7.5.1 Inference and proofs

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

- 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>Expression</th>
<th>Name</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(\alpha \land \beta) \equiv (\neg \alpha \lor \neg \beta))</td>
<td>De Morgan</td>
</tr>
<tr>
<td>(\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] \((-P_{1,2}\))

By applying biconditional elimination to \(R_2\): \(B_{1,1} \iff (P_{1,2} \lor P_{2,1})\) we get
\[
R_6: (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})
\]

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
\[
R_{10}: \neg P_{1,2} \land \neg P_{2,1}
\]
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 $R_2 - B_{1.1}$ and $P_{2.1}$ 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.
7.5.2 Proof by Resolution

- The inference rules covered so far are sound
- Combined with any complete search algorithm they also constitute a complete inference algorithm
- However, removing any one inference rule will lose the completeness of the algorithm
- Resolution is a single inference rule that yields a complete inference algorithm when coupled with any complete search algorithm
- Restricted to two-literal clauses, the resolution step is
  \[
  \frac{\ell_1 \lor \ell_2, \neg \ell_2 \lor \ell_3}{\ell_1 \lor \ell_3}
  \]

When the agent in the wumpus-world moves to examine \([1,2]\) for the first time, it perceives a stench, but no breeze:

- \(R_{13}: \neg B_{1,2}\) and
- \(R_{10}: R_{1,2} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{1,3})\)

- By the same process that led to \(R_{10}\), we can derive the absence of pits in \([2,1]\) and \([1,3]\):
  - \(R_{13}: \neg P_{2,2}\) and \(R_{14}: \neg P_{1,3}\)

- Biconditional elimination to \(R_{6}: B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})\), followed by Modus Ponens with \(R_{5}: B_{2,1}\) yields
  - \(R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1}\)

- Resolution rule applied to \(R_{13}\) and \(R_{15}\) gives:
  - \(R_{16}: P_{1,1} \lor P_{3,1}\)

- Similarly, the literal \(\neg P_{1,1}\) in \(R_{1}\) resolves with the literal \(P_{1,1}\) in \(R_{16}\) to give
  - \(R_{17}: P_{3,1}\)
Knowledge Base (3)

\[ R_1: \neg P_{1,1} \]
\[ R_2: B_{1,1} \iff (P_{1,1} \lor P_{2,1}) \]
\[ R_3: B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \]
\[ R_4: \neg B_{1,1} \]
\[ R_5: B_{2,1} \]
\[ R_6: (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \]
\[ R_7: (P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1} \]
\[ R_8: (\neg B_{1,1} \Rightarrow \neg (P_{1,2} \lor P_{2,1})) \]
\[ R_9: \neg (P_{1,2} \lor P_{2,1}) \]
\[ R_{10}: \neg P_{2,2} \land \neg P_{2,1} \]
\[ R_{11}: \neg B_{1,2} \]
\[ R_{12}: B_{1,2} \iff (P_{1,1} \lor P_{2,2} \lor P_{1,3}) \]
\[ R_{13}: \neg P_{2,2} \]
\[ R_{14}: \neg P_{1,3} \]
\[ R_{15}: P_{1,1} \lor P_{2,2} \lor P_{3,1} \]
\[ R_{16}: P_{1,1} \lor P_{3,1} \]
\[ R_{17}: P_{3,1} \]

Above we applied the unit resolution inference rule:

\[ \ell_1 \lor \cdots \lor \ell_k, \quad m \]
\[ \ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \]

where \( \ell_i \) and \( m \) are complementary literals (one is the negation of the other).

In the general form the resolution rule allows to have an arbitrary number of complementary literals.

- **Factoring:** one must remove duplicate literals from the resulting clause
  
  \[
  \frac{A \lor B \quad A \lor \neg B}{A}
  \]
Conjunctive normal form

- Resolution is a sound inference rule, and it is also complete
- Though, given that \( A \) is true, we cannot use resolution to automatically generate the consequence \( A \lor B \)
- However, we can use resolution to answer the question whether \( A \lor B \) is true

- To show that \( KB \not\models \alpha \), we show that \((KB \land \neg \alpha)\) is unsatisfiable
- First \((KB \land \neg \alpha)\) is converted into conjunctive normal form (CNF) and the resolution rule is applied to the resulting clauses
- A CNF formula is a conjunction of disjunctions of literals (clauses) \( (\ell_{1,1} \lor \cdots \lor \ell_{1,k}) \land \cdots \land (\ell_{n,1} \lor \cdots \lor \ell_{n,k}) \)
- Every sentence of propositional logic is logically equivalent to a CNF formula

