all its percepts and ASKs the KB for what actions to take.
- Both TELL and ASK may involve logical inference – deriving new sentences from old.

- Choosing an action based on the knowledge in the KB may involve extensive reasoning.
- Also the information about the executed action is stored in KB.
- Using a KB makes the agent amenable to a description at the knowledge level rather than giving a direct implementation for the agent.
- One can build a knowledge-based agent by simply TELLing it what it needs to know.
- Operating on the knowledge level corresponds to the declarative approach.
- In the procedural approach one encodes the desired behaviors directly as program code.
7.2 The Wumpus World

- An agent operates in a $4 \times 4$ grid of rooms always starting in the square $[1,1]$, facing to the right.
- In the cave of rooms there is also one static wumpus (a beast) and a heap of gold, in addition rooms can contain bottomless pits.
- The agent's possible actions are:
  - Move forward one square at a time,
  - Turn left or right by 90°,
  - Pick up the gold by grabbing, and
  - Shooting one single arrow in the direction it is facing.
- The agent dies a miserable death if it enters a square containing a pit or a live wumpus.
- The game ends in either the agent's death or picking up of gold.

- The locations of the gold and the wumpus are chosen randomly, with a uniform distribution, from the squares other than the start square $[1,1]$.
- In addition, each square other than the start $[1,1]$ can be a pit, with probability 0.2.

- In a square directly (not diagonally) adjacent to the wumpus ($W$) the agent ($A$) will perceive a stench ($S$) and in one adjacent to a pit ($P$) the agent will perceive a breeze ($B$).
- The glitter of gold is perceived in the same square, walking into a wall, the agent perceives a bump, and when the wumpus is killed, its woeful scream that can be perceived anywhere in the cave.
From the fact that there is no stench or breeze in \([1,1]\), the agent can infer that the neighboring squares \([1,2]\) and \([2,1]\) are free of dangers.

Moving forward to \([2,1]\) makes the agent detect a breeze, so there must be a pit in a neighboring square \([2,2]\) or \([3,1]\) or both.

At this point only square \([1,2]\) is a safe unvisited square.

The percept in square \([1,2]\) is stench; hence

- The wumpus cannot – by the rules of the game – be in \([1,1]\).
  It cannot be in square \([2,2]\) (or we would have detected a stench in \([2,1]\)), therefore the wumpus must be in \([1,3]\).
- The lack of breeze in \([1,2]\) implies that there is no pit in \([2,2]\), so this means it must be in \([3,1]\).
7.3 Logic

- The syntax of the sentences constituting the KB is specified by the chosen knowledge representation language.
- In logic, the semantics of the language defines the truth of each sentence with respect to each model (possible world).
- Sentence \( \beta \) follows logically from sentence \( \alpha \), \( \alpha \models \beta \), if and only if (iff) in every model in which \( \alpha \) is true, \( \beta \) is also true.
- In other words: if \( \alpha \) is true, then \( \beta \) must also be true.
- We say that the sentence \( \alpha \) entails the sentence \( \beta \).

- If a sentence \( \alpha \) is true in model \( m \), we say that \( m \) is a model of \( \alpha \).
- Let \( M(\alpha) \) denote the set of all models of \( \alpha \).
- Observe that \( \alpha \models \beta \) if and only if \( M(\alpha) \subseteq M(\beta) \).

Consider the squares \([1,2] \), \([2,2] \), and \([3,1] \) in the wumpus-world and the question whether they contain pits.
- This is a binary information, so there are \( 2^3 = 8 \) possible models for this situation.
Logic /2

• Because there is no breeze in [1,1] and in [2,1] there is a breeze, the models in which the KB is true are those that have a pit in [2,2] or [3,1] or both
• Let $a_1 = \text{"There is no pit in [1,2]"}$
• The three models of the KB together with the model that has no pit in any of the three squares are the models of the conclusion $a_1$
• In every model in which KB is true, $a_1$ is also true
• Hence, $KB \models a_1$; there is no pit in [1,2]

• Let $a_2$ be the conclusion “There is no pit in [2,2]”
• In some models in which the KB is true, $a_2$ is false
• Hence, $KB \not\models a_2$
• The agent cannot conclude that there is no pit in [2,2] (nor that there is a pit in [2,2])
The logical inference algorithm working as described above is called **model checking**, because it enumerated all possible models are to check that $\varphi$ is true in all models in which $\text{KB}$ is true.

