Part I
Logical agents

Outline

Contents

1 Logical agents 1
1 Knowledge-based agents 1
2 Logic in general models and entailment 1
3 Propositional Logic: a very simple logic 3
4 Equivalence, validity, satisfiability 4
5 Inference rules and theorem proving in propositional logic 5

1 Knowledge-based agents

What is an agent?

Agents interact with environments through sensors and actuators

Knowledge-based agent

Definition 1. A Knowledge base is a set of sentences in a formal language.

Definition 2. Knowledge-based agent: Agents can be viewed at the knowledge level i.e., what they know, regardless of how implemented

Generic knowledge-based agent:
1. PERCEIVE an input
2. **TELL** the knowledge base what it perceives (same language)
3. **ASK** the knowledge base for an action to return (reasoning in the KB for a query, inference)
4. **TELL** the knowledge base about this action that has been executed (update of the KB)

A good way to represent and interact with a KB is *Logic*.

## 2 Logic in general models and entailment

**Logic? But what is it?**

Logics are *formal languages* for representing information such that conclusions can be drawn. To define a logic, we need:

1. **syntax**: how a sentence of the logic looks like?
2. **semantic**: what is the meaning of the sentence?
   - Given a *world*, is the sentence *true* or *false*?

**Example 3.** The language of arithmetic

**Syntax:** $x + 2 \geq y$ is a sentence; $x^2 + y \geq y$ is not a sentence

**Semantic:** $x + 2 \geq y$ 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 sentence ($\alpha$) follows from other sentences (KB) and is denoted:

$$KB \models \alpha$$

We say that the Knowledge Base $KB$ entails $\alpha$ if and only if $\alpha$ is true in all worlds where $KB$ is true. Entailment is a relationship between sentences (i.e. syntax) that is based on semantics.

**Example 4.** Knowledge Base = {“The car is blue” “The bicycle is green or yellow”}

$KB$ entails sentences $\alpha$ like:

- “The car is blue”
- $true$
- “The car is blue or the bicycle is yellow”

The sentence “The car is blue and the bicycle is yellow” is not entailed by $KB$.

**World in Logic = Model**

**Definition 5.** We say $m$ is a model of a sentence $\alpha$ if $\alpha$ is true in the world $m$. We denote by $M(\alpha)$ the set of models

$KB$ entails $\alpha$ if and only if $M(KB) \subseteq M(\alpha)$.

**Example 6.**
Inference

Definition 7. Inference: A sentence $\beta$ can be inferred from another sentence $\alpha$ by some inference algorithm $i$. This is denoted:

$$\alpha \vdash_i \beta$$

Definition 8. Soundness: An inference algorithm is sound if it produces entailed sentences.

Definition 9. Completeness: An inference algorithm is complete if it can derive all the sentences which it entails.

Well-known logics

1. Propositional logic
2. First-order logic
3. Default logic
4. Circumscription
5. Temporal logic
6. Modal logic
7. ..

Every logic has its Pros and Cons (expressivity, soundness and completeness of inference algorithm)

3 Propositional Logic: a very simple logic

Propositional Logic: a very simple logic

We consider a set of proposition symbols $\{p_1, p_2, \ldots\}$.

Definition 10. Syntax: What is a sentence in the propositional logic?

1. any proposition symbol $p_i$ is a sentence (atomic sentence)
2. if $S$ is a sentence then $\neg S$ is a sentence
3. if $S_1$ and $S_2$ are sentences then $S_1 \land S_2$ is a sentence
4. if $S_1$ and $S_2$ are sentences then $S_1 \lor S_2$ is a sentence
5. if $S_1$ and $S_2$ are sentences then $S_1 \Rightarrow S_2$ is a sentence
6. if $S_1$ and $S_2$ are sentences then $S_1 \Leftrightarrow S_2$ is a sentence

Example 11. $p_1, p_1 \land p_2, p_1 \lor (\neg p_2 \land p_3), (p_4 \Rightarrow p_4) \land (p_4 \Rightarrow p_3), \ldots$
Propositional Logic: a very simple logic

**Definition 12. Semantics.** What is the meaning of a sentence? A model \( m \) is a mapping between the proposition symbols \( \{p_1, p_2, \ldots \} \) and \{true, false\}. Given \( m \), we have:

1. \( \neg S \) is true iff \( S \) is false ("not" \( S \))
2. \( S_1 \land S_2 \) is true iff \( S_1 \) is true and \( S_2 \) is true (\( S_1 \) “and” \( S_2 \))
3. \( S_1 \lor S_2 \) is true iff \( S_1 \) is true or \( S_2 \) is true (\( S_1 \) “or” \( S_2 \))
4. \( S_1 \Rightarrow S_2 \) is true iff \( S_1 \) is false or \( S_2 \) is true (\( S_1 \) “implies” \( S_2 \))
   - i.e. \( S_1 \Rightarrow S_2 \) is false iff \( S_1 \) is true or \( S_2 \) is false
5. \( S_1 \Leftrightarrow S_2 \) is true iff \( S_1 \Rightarrow S_2 \) is true and \( S_2 \Rightarrow S_1 \) is true (\( S_1 \) “is equivalent to” \( S_2 \))

Propositional Logic: a very simple logic

**Example 13.** Symbols = \{abcd\}

Given the model \( m = \{a = true, b = false, c = true, d = false\} \), then

- \( \neg a = false \),
- \( a \land \neg b = true \),
- \( a \lor (b \land \neg c) = true \),
- \( d \Rightarrow c = true \),
- \( d \Rightarrow \neg c = true \),
- \( \neg d \Rightarrow \neg c = false \),
- \( \neg (a \lor b) \Leftrightarrow d = true \)

4 Equivalence, validity, satisfiability

**Logical equivalence**

**Definition 14.** Two sentences \( \alpha, \beta \) are logically equivalent IF AND ONLY IF they are true in the same models. \( \alpha \) entails \( \beta \) and vice-versa.

\[
\begin{align*}
(\alpha \land \beta) & \equiv (\beta \land \alpha) \quad \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv (\beta \lor \alpha) \quad \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv (\alpha \land (\beta \land \gamma)) \quad \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv (\alpha \lor (\beta \lor \gamma)) \quad \text{associativity of } \lor \\
\neg(\neg \alpha) & \equiv \alpha \quad \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \beta \Rightarrow \neg \alpha) \quad \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv (\neg \alpha \lor \beta) \quad \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) \quad \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv (\neg \alpha \lor \neg \beta) \quad \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv (\neg \alpha \land \neg \beta) \quad \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv ((\alpha \land \beta) \lor (\alpha \land \gamma)) \quad \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv ((\alpha \lor \beta) \land (\alpha \lor \gamma)) \quad \text{distributivity of } \lor \text{ over } \land 
\end{align*}
\]
Validity and satisfiability

**Definition 15.** A sentence is **valid** if it is true in **ALL models**:

\[ a \lor \neg a, a \Rightarrow a, (a \land (a \Rightarrow b)) \Rightarrow b \]

KB entails \( \alpha \) (\( KB \models \alpha \)) if the sentence \( KB \Rightarrow \alpha \) is valid. Validity is then connected to inference.

**Definition 16.**

1. A sentence is **satisfiable** if it is true in **SOME models**. A valid sentence is satisfiable, but a satisfiable sentence may be not valid.

2. A sentence is **unsatisfiable** if it is true in **NO models**:

\[ a \land \neg a, (a \land b) \Leftrightarrow (\neg a \land c) \]

KB entails \( \alpha \) (\( KB \models \alpha \)) if the sentence \( KB \land \neg \alpha \) is unsatisfiable.

5 Inference rules and theorem proving in propositional logic

**Inference, Theorem proving**

Given \( KB \), can I prove the sentence \( \alpha \)? Is \( \alpha \) satisfiable in \( KB \)? \( KB \models \alpha \)? Is \( KB \land \neg \alpha \) unsatisfiable?

Model (Truth table) enumeration, we check that \( M(KB) \subseteq M(\alpha) \).

Bad news: exponential in the number of proposition symbols involved in \( KB, \alpha \).

Some improved methods: Davis-Putnam-Logemann-Loveland (complete), min-conflicts-like (incomplete) hill-climbing (incomplete).

