Artificial Intelligence

An introduction

Maik Kschischo

Institute for Computer Science
University of Koblenz

Table of Contents

Literature

Recommendations

1. Russel, P. and Norvig, S,.Artificial Intelligence: A Modern Approach., 4th ed., Pearson. 2021.
2. Aggarwal, C.C. Artificial Intelligence: A textbook., Springer Nature Switzerland AG 2021. `https://doi.org/10.1007/978-3-030-72357-6`
3. Ertel, W. Introduction to Artificial Intelligence., 2nd ed., Springer Nature Switzerland AG 2017. `https://doi.org/10.1007/978-3-319-58487-4`

Assignments

- Oral or written exam, depending in the number of students

Recommendations

- Follow the lectures!
- Read!!
- Try to solve the problems!!!

Chap. 1 Overview and history of AI

Chap. 2 Propositional logic

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through “rules”

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet
- Another approach is "subsymbolic AI" or "inductive AI"

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet
- Another approach is "subsymbolic AI" or "inductive AI"
  - Motivation is mainly about simulation/imitation

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet
- Another approach is "subsymbolic AI" or "inductive AI"
  - Motivation is mainly about simulation/imitation
  - Example: neural networks

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet
- Another approach is "subsymbolic AI" or "inductive AI"
  - Motivation is mainly about simulation/imitation
  - Example: neural networks
  - Approaches are usually black boxes

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through “rules”
  - The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
  - Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
  - Motivation is mainly about simulation/imitation
  - Example: neural networks
  - Approaches are usually black boxes
  - Very successful, but explainability is a problem

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet
- Another approach is "subsymbolic AI" or "inductive AI"
  - Motivation is mainly about simulation/imitation
  - Example: neural networks
  - Approaches are usually black boxes
  - Very successful, but explainability is a problem
- For subsymbolic AI see the course "Machine learning" and "Deep Learning"

Symbolic AI vs. subsymbolic AI

- the area of “symbolic AI” is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through “rules”
  - The motivation behind this is not only “simulation” of intelligent processes but also “explanation” (white box approach)
  - Not yet
- Another approach is “subsymbolic AI” or “inductive AI”
  - Motivation is mainly about simulation/imitation
  - Example: neural networks
  - Approaches are usually black boxes
  - Very successful, but explainability is a problem
- For subsymbolic AI see the course “Machine learning” and “Deep Learning”

Symbolic AI vs. subsymbolic AI

- the area of "symbolic AI" is about formalisation of thought/reasoning through symbolic manipulation
  - Relationships are modelled through "rules"
  - The motivation behind this is not only "simulation" of intelligent processes but also "explanation" (white box approach)
  - Not yet
- Another approach is "subsymbolic AI" or "inductive AI"
  - Motivation is mainly about simulation/imitation
  - Example: neural networks
  - Approaches are usually black boxes
  - Very successful, but explainability is a problem
- For subsymbolic AI see the course "Machine learning" and "Deep Learning"

$\rightarrow$ the foundation for symbolic AI is *logic*

Logic and formal systems

- Logics can be used to model reasoning processes

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

  Anna is a student
  All students are humans
  $\rightarrow$ Anna is a human

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

  Anna is a student
  All students are humans
  $\rightarrow$ Anna is a human
- Analysis:

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

  Anna is a student
  All students are humans
  $\rightarrow$ Anna is a human
- Analysis:
  - Given that the statement "Anna is a student" is true

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

  Anna is a student
  All students are humans
  $\rightarrow$ Anna is a human
- Analysis:
  - Given that the statement “Anna is a student” is true
  - Given that the statement “All students are humans” is true

Logic and formal systems

- Logics can be used to model reasoning processes
- A (logical) statement is an abstract construct that is either T or F
- Formal logic is about making statement about statements
- Example:

  Anna is a student
  All students are humans
  $\rightarrow$ Anna is a human
- Analysis:
  - Given that the statement "Anna is a student" is true
  - Given that the statement "All students are humans" is true
  - then the statement "Anna is a human" is a necessarily true statement