If an inference algorithm $i$ can derive $\varphi$ from $\text{KB}$, we write $\text{KB} \vdash_i \varphi$.

An inference algorithm that derives only entailed sentences is called **sound** (or truth-preserving).

Another desired property of an inference algorithm is **completeness**: it can derive any sentence that is entailed.
7.4 Propositional Logic

- **Atomic sentences** consist of a single proposition symbol $P, Q, R, \ldots$

- Each such symbol stands for a proposition that can be true or false

- Proposition symbols with fixed meanings: $T$ is always true and $F$ is always false

- Complex sentences are constructed from simpler ones using logical connectives
  - $\neg$ **Negation**. A literal is either an atomic sentence (a positive literal) or a negated atomic sentence (a negative literal)
  - $\land$ (and) **Conjunction** the parts of which are conjunctions
  - $\lor$ (or) **Disjunction** the parts of which are disjunctions
  - $\Rightarrow$ **Implication** has a premise (or antecedent) and conclusion (or consequent)
  - $\iff$ (iff) **Equivalence** (or biconditional)

A BNF grammar of sentences in propositional logic:

- $\text{Sentence} \rightarrow \text{AtomicSentence} \mid \text{ComplexSentence}$
- $\text{AtomicSentence} \rightarrow T \mid F \mid \text{Symbol}$
- $\text{Symbol} \rightarrow P \mid Q \mid R \mid \ldots$
- $\text{ComplexSentence} \rightarrow \neg \text{Sentence}$
  - $\mid (\text{Sentence} \land \text{Sentence})$
  - $\mid (\text{Sentence} \lor \text{Sentence})$
  - $\mid (\text{Sentence} \Rightarrow \text{Sentence})$
  - $\mid (\text{Sentence} \iff \text{Sentence})$
To avoid using an excessive amount of parentheses, we agree the order of precedence for the connectives: \( \neg, \land, \lor, \Rightarrow, \Leftrightarrow \).

Hence, the sentence \( \neg P \lor Q \land R \Rightarrow S \) is equivalent to the sentence
\[
((\neg P) \lor (Q \land R)) \Rightarrow S
\]

The semantics of propositional logic defines the rules for determining the truth of a sentence with respect to a particular model.

In propositional logic, a model simply fixes the truth value for every proposition symbol.

\[ M_1 = \{ P_{1,2} = F, P_{2,2} = F, P_{3,1} = T \} \]

---

**Truth Table**

The truth value of an arbitrary sentence can be computed recursively:
- \( F \) is false and \( T \) true in every model.
- The model assigns a truth value to every proposition symbol.
- The value of a complex sentence is determined by truth table.

<table>
<thead>
<tr>
<th>( P )</th>
<th>( Q )</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>( F )</td>
<td>( F )</td>
<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>
<td>( T )</td>
<td>( F )</td>
</tr>
<tr>
<td>( T )</td>
<td>( F )</td>
<td>( F )</td>
<td>( F )</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>( T )</td>
</tr>
</tbody>
</table>
For example, the sentence $\neg P_{1,2} \land (P_{2,2} \lor P_{3,1})$ evaluated in $M_1$ gives $T \land (F \lor T) = T \land T = T$.

A logical knowledge base $KB$ which started as empty and has been constructed by operations $\text{Tell}(KB, S_1), \ldots, \text{Tell}(KB, S_n)$ is a conjunction of sentences $KB = S_1 \land \ldots \land S_n$.

We can, thus, treat knowledge bases and sentences interchangeably.

In the following the interpretation of proposition symbols is:
- $P_{i,j}$ is true if there is a pit in $[i, j]$.
- $B_{i,j}$ is true if there a breeze in $[i, j]$.