- Sound generation of new sentences from old.
- Proof = a sequence of inference rules that finally generate \( \alpha \)

Methods: Forward-chaining, Backward chaining, Resolution

**Inference rules: examples**

**Example 17.** Modus Ponens:

\[ \begin{array}{c}
  a, a \Rightarrow b \\
  \hline
  b
\end{array} \]

And-elimination:

\[ \begin{array}{c}
  a \land b \\
  \hline
  a
\end{array} \]

Factoring:

\[ \begin{array}{c}
  a \lor a \\
  \hline
  a
\end{array} \]

Logical equivalences:

\[ \begin{array}{c}
  \neg a \lor \neg b \\
  \hline
  \neg (a \land b)
\end{array} \]

\[ \begin{array}{c}
  a \Leftrightarrow b \\
  \hline
  a \Rightarrow b \land b \Rightarrow a
\end{array} \]
Forward and backward chaining methods
Proof methods that are simple and efficient. They need a restriction on the form of $KB$.

$$KB = \text{conjunction of Horn clauses}$$

**Definition 18.** A Horn clause is

- a propositional symbol $a$, or
- something like $p_1 \cdots p_n \Rightarrow c$ ($p_i$ is a premise and $c$ the conclusion)

**Inference problem**

**Rule 1** $P \Rightarrow Q$

**Rule 2** $L \land M \Rightarrow P$

**Rule 3** $B \land L \Rightarrow M$

**Rule 4** $A \land P \Rightarrow L$

**Rule 5** $A \land B \Rightarrow L$

**Rule 6** $A$

**Rule 7** $B$

Is the proposition $Q$ true or not?
**Forward chaining method**

*Forward chaining:*

1. Fire any rule whose premises are satisfied in the Knowledge Base
2. Add its conclusion to the Knowledge Base
   - Management of a *Working Memory* (an Agenda)
3. Repeat 1 and 2 until the proposition $Q$ is true or no more conclusion can be derived

\[
\begin{align*}
\frac{a_1, \ldots, a_n}{b} & \quad a_1 \land \cdots \land a_n \Rightarrow b
\end{align*}
\]

**Forward chaining: example**

$A$ and $B$ are known in the knowledge base (Rules 6 and 7).

\[
Agenda = \{A, B\}
\]

**Forward chaining: example**

Rule 5 can be fired since $A$ and $B$ are known.

\[
A \land B \Rightarrow L
\]
Forward chaining: example

The agenda is updated with the conclusion (head) of Rule 5:

\[ \text{Agenda} = \{A, B, L\} \]

Forward chaining: example

Rule 3 can be fired since \( B \) and \( L \) are known.

\[ B \land L \Rightarrow M \]
Forward chaining: example

The agenda is updated with the conclusion (head) of Rule 3:

\[ \text{Agenda} = \{ A, B, L, M \} \]

Forward chaining: example

Rule 2 can be fired since \( M \) and \( L \) are known.

\[ M \land L \Rightarrow P \]
Forward chaining: example
The agenda is updated with the conclusion (head) of Rule 2:

\[ \text{Agenda} = \{A, B, L, M, P\} \]

Forward chaining: example
Rule 1 can be fired since \( P \) is known.

\[ P \Rightarrow Q \]
Forward chaining: example

The agenda is updated with the conclusion (head) of Rule 1:

$$\text{Agenda} = \{A, B, L, M, P, Q\}$$

$Q$ has been derived. STOP

Properties of the FC algorithm

**Theorem 19.** The forward chaining algorithm is complete.

- Every atomic proposition that is provable in the knowledge base can be derived thanks to the algorithm.
- The result is due to the fact that:
  1. The knowledge base is a conjunction of Horn clauses.
  2. In the worst case of FC, every clause is fired.
Theorem 20. The forward chaining algorithm runs in linear time.

- Every rule is fired at most once.

Backward chaining method

At a given step, the choice between the firable rules is random. FC may apply rules that are useless for the proof of $Q$. Data-driven algorithm

Backward chaining: Goal-driven algorithm

- To prove $Q$ every premise of one rule which concludes $Q$ has to be at least proved.
- Basically, BackwardChaining($Q$) =
  1. If $Q$ is known in $KB$ then $Q$ is proved
  2. Otherwise, select a rule $P_1 \land \cdots \land P_n \Rightarrow Q$ in KB
  3. Recursively apply BackwardChaining($P_1$), ... , BackwardChaining($P_n$) to prove the premises.
  4. Repeat 2-3 until $Q$ is proved or no more rules can be selected.