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? $(\Sigma = \{\text{Anna}, \text{human}, \text{student}\})$

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? $(\Sigma = \{\text{Anna}, \text{human}, \text{student}\})$
   2. Grammar: how can symbols be combined in order to obtain complex statements? $(\text{student} \Rightarrow \text{human})$

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? $(\Sigma = \{\text{Anna}, \text{human}, \text{student}\})$
   2. Grammar: how can symbols be combined in order to obtain complex statements? $(\text{student} \Rightarrow \text{human})$
2. Semantics: Which are the "true" statements? What is the relationship between true statements?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? ($\Sigma = \{\text{Anna}, \text{human}, \text{student}\}$)
   2. Grammar: how can symbols be combined in order to obtain complex statements? ($\text{student} \Rightarrow \text{human}$)
2. Semantics: Which are the “true” statements? What is the relationship between true statements?
   1. Interpretations (or “possible worlds”): assign meaning to symbols of language.

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? ($\Sigma = \{\text{Anna}, \text{human}, \text{student}\}$)
   2. Grammar: how can symbols be combined in order to obtain complex statements? ($\text{student} \Rightarrow \text{human}$)
2. Semantics: Which are the "true" statements? What is the relationship between true statements?
   1. Interpretations (or "possible worlds"): assign meaning to symbols of language.
   2. Models: What are the interpretations in which a statement is true?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? $(\Sigma = \{\text{Anna}, \text{human}, \text{student}\})$
   2. Grammar: how can symbols be combined in order to obtain complex statements? $(\text{student} \Rightarrow \text{human})$
2. Semantics: Which are the "true" statements? What is the relationship between true statements?
   1. Interpretations (or "possible worlds"): assign meaning to symbols of language.
   2. Models: What are the interpretations in which a statement is true?
   3. Entailment: when is one statement entailed (follows logically from) another?

Logic and formal systems: structure

Every logic (=formal system) has the following components:

1. Syntax: What are the possible statements?
   1. Signature: What symbols are allowed? ($\Sigma = \{\text{Anna}, \text{human}, \text{student}\}$)
   2. Grammar: how can symbols be combined in order to obtain complex statements? ($\text{student} \Rightarrow \text{human}$)
2. Semantics: Which are the "true" statements? What is the relationship between true statements?
   1. Interpretations (or "possible worlds"): assign meaning to symbols of language.
   2. Models: What are the interpretations in which a statement is true?
   3. Entailment: when is one statement entailed (follows logically from) another?
3. Calculus: How can entailment be implemented?

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**
- Propositions can be connected to form new propositions using a **formula**

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**
- Propositions can be connected to form new propositions using a **formula**

Propositions and logical expressions

- **Atoms** are individual propositions which can not be further broken down into smaller constituents
- The propositions are represented by **symbols**
- Propositions can be connected to form new propositions using a **formula**

Example

$a =$ "The street ist wet."
$b =$ "It is raining."
We can connect these two propositions to form a new proposition
$c =$ "If it is raining, the street is wet."
The proposition c can be expressed as a formula
$b \Rightarrow a$

We will see, that logic processing is independent of the semantics of the propositional symbols.

Syntax of propositional logic I

Definition

Let Σ be a set of symbols and $\mathcal{O} = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ be the set of logical operators. The sets $\mathcal{O}, \Sigma$ and $\{T, F\}$ are pairwise disjoint. Σ is called the **signature** and its elements are the **proposition variables**. The set of propositional logic formulas is recursively defined via:

- T and F are (atomic) formulas.
- All proposition variables, i.e. all elements of Σ are (atomic) formulas.
- If a and b are formulas, then $\neg a$, (a), $a \vee b$, $a \wedge b$, $a \Rightarrow b$, $a \Leftrightarrow b$ are also formulas.