Let us convert the sentence \( R_2: B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \) into CNF

- Eliminate \( \iff \):
  \[
  (B_{1,1} \Rightarrow (P_{1,2} \lor P_{2,1})) \land ((P_{1,2} \lor P_{2,1}) \Rightarrow B_{1,1})
  \]

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

- Move \( \neg \) inwards:
  \[
  (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \land \neg P_{2,1}) \lor B_{1,1})
  \]

- Apply distributivity law to yield \( R_2 \) finally in CNF:
  \[
  (\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}) \land ((\neg P_{1,2} \lor B_{1,1}) \land (\neg P_{2,1} \lor B_{1,1})
  \]
A resolution algorithm

- First \((KB \land \neg \alpha)\) is converted into CNF and the resolution rule is applied to the resulting clauses.
- Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it not already present.
- Eventually,
  - there are no new clauses that can be added, in which case the \(KB\) does not entail \(\alpha\), or
  - an application of the resolution rule derives the empty clause (= \(F\)), in which case \(KB \not\vdash \alpha\).
- Resolution rule applied to a contradiction \(P \land \neg P \equiv F\) yields the empty clause.

7.5.3 Horn clauses and definite clauses

- In real-world knowledge bases it is often enough to use implication rules, like \((L_{1,1} \land Breeze) \Rightarrow B_{1,1}\).
- The premise of the rule, which is a conjunction of positive literals, is called the body of the clause.
- Also the conclusion is non-negated; it is called the head of the clause.
- If a Horn clause has no body at all, it is called a definite clause or a fact.
- The Horn clause \(P_1 \land \cdots \land P_n \Rightarrow Q\) is logically equivalent with disjunction \((\neg P_1 \lor \cdots \lor \neg P_n \lor Q)\).
- Thus, one allows that at most one of the literals is a positive proposition.
7.5.4 Forward and backward chaining

- Inference with Horn clauses can be done through the forward chaining or backward chaining algorithm.
- Both directions for inference are very natural.
- Deciding entailment with Horn clauses can be done in time that is linear in the size of the knowledge base.
- Forward chaining (data-driven reasoning) examines the body of which rules is satisfied by the known facts and adds their heads as new facts to the knowledge base.
- Every inference is essentially an application of Modus Ponens.
- This process continues until the given query $q$ is added or until no further inferences can be made.

AND-OR graph

$P \Rightarrow Q$
$L \land M \Rightarrow P$
$B \land L \Rightarrow M$
$A \land P \Rightarrow L$
$A \land B \Rightarrow L$
$A$
$B$
• Forward chaining is a sound and also a complete inference algorithm
• Backward chaining is a form of goal-directed reasoning
• We aim at showing the truth of a given query \( q \) by considering those implications in the knowledge base that conclude \( q \) (have it as the head)
• By backward chaining one examines whether all the premises of one of those implications can be proved true
• Backward chaining touches only relevant facts, while forward chaining blindly produces all facts
• One can limit forward chaining to the generation of facts that are likely to be relevant to queries that will be solved by backward chaining

7.7 Agents Based on Propositional Logic

• Already in our extremely simple wumpus-world it turns out that using propositional logic as the knowledge representation language suffers from serious drawbacks
• To initialize the knowledge base with the rules of the game, we have to give for each square \([x, y]\) the rule about perceiving a breeze (and the corresponding rule for perceiving a stench)
  \[ B_{x,y} \Leftrightarrow (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}) \]
• There is at least one wumpus:
  \[ W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,3} \lor W_{4,4} \]
and, on the other hand, at most one wumpus, which can be expressed by sentences such as
  \[ \neg W_{1,1} \lor \neg W_{1,2} \]
• When we add rules $\neg P_{1,1}$ and $\neg W_{1,1}$ to the KB, the simple $4 \times 4$ world has required in total $2 + 2 \cdot 16 + 1 + 120 = 155$ initial sentences containing 64 symbols.

• Model checking should enumerate $2^{64} \approx 1.8 \times 10^{19}$ possible models.

• More efficient inference algorithms, though, can take advantage of propositional knowledge representation.

• In addition, the KB does not yet contain all the rules of the game (e.g., arrow, gold, walls) and no knowledge of the possible actions of the agent (direction the agent is facing, taking steps), so it cannot yet be used to choose actions.

• In addition to being inefficient, propositional logic is also an intellectually unsatisfactory choice.