Knowledge Base (1)

Part of the background knowledge – i.e., the rules of the game – and the first percepts:

- $R_1$: $\neg P_{1,1}$
- $R_2$: $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
- $R_3$: $B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$
- $R_4$: $\neg B_{1,1}$
- $R_5$: $B_{2,1}$
• The aim of logical inference is to decide whether $KB \models \alpha$ for some sentence $\alpha$.
• For example, is $P_{2,2}$ entailed?

• In the Wumpus-world, while examining the first two squares, there are 7 relevant proposition symbols; hence, $2^7 = 128$ possible models.
• Only in three of these $KB = R_1 \land \ldots \land R_5$ is true.

• In those three models $\neg P_{1,2}$ is true and there is no pit in [1,2].
• On the other hand, $P_{2,2}$ is true in two of the three models and false in one, so we cannot yet tell whether there is a pit in [2,2].

• Model checking algorithm is sound, because it implements directly the definition of entailment.
• It is also complete, because it works for any $KB$ and $\alpha$ and always terminates since there are “only” finitely many models to check.

• If $KB$ and $\alpha$ contain $n$ symbols in all, then there are $2^n$ models and the time complexity of the algorithm is exponential.
• In fact, every known inference algorithm for propositional logic has a worst-case complexity that is exponential in the size of the input.
• Propositional entailment is co-NP-complete.
7.5 Propositional Theorem Proving

- Two sentences $\alpha$ and $\beta$ are logically equivalent, $\alpha \equiv \beta$, if they are true in the same models:
  $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

- $(P \land Q) \equiv (Q \land P)$

- A sentence is valid a.k.a. tautology if it is true in all models
  $P \lor \neg P$

- Every valid sentence is logically equivalent to $T$

- Deduction theorem: For any sentences $\alpha$ and $\beta$,
  $\alpha \models \beta$ iff the sentence $(\alpha \Rightarrow \beta)$ is valid

- We can think the model checking algorithm as checking the validity of $(KB \models \alpha)$

- A sentence is satisfiable if it is true in some model
- E.g., $KB = R_1 \land \ldots \land R_5$ is satisfiable because there are three models in which it is true (these three satisfy $KB$)

- Determining the satisfiability of sentences in propositional logic was the first problem proved to be NP-complete (Cook 1971)

- $\alpha$ is valid iff $\neg \alpha$ is unsatisfiable
- Contrapositively: $\alpha$ is satisfiable iff $\neg \alpha$ is not valid

- Proof by contradiction (refutation):
  $\alpha \models \beta$ iff the sentence $(\alpha \land \neg \beta)$ is unsatisfiable
  $\alpha \models \beta \iff (\alpha \Rightarrow \beta)$ is valid $\iff (\neg (\alpha \Rightarrow \beta))$ is unsatisfiable
  $\iff (\neg (\neg \alpha \lor \beta))$ is unsatisfiable $\iff (\alpha \land \neg \beta)$ is unsatisfiable
7.5.1 Inference and proofs

- **Modus Ponens**
  \[ \alpha \Rightarrow \beta, \quad \alpha \quad \Rightarrow \beta \]

- **And-elimination**
  \[ \alpha \land \beta \quad \Rightarrow \quad \alpha \]

- These two inference rules are sound once and for all, there is no need to enumerate models, the rules can be used directly.
- The following well-known logical equivalences each give two inference rules:
  - We cannot, though, run Modus Ponens to opposite direction.