Example

Given $\Sigma = \{a, b, c\}$ we can form the formulas
$a \wedge b$, , $a \wedge b \wedge c$, $a \vee c \wedge b$, $(\neg a \wedge b) \Rightarrow (\neg c \vee a)$

Syntax of propositional logic II

Definition

We read the symbols in the following way:

$$
\begin{aligned}
T &: \text{"true"} \\
F &: \text{"false"} \\
\neg a &: \text{"not a"} && \text{(negation)} \\
a \land b &: \text{"a and b"} && \text{(conjunction)} \\
a \lor b &: \text{"a or b"} && \text{(disjunction)} \\
a \Rightarrow b &: \text{"if a then b"} && \text{(implication))} \\
a \Leftrightarrow b &: \text{"a if and only if b"} && \text{(equivalence))}
\end{aligned}
$$

Semantics of logical formulas

There are only two truth values in propositional logic, T for "true" and F for "false".

Example

When is the formula $a \wedge b$ true? It depends, whether a and b are true. If $a =$ "It is cold today." and $b =$ "It is raining." and both are true, then $a \wedge b$ is true.
If, however, $b =$ "It is dry.", but b is false, then $a \wedge b$ is false.

This means, that we have to assign T or F to the proposition variables, reflecting the state of the world.

Why we need interpretations I

Logic formulals alone can not represent any knowledge. We need to assign objects of the real world to the logic symbols to decide about the truth of a logic formula.

Example

A safety system with a 2 of 3 logic.

- Let Sig_1, Sig_2, and Sig_3 be three sensors measuring the same signal.
- If at least two sensors have values above a critical value, the system should indicate an error.

We introduce symbols describing the safety system

- $a =$ "Signal Sig_1 is above a critical value"
- $b =$ "Signal Sig_2 is above a critical value"
- $c =$ "Signal Sig_3 is above a critical value"
- $z =$ "The safety system indicates an error."

Why we need interpretations II

Example

Now, consider the following formulas

- $a \wedge c \Rightarrow z$: If Sig_1 is above critical ($a = T$) and Sig_3 is above critical ($c = T$) implies that system indicates an error ($z = T$),
- $\neg a \vee \neg b \Rightarrow z$: This states, that if Sig_1 is not above critical ($\neg a = T$) and Sig_3 is not above critical ($\neg c = T$), then the system should indicate an error ($z = T$).

For our problem, the formula $a \wedge c \Rightarrow z$ is true, but $\neg a \vee \neg b \Rightarrow z$ is false.

This example illustrates

1. We need interpretations to assign a truth value to logical formulas
2. A given set of formulas can be be valid in different domains (worlds). The logical formulas as are the same, but they consider different facts of the world.

Semantics of logical formulas

Interpretations

Definition

A mapping $I : \Sigma \rightarrow \{T, F\}$ assigning to very proposition variable $s \in \Sigma$ a truth value $I(s)$ is called an **interpretation**.

If there are n variables in a formula, there are 2^n different interpretations.

Semantics of logical formulas

Truth table for logical operators

a	b	(a)	$\neg a$	$a \wedge b$	$a \vee b$	$a \Rightarrow b$	$a \Leftrightarrow b$
T	T	T	F	T	T	T	T
T	F	T	F	F	T	F	F
F	T	F	T	F	T	T	F
F	F	F	T	F	F	T	T

The empty formular is true for all interpretations.