Backward chaining: example

To prove $Q$, we need to prove $P$. To prove $P$ we need to prove $L$ and $M$ etc. So it goes until $A$ and $B$ are reached. $A$ and $B$ are known in KB so $Q$ is proved.

Backward chaining method (2)

Theorem 21. The backward chaining algorithm is complete.

Theorem 22. The backward chaining algorithm runs in linear time.

- Every rule is fired at most once
- In practice, it is much less than linear in size of KB: it is linear in the size of the set of rules that are involved in the proof of the premises of $Q$. 

12
**Propositional logic inference: resolution algorithm**

FC and BC are efficient algorithms but they make the assumption that KB is a set of Horn clauses.

Given a KB, is there an algorithm which is sound and complete?

**YES**, this is called a *resolution algorithm* based on the resolution inference rule.

**Resolution inference rule**

\[
\ell_1 \vee \cdots \vee \ell_i \vee \cdots \vee \ell_k, \quad \ell'_1 \vee \cdots \vee \ell'_j \vee \cdots \vee \ell'_n
\]

\[
\ell_1 \vee \cdots \vee \ell_{i-1} \vee \ell_{i+1} \vee \cdots \vee \ell_k \vee \ell'_1 \vee \cdots \vee \ell'_{j-1} \vee \ell'_{j+1} \vee \cdots \vee \ell'_{n}
\]

with \( \ell_i = a \) and \( \ell_j = \neg a \) or \( \ell_i = \neg a \) and \( \ell_j = a \)

(\( a \) is an atomic sentence)

**Example 23.**

\[
\begin{align*}
\ell_1 & = e, \quad \ell_2 = \neg e \\
v \lor \neg w \lor \neg x \lor w & \lor \neg x \lor y \lor z
\end{align*}
\]

**Resolution inference rule (2): CNF**

The rule is applied on sentences like \((\ell_1 \lor \cdots \lor \ell_k) \land \cdots \land (\ell_m \lor \cdots \lor \ell_n)\) where \( \ell_i \) are positive/negative literals.

This form is called *Conjunctive Normal Form* (CNF for short).

Every sentence in proposition logic is logically equivalent to a CNF sentence.

**Example 24.** \((a \lor b) \iff (c \land d)\) is logically equivalent to the CNF \((-a \lor c) \land (-a \lor d) \land (-b \lor c) \land (-b \lor d) \land (-c \lor -d \lor a \lor b).\)

**Conversion to a CNF**

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 \(\neg(a \land b) \equiv (\neg a \lor \neg b), \neg(a \lor b) \equiv (\neg a \land \neg b)\) and double negation rule \(\neg\neg a \equiv a\)
4. Apply distributivity law (\( \lor \) over \( \land \)) and flatten \((a \land b) \lor c \equiv (a \lor c) \land (b \lor c)\)
Resolution algorithm

Definition 25. **Proof by contradiction:** given $KB$, to prove $\alpha$, we prove that $KB \land \neg \alpha$ is not satisfiable.

**Example 26.** Symbols:

- $Und$: “The students have understood this lecture”
- $Gt$: “I am a good teacher”
- $Party$: “The students went to a party last night”

Knowledge base:

$$KB = (\neg Und \iff (\neg Gt \lor Party)) \land Und$$

Query to prove: *I am a good teacher*

$$\alpha = Gt$$

Resolution algorithm

**Example 27.** Conversion to CNF: $KB \land \neg \alpha \rightarrow (KB \land \neg \alpha)_{CNF}$

$$(KB \land \neg \alpha)_{CNF} = (\neg Party \lor \neg Und) \land (\neg Gt \lor Und \lor Party) \land (\neg Und \lor Gt) \land Und \land \neg Gt$$

**Example 28.**

```
¬ Party ∨ ¬ Und  ¬ Gt ∨ Und ∨ Party  ¬ Und ∨ Gt  Und  ¬ Gt
```

Resolution algorithm

**Example 29.**

```
¬ Party ∨ ¬ Und  ¬ Gt ∨ Und ∨ Party  ¬ Und ∨ Gt  Und  ¬ Gt
```

Resolution algorithm

**Example 30.**

```
¬ Und ∨ ¬ Gt ∨ Und
```
Resolution algorithm

Example 31.

Resolution algorithm

Example 32.

Resolution algorithm

Example 33.
Resolution algorithm

Example 35.

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

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

Propositional logic lacks expressive power