### Standard logical equivalences

<table>
<thead>
<tr>
<th>Logical equivalence</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>((\alpha \land \beta) \equiv (\beta \land \alpha))</td>
<td>Commutativity of (\land)</td>
</tr>
<tr>
<td>((\alpha \lor \beta) \equiv (\beta \lor \alpha))</td>
<td>Commutativity of (\lor)</td>
</tr>
<tr>
<td>(((\alpha \land \beta) \land \gamma) \equiv (\alpha \land (\beta \land \gamma)))</td>
<td>Associativity of (\land)</td>
</tr>
<tr>
<td>(((\alpha \lor \beta) \lor \gamma) \equiv (\alpha \lor (\beta \lor \gamma)))</td>
<td>Associativity of (\lor)</td>
</tr>
<tr>
<td>(\neg(\neg \alpha) \equiv \alpha)</td>
<td>Double-negation elimination</td>
</tr>
<tr>
<td>((\alpha \Rightarrow \beta) \equiv (\neg \beta \Rightarrow \neg \alpha))</td>
<td>Contraposition</td>
</tr>
<tr>
<td>((\alpha \Rightarrow \beta) \equiv (\neg \alpha \lor \beta))</td>
<td>Implication elimination</td>
</tr>
<tr>
<td>((\alpha \Leftrightarrow \beta) \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)))</td>
<td>Biconditional elimination</td>
</tr>
<tr>
<td>(\neg(\neg \alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta))</td>
<td>de Morgan</td>
</tr>
<tr>
<td>(\neg(\neg \alpha \lor \beta) \equiv (\neg \alpha \land \neg \beta))</td>
<td>de Morgan</td>
</tr>
<tr>
<td>((\alpha \land (\beta \lor \gamma)) \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)))</td>
<td>Distributivity of (\land) over (\lor)</td>
</tr>
<tr>
<td>((\alpha \lor (\beta \land \gamma)) \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)))</td>
<td>Distributivity of (\lor) over (\land)</td>
</tr>
</tbody>
</table>
Let us prove that there is no pit in \([1,2] \rightarrow \neg P_{1,2}\)

By applying biconditional elimination to \(R_2\) we get

\[ R_6: (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]

Then we apply and-elimination to \(R_6\) to obtain

\[ R_7: ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]

Logical equivalence for contrapositives gives

\[ R_8: (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \]

Now we can apply Modus Ponens with \(R_8\) and the percept \(R_4: \neg B_{1,1}\) to obtain

\[ R_9: \neg (P_{1,2} \lor P_{2,1}) \]

Finally, we apply de Morgan’s rule, giving the conclusion

\[ R_{10}: \neg P_{1,2} \land \neg P_{2,1} \]

**Knowledge Base (2)**

\[ R_1: \neg P_{1,1} \]
\[ R_2: B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1}) \]
\[ R_3: B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \]
\[ R_4: \neg B_{1,1} \]
\[ R_5: B_{2,1} \]
\[ R_6: (B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]
\[ R_7: ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1}) \]
\[ R_8: (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \]
\[ R_9: \neg (P_{1,2} \lor P_{2,1}) \]
\[ R_{10}: \neg P_{1,2} \land \neg P_{2,1} \]
• Because inference in propositional logic is NP-complete, in the worst case, searching for a proof is going to be no more efficient than enumerating models.

• In practice, however, the possibility of ignoring irrelevant propositions can make finding a proof highly efficient. E.g., in the proof above the goal proposition $P_{1,2}$ appears only in sentence $R_2$, the other propositions mentioned in $R_2 - B_{1,1} \land P_{2,1} -$ appear additionally only in sentence $R_4$.

• Therefore, the symbols $B_{2,1}, P_{1,1}, P_{2,2},$ and $P_{3,1}$ mentioned in sentences $R_1, R_3,$ and $R_5$ have no bearing on the proof.

• Monotonicity of logic: the number of entailed sentences can only increase as information is added to the knowledge base.

• For any sentences $\alpha$ and $\beta$,
  
  if $KB \models \alpha$ then $KB \land \beta \models \alpha$

• Additional assertion $\beta$ might help to draw additional conclusions, but it cannot invalidate any conclusion $\alpha$ already inferred.

• For example, $\beta$ could be the information that there are exactly eight pits in the world.

• Nevertheless, it cannot invalidate any conclusion $\alpha$ already made, such as that there is no pit in $[1,2]$.

• Monotonicity means that inference rules can be applied whenever suitable premises are found in the KB.

• The conclusion of the rule must follow regardless of what else is in the KB.