Semantics of logical formulas

Priority of logical operators

Order of evaluating more complex formulas

1. Expression within parentheses $(\cdot)$ are evaluated first.
2. Order of priorities for unparenthesized formulas (from left to right):

$$\neg, \quad , \wedge, \quad , \vee, \quad , \Rightarrow, \quad , \Leftrightarrow$$

Semantics of logical formulas

Priority of logical operators

Order of evaluating more complex formulas

1. Expression within parentheses $(\cdot)$ are evaluated first.
2. Order of priorities for unparenthesized formulas (from left to right):

$$\neg, \quad ,\wedge, \quad ,\vee, \quad ,\Rightarrow, \quad ,\Leftrightarrow$$

Example

1. $(\neg(a \wedge b)) \Rightarrow (c \wedge d)$ means the same as $\neg(a \wedge b) \Rightarrow (c \wedge d)$ or $\neg(a \wedge b) \Rightarrow c \wedge d$

Semantics of logical formulas

Priority of logical operators

Order of evaluating more complex formulas

1. Expression within parentheses $(\cdot)$ are evaluated first.
2. Order of priorities for unparenthesized formulas (from left to right):

$$\neg, \quad ,\wedge, \quad ,\vee, \quad ,\Rightarrow, \quad ,\Leftrightarrow$$

Example

1. $(\neg(a \wedge b)) \Rightarrow (c \wedge d)$ means the same as $\neg(a \wedge b) \Rightarrow (c \wedge d)$ or $\neg(a \wedge b) \Rightarrow c \wedge d$
2. $(((a \wedge b) \vee c) \wedge (\neg d)) \Rightarrow e$ means the same as $(a \wedge b \vee c) \wedge \neg d \Rightarrow e$

Models of formulas

Definition

An interpretation $I : \Sigma \to \{T, F\}$ that satisfies a formula s is called a **model** (world) of the formula s.

We will denote all models of a given formula s by $\mathcal{M}(s)$.

Example

Let $\Sigma = \{a, b\}$ be the signature (set of symbols) and $s = a \vee b$.

- One model of $s = a \vee b$ is $\{I(a) = T, I(b) = T\}$
- All models are given by the set $\mathcal{M}(a \vee b) = \{(I(a) = T, I(b) = T), (I(a) = T, I(b) = F), (I(a) = F, I(b) = T)\}$

Determining models of formuals using truth tables

For a given set of symbols Σ we can obtain the set $\mathcal{M}(s)$ by the following algorithm:

1. Determine all interpretations $I \to \{t, f\}$ and assign a truth table.
2. For all interpretations determine the truth value of s.
3. Delete all rows from the truth table with $I(s) = F$. Each remaining row of the truth table describes one model of s.

Example

$s = \neg a \vee b$

a	b	$\neg a \vee b$
T	T	T
T	F	F
F	T	T
F	F	T

$\mathcal{M}(\neg a \vee b) =$
$\{(I(a) = T, I(b) = T), (I(a) = F, I(b) = T), (I(a) = F, I(b) = F)\}$

Classification of logic formulas

Definition

A formula s is called

- **satisfiable**, if it is true for at least interpretation, i.e. $\mathcal{M}(s) \neq \emptyset$.
- **falsifiable**, if it is false for at least one interpretation, i.e. $\mathcal{M}(\neg s) \neq \emptyset$.
- **logically valid (true)**, if it is true for all interpretations, i.e. $\mathcal{M}(\neg s) = \emptyset$. True formulas are also called **tautologies**.
- **unsatisfiable** if is not true for any interpretation.

Iff s is a tautology, then $\neg s$ is unsatisfiable.
The formula s is satisfiable, iff $\neg s$ is falsifiable.

Semantic equivalence I

versus syntactic equivalence

Definition

Two formulas a and b are called **semantically equivalent** if they take on the same value for all their interpretations. We write $a \equiv b$.

Syntactic equivalence $a \Leftrightarrow b$ is just a syntactic object of the formal language of propositional logic, defined via the truth table. Semantic equivalence $a \equiv b$ is used to describe the same meaning of the propositions a and b.

Semantic equivalence II

versus syntactic equivalence

Example

The equivalence

$$a \Rightarrow b \equiv b \vee \neg a$$

can be verified from the truth table

a	b	$a \Rightarrow b$	$b \vee \neg a$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

Semantic equivalence III

versus syntactic equivalence

Example

The equivalence

$$a \Leftrightarrow b \equiv (b \vee \neg a) \wedge (a \vee \neg b)$$

can also be verified from the truth table (reading exercise).