8 FIRST-ORDER LOGIC

Sentence $\rightarrow$ AtomicSentence $|$ (Sentence Connective Sentence) $|$ Quantifier VariableList: Sentence $|$ $\neg$ Sentence

AtomicSentence $\rightarrow$ Predicate $|$ Predicate(TermList) $|$ Term = Term

TermList $\rightarrow$ Term $|$ Term,TermList

Term $\rightarrow$ Function(TermList) $|$ Constant $|$ Variable

Connective $\rightarrow$ $\land$ $|$ $\lor$ $|$ $\Rightarrow$ $|$ $\Leftrightarrow$

Quantifier $\rightarrow$ $\exists$ $|$ $\forall$

VariableList $\rightarrow$ Variable $|$ Variable,VariableList

Constant $\rightarrow$ A $|$ X $|$ John $|$ Mary $|$ ...

Variable $\rightarrow$ a $|$ x $|$ s $|$ ...

Predicate $\rightarrow$ T $|$ F $|$ Before $|$ HasColor $|$ Raining $|$ ...

Function $\rightarrow$ Mother $|$ LeftLeg $|$ ...
In the realm of first-order predicate logic there are **objects**, which have **properties**, and between which there are **relations**.

- Constant symbols stand for objects,
- Predicate symbols stand for relations, i.e., relations of arity $n$, \( \text{Mother(John, Mary)} \),
- and properties are unary relations, \( \text{Student(John)} \).

A function is a total mapping that associates one single value to the ordered collection of its arguments.

Quantifiers let us express properties of entire collections of objects, instead of enumerating the objects by name.

We have the possibility to quantify objects universally and existentially:

\[ \forall y: \text{Student(y)} \Rightarrow \text{EagerToLearn(y)} \]
\[ \exists x: \text{Father(John)} = x \]

To determine the semantics for the syntactic notions (constants, predicates, and functions) need to be bound with an **interpretation** to corresponding (real-)world objects.

The world together with the interpretation of the syntax constitutes the model in predicate logic.
- Term $f(t_1, \ldots, t_n)$ refers to the object that is the value of function $F$ applied to objects $d_1, \ldots, d_n$, where $F$ is the interpretation of $f$ and $d_1, \ldots, d_n$ are the objects that the argument terms $t_1, \ldots, t_n$ refer to.

- Atomic sentence $P(t_1, \ldots, t_n)$ is true if the relation referred to by the predicate symbol $P$ holds among the objects referred to by the arguments $t_1, \ldots, t_n$.

- The semantics of sentences formed with logical connectives is identical to that in propositional logic.

- The sentence $\exists x: \varphi(x)$ is true in a given model under a given interpretation if sentence $\varphi$ is true by assigning some object as the interpretation of variable $x$.

- The sentence $\forall x: \varphi(x)$ is true in a given model under a given interpretation if sentence $\varphi$ is true by assigning any object as the interpretation of variable $x$.

- Because $\forall$ is really a conjunction over the universe of objects and $\exists$ is a disjunction, they obey de Morgan’s rules:

  $\forall x: \neg \varphi(x) \leftrightarrow \neg \exists x: \varphi(x)$

  $\neg \forall x: \varphi(x) \leftrightarrow \exists x: \neg \varphi(x)$

  $\forall x: \varphi(x) \leftrightarrow \exists x: \neg \varphi(x)$

  $\neg \forall x: \neg \varphi(x) \leftrightarrow \exists x: \varphi(x)$

- Equality $=$ is a special relation, whose interpretation cannot be changed.

- Terms $t_1$ and $t_2$ have this relation if and only if they refer to the same object.
The kinship domain

\( \forall m, c: \text{Mother}(c) = m \iff \text{Female}(m) \land \text{Parent}(m, c). \)

\( \forall w, h: \text{Husband}(h, w) \iff \text{Male}(h) \land \text{Spouse}(h, w). \)

\( \forall x: \text{Male}(x) \iff \neg \text{Female}(x). \)

\( \forall p, c: \text{Parent}(p, c) \iff \text{Child}(c, p). \)

\( \forall g, c: \text{Grandparent}(g, c) \iff \exists p: \text{Parent}(g, p) \land \text{Parent}(p, c). \)

\( \forall x, y: \text{Sibling}(x, y) \iff x \neq y \land \exists p: \text{Parent}(p, x) \land \text{Parent}(p, y). \)

Peano axioms

- Define natural numbers and addition using one constant symbol \(0\) and a successor function \(S\):

\[
\begin{align*}
\text{NatNum}(0). \\
\forall n: \text{NatNum}(n) \Rightarrow \text{NatNum}(S(n)).
\end{align*}
\]