Equivalent logical expressions

Theorem

The junctors $\wedge$ and $\vee$ are commutative, associative and idempotent.

This means for example

$$a \wedge b \equiv b \wedge a \qquad \text{and} \qquad a \vee b \equiv b \vee a \qquad \text{(commutativity)}$$

$$a \vee (b \vee c) \equiv (a \vee b) \vee c \qquad \text{(associativity)}$$

$$a \wedge a \equiv a \qquad \text{(idempotency)}$$

Theorem (Equivalences)

$$
\begin{array}{rcll}
\neg a \vee b & \Leftrightarrow & a \Rightarrow b & \textit{(implication)} \\
a \Rightarrow b & \Leftrightarrow & \neg b \Rightarrow \neg a & \textit{(contraposition)} \\
(a \Rightarrow b) \wedge (b \Rightarrow a) & \Leftrightarrow & (a \Leftrightarrow b) & \textit{(equivalence)} \\
\neg(a \wedge b) & \Leftrightarrow & \neg a \vee \neg b & \textit{(De Morgan´s law)} \\
\neg(a \vee b) & \Leftrightarrow & \neg a \wedge \neg b & \textit{(De Morgan´s law)} \\
a \vee (b \wedge c) & \Leftrightarrow & (a \vee b) \wedge (a \vee c) & \textit{(distributive law)} \\
a \wedge (b \vee c) & \Leftrightarrow & (a \wedge b) \vee (a \wedge c) & \textit{(distributive law)} \\
a \vee \neg a & \Leftrightarrow & t & \textit{(tautology)} \\
a \wedge \neg a & \Leftrightarrow & f & \textit{(contradiction)} \\
a \vee f & \Leftrightarrow & a & \\
a \vee t & \Leftrightarrow & t & \\
a \wedge f & \Leftrightarrow & f & \\
a \wedge t & \Leftrightarrow & a &
\end{array}
$$

Clauses and conjunctive normal form I

Definition (Clauses and conjunctive normal form (CNF))

1. A **literal** L is a variable or a negated variable.
2. A clause K consists of a **disjunction**

$$L_1 \lor L_2 \lor \ldots \lor L_m$$

   of literals.
3. A formula is in **conjunctive normal form** if and only if it consists of a **conjunction**

$$K_1 \land K_2 \land \ldots \land K_n$$

   of clauses K_i.

Clauses and conjunctive normal form II

Example

The formulas

$$a \vee b \tag{1}$$

$$p \vee q \vee \neg r \tag{2}$$

are clauses. Here, the symbol r and its negation $\neg r$ is one literal. Combining these by conjunctions provides a formula

$$(a \vee b) \wedge (p \vee q \vee \neg r)$$

in CNF.

Transformation into an equivalent CNF I

Theorem

Every propositional logic formula can be transformed into an equivalent conjunctive normal form.

Algorithm to transform a formula into a CNF

1. Use the equivalences

$$\begin{aligned} \neg a \vee b &\Leftrightarrow a \Rightarrow b && \text{(implication)} \\ (a \Rightarrow b) \wedge (b \Rightarrow a) &\Leftrightarrow (a \Leftrightarrow b) && \text{(equivalence)} \end{aligned}$$

   to remove implications ($\Rightarrow$) and equivalences ($\Leftrightarrow$).

Transformation into an equivalent CNF II

2. Use the equivalences

$$
\begin{aligned}
\neg(a \wedge b) \quad &\Leftrightarrow \quad \neg a \vee \neg b \qquad \text{(De Morgan´s law)} \\
\neg(a \vee b) \quad &\Leftrightarrow \quad \neg a \wedge \neg b \qquad \text{(De Morgan´s law)} \\
\neg(\neg a) = a &
\end{aligned}
$$

   to move the negation symbolc ($\neg$) directly in front of each symbol.

3. Use the distributive laws

$$
\begin{aligned}
a \vee (b \wedge c) \quad &\Leftrightarrow \quad (a \vee b) \wedge (a \vee c) \qquad \text{(distributive law)} \\
a \wedge (b \vee c) \quad &\Leftrightarrow \quad (a \wedge b) \vee (a \wedge c) \qquad \text{(distributive law)}
\end{aligned}
$$

   to generate the CNF.

Transformation into an equivalent CNF III

Example

$$
\begin{aligned}
(p \vee \neg r) \Rightarrow q &\overset{(1)}{=} q \vee \neg(p \vee \neg r) \\
&\overset{(2)}{=} q \vee (\neg p \wedge r) \\
&\overset{(3)}{=} (q \vee \neg p) \wedge (q \vee r)
\end{aligned}
$$

Entailment

Assume we are given a knowledge base s as propositional logical formula. From this we want to deduce whether a query q is true or false.

Definition

A formula s entails a formula q (or q follows from s), if every model of s is also a model of q

$$\mathcal{M}(s) \subseteq \mathcal{M}(q)$$

Then we write $s \models q$

- This means that the truth of q is contained in the truth of s, but not necessarily vice versa. This means, that q can be true, even when $I(s) = F$.
- For a tautology y we have $I(y) = T$ for all interpretations I, i.e. $\emptyset \models y$. We write $\models y$.

Entailment II

An example

Example

The formula

$$s = a \wedge b \wedge (b \wedge c \vee a \wedge b \vee a \wedge c \Rightarrow z)$$

has two models, which are shown in the shortened truth table

a	b	c	z	s
T	T	T	T	T
T	T	F	T	T

Each formula, which is true for the two interpretations of a, b, c and z, follows from s. Examples are: z, $b \wedge z$, $a \vee b \vee c$, $c \wedge b \vee \neg c$.

The deduction theorem I

Theorem

$a \models b$ if and only if $\models a \Rightarrow b$

If a entails b (b follows from a), $a \Rightarrow b$ is a tautology.

The deduction theorem II

Proof.

Remember the truth table for the implication

a	b	$a \Rightarrow b$
T	T	T
T	F	F
F	T	T
F	F	T

- Assume, that $a \models b$ holds. This means that $\mathcal{M}(a) \subseteq \mathcal{M}(b)$, This means that the second line with $a \Rightarrow b = F$ can not be realized and thus $a \Rightarrow b = T$.
- Assume $a \Rightarrow b = T$. Then, the second line as again excluded and every model of a is again a model of b.

$\square$

A first proof system

From the deduction theorem we learn:

- To show that $a \models b$ we can show that $a \Rightarrow b$ is a tautology, i.e, always true.
- This can be done automatically by checking the truth table.
- Drawback: For large formulas a and b, it can take a long time, because for a formula with a wth n proposition variables there are 2^n different interpretations.
- We will use derivation to overcome this limitation.

Proof by contradiction

- The deduction theorem tells us that $a \models b$ can be proofed by showing $a \Rightarrow b$
- Therefore, the negation $\neg(a \Rightarrow b)$ is unsatisfiable
- With
$$\neg(a \Rightarrow b) \equiv \neg(\neg a \lor b) \equiv a \land \neg b$$
we see that $a \land \neg b$ must be unsatisfiable.

Theorem (Proof by contradiction)

$a \models b$ if and only if $a \land \neg b$ is unsatisfiable.

If we have a query b and ask, whether b follows from a knowledge base a, we can add $\neg b$ to the knowledge base and derive a contradiction, because $b \land \neg b$ is unsatisfiable.

Derivation

- Idea: To show entailment $a \models b$ we replace the test of all interpretations using a truth table by a **syntactic manipulation**.
- This syntactic manipulation is called a **derivation** and we write $a \vdash b$.
- The rules of derivation form a **propositional calculus**.