- So the natural numbers are \(0, S(0), S(S(0)), \ldots\)

- Axioms to constrain the successor function:

\[
\begin{align*}
\forall n: 0 \neq S(n). \\
\forall m, n: m \neq n \Rightarrow S(m) \neq S(n).
\end{align*}
\]

- Addition in terms of the successor function:

\[
\begin{align*}
\forall m: \text{NatNum}(m) \Rightarrow +(m, 0) = m. \\
\forall m, n: \text{NatNum}(m) \land \text{NatNum}(n) \Rightarrow +(S(m), n) = S(+{(m, n)})
\end{align*}
\]
Axioms for sets

To define sets we use:
- Constant symbol $\emptyset$, which refers to the empty set
- Unary predicate $\text{Set}$, which is true of sets
- Binary predicates $x \in s$ and $s_1 \subseteq s_2$
- Binary function are $s_1 \cup s_2$, $s_1 \cap s_2$, and $\{x|s\}$, which refers to the set resulting from adjoining element $x$ to set $s$

The only sets are the empty set and those made by adjoining something to a set:
$$\forall s: \text{Set}(s) \iff s = \emptyset \lor \exists x, s_2: \text{Set}(s_2) \land s = \{x|s_2\}$$

The empty set has no elements adjoined into it:
$$\neg \exists x, s: \{x|s\} = \emptyset$$

Adjoining an element already in the set has no effect:
$$\forall x, s: x \in s \iff s = \{x|s\}.$$  

The only members of a set are those elements that were adjoined into it:
$$\forall x, s: x \in s \iff \exists y, s_2: (s = \{y|s_2\} \land (x = y \lor x \in s_2)).$$

Set inclusion:
$$\forall s_1, s_2: s_1 \subseteq s_2 \iff (\forall x: x \in s_1 \Rightarrow x \in s_2).$$

Two sets are equal iff each is a subset of the other:
$$\forall s_1, s_2: (s_1 = s_2) \iff (s_1 \subseteq s_2 \land s_2 \subseteq s_1).$$  

$$\forall x, s_1, s_2: x \in (s_1 \cap s_2) \iff (x \in s_1 \land x \in s_2)$$  

$$\forall x, s_1, s_2: x \in (s_1 \cup s_2) \iff (x \in s_1 \lor x \in s_2).$$
9 INFERENCE IN FIRST-ORDER LOGIC

- By eliminating the quantifiers from the sentences of predicate logic we reduce them to sentences of proposition logic and can turn to familiar inference rules.
- We may substitute the variable \( v \) in an universally quantified sentence \( \forall v: \alpha \) with any ground term, a term without variables.
- Substitution \( \theta \) is a binding list – set of variable/term pairs – in which each variable is given the ground term replacing it.
- Let \( \alpha(\theta) \) denote the result of applying the substitution \( \theta \) to the sentence \( \alpha \).
- Universal instantiation to eliminate the quantifier is the inference
  \[
  \frac{\forall v: \alpha}{\alpha(v/g)},
  \]
  for any variable \( v \) and ground term \( g \).

- In existential instantiation the variable \( v \) is replaced by a Skolem constant \( k \) a symbol that does not appear elsewhere in the KB
  \[
  \frac{\exists v: \alpha}{\alpha((v/k))},
  \]
  E.g., from the sentence \( \exists x: Father(John) = x \) we can infer the instantiation \( Father(John) = F_1 \), where \( F_1 \) is a new constant.
- Eliminating existential quantifiers by replacing the variables with Skolem constants and universal quantifiers with the set of all possible instantiations turns the KB essentially propositional.
- Ground atomic sentences such as \( Student(John) \) and \( Mother(John, Mary) \) must be viewed as proposition symbols.
- Therefore, we can apply any complete propositional inference algorithm to obtain first-order conclusions.
When the KB includes a function symbol, the set of possible ground term substitutions is infinite

\[ \text{Father(Father(Father(John)))} \]

By the famous theorem due to Herbrand (1930) if a sentence is entailed by the original, first-order KB, then there is a proof involving just a finite subset of the propositionalized knowledge base.

Thus, nested functions can be handled in the order of increasing depth without losing the possibility to prove any entailed sentence.

Inference is hence complete.

Analogy with the halting problem for Turingin machines however shows that the problem is undecidable.

More exactly: the problem is semidecidable, the sketched approach comes up with a proof for entailed sentences.