Entailment: b follows from a ($a \models b$) is a **semantic** concept. It says, that for all models b are also models of a.

Derivation: is a syntactic approach to apply rules to derive b from a, i.e. to manipulate a syntactiacally to show $a \vdash b$

Soundness and completeness of syntactic calculi

When does a calculus produce correct answers?

Definition

A calculus is called **sound** or **correct**, if a derived proposition follows semantically. That is for formulas a and b we have

$$(a \vdash b) \Rightarrow (a \models b).$$

A calculus is called **complete** if all semantic consequences can be derived. That is for formulas a and b we have

$$(a \models b) \Rightarrow (a \vdash b).$$

Some notation I

for derivation rules

- We write

$$\{f_1, \ldots, f_n\} \qquad \text{or} \qquad \begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_n \end{array}$$

for the formula $f \equiv f_1 \wedge f_2, \ldots, \wedge f_n$ with conjunctions between the propositions.

- All propositions which are listed in these expression are set to true T.
- $\{f_1, \ldots, f_n\} \models s$ if and only if for $I(f_1) = T \ldots, I(f_n) = T$ we also have $I(s) = T$.

Some notation II

for derivation rules

Using truth tables can be computationally expensive. Instead, we use a syntactic calculus. We write for a **derivation**

$$\{f_1, \ldots, f_n\} \vdash s \qquad \text{or} \qquad \begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_n \\ \hline s \end{array}$$

We call $f_1, \ldots, f_n$ the **premises** and s the **conclusion**.

Modus Ponens I

A simple rule of derivation

Theorem (Modus Ponens)

$$\frac{\begin{array}{l} p \Rightarrow q \\ p \end{array}}{q} \qquad \text{or} \qquad \{p \Rightarrow q,\, p\} \vdash q \qquad \text{or} \qquad \frac{\begin{array}{l} I(p \Rightarrow q) \equiv T \\ I(p) \equiv T \end{array}}{I(q) \equiv T}$$

"If q follows from p AND p is true, then q is also true."

The three notations are equivalent.

Modus Ponens II

A simple rule of derivation

Proof.

$I(p)$	$I(q)$	$I(p \wedge (p \Rightarrow q))$
T	T	T
T	F	F
F	T	F
F	F	F

The premises q and $p \Rightarrow q$ are true by assumption. Then, there is only one model and in this model q is also true. $\square$

Task

Given:
1. Set of true propositions (Axioms, knowledge base)
2. A proposition which might be true or not (query)

Wanted: A proof that the query is true

Definition (Decidability)

A propositional calculus is called **decidable** if there is an algorithm which provides a decision whether any query against any set of axioms is true or false in a finite number of steps.

Remark: Finite can still mean that we need many steps.

The resolution rule

The modus ponens rule can be generalized. Assume we are given three clauses and let A, B and C be their literals. Then

$$\frac{\begin{array}{c} A \vee B \\ \neg A \vee C \end{array}}{B \vee C}$$

The derived clause is called the **resolvent**.

Proof.

Using truth tables. □

The general resolution rule I

Theorem

Given the two clauses $A \vee B_1 \vee \ldots \vee B_n$ and $\neg A \vee C_1 \vee \ldots \vee C_m$ with the complementary literals A and $\neg A$ are both true we can derive $B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m$.

$$\frac{\begin{array}{l} A \vee B_1 \vee \ldots \vee B_n \\ \neg A \vee C_1 \vee \ldots \vee C_m \end{array}}{B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m}$$

The general resolution rule II

Example

Assume $a = T$, $b = T$ and $c \vee \neg a \vee \neg b = T$. We want to show that C is true. We look for complementary literals and apply the resolution rule

$$\frac{\begin{array}{l} a \\ c \vee \neg a \vee \neg b \end{array}}{c \vee \neg b}$$

Now we apply the resolution rule to the resolvent

$$\frac{\begin{array}{l} b \\ c \vee \neg b \end{array}}{c}$$

Proof by contradiction I

Typically, the resolution rule is used in a different way

- Add the negated query to the axioms
- Generate the empty clause $\emptyset$ which corresponds to a contradiction. This means, in the last step we use

$$\frac{\begin{array}{l} A \\ \neg A \end{array}}{\emptyset}$$

Proof by contradiction II

Example

Assume $a = T$, $b = T$ and $c \vee \neg a \vee \neg b = T$. We want to show that C is true. To proof c we add $\neg c$ to the axioms.

$$\begin{array}{l} c \vee \neg a \vee \neg b \\ \neg c \\ \hline \neg a \vee \neg b \end{array}$$

Now we apply the resolution to the resolvent and b

$$\begin{array}{l} \neg a \vee \neg b \\ b \\ \hline \neg a \end{array}$$

Proof by contradiction III

Example

Now we combine this with a from the axioms to obtain the empty clause

$$\begin{array}{c} \neg a \\ \underline{a \quad} \\ \emptyset \end{array}$$

Resolutional calculus

Resolution refutation

Given: Axioms (knowledge base) and the proposition (query) to be proofed.

1. Formulate the axioms and the query as clauses and put these to the memory.
2. Chose two clauses with complementary literals. If there are no such clauses, then stop (Proof is impossible).
3. Apply the resolution rule to the selected clauses with the complementray literals.
4. If the resolvent is the empty clause $\emptyset$, then Stop (Proof successfully finished). Otherwise, add the resolvent to the memory and go to step 2.

Result: Proof of the query (if it is possible)

Properties of resolutional calculus

Theorem

The resolution calculus for the proof of unsatisfiability of formulas in conjunctive normal form is sound and complete.

1. The resolutional calculus is sound, because the resolution rule is correct and the proof by contradiction is correct, This means, that the query q follows from the axioms a via unstaisfiability of $a \land \neg q$, which is equivalent to $a \models q$.
2. The resolutional calculus is not per se complete. A counterexample is the proof of $q \lor \neg q$, which is not possible by resolution. However, we can proof this by contradiction and use the negation $\neg(q \lor \neg q) \equiv \neg q \land q$ and now we can use resolution to derive the empty clause $\emptyset$. In this sense the resolutional calculus is complete, if the proof by contradiction is included.

Example I

Control of traffic lights

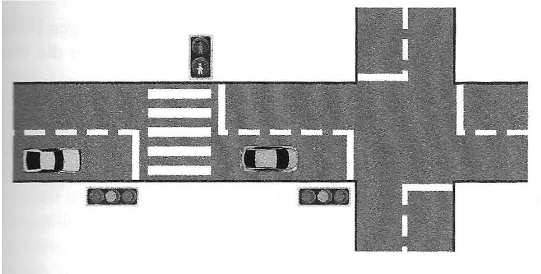

Figure: Two traffic lights and an on demand pedestrian crossing. From Lunze, KI für Ingenieure, Fig. 7.11

Specification: The on demand pedestrian crossing should not interrupt the traffic flow (the car friendly solution, just like Koblenz).

Example II

Control of traffic lights

Description of the state of the three traffic lights:
We use the propositions *pgreen*, *pyellow*, *pred and predyellow* and *cgreen*, *cyellow*, *cred and credyellow* with the following meaning

- $pgreen =$ "The pedestrian traffic light is green for cars."
- $cgreen =$ "The traffic light at the cross roads is green for cars."
- ...

Specification: The on demand pedestrian crossing should not interrupt the traffic flow. The formula

$$s \equiv \neg((pyellow \vee pred \vee predyellow) \wedge cgreen)$$

must always be true.

Example III

Control of traffic lights

Control:

- The traffic light at the cross roads follows the usual sequence $credyellow - cgreen - cyellow - cred$.
- The pedestrian traffic light can only switch from $pgreen - pyellow$ if the traffic light at the crossing is in state $cyellow$.

Example IV

Control of traffic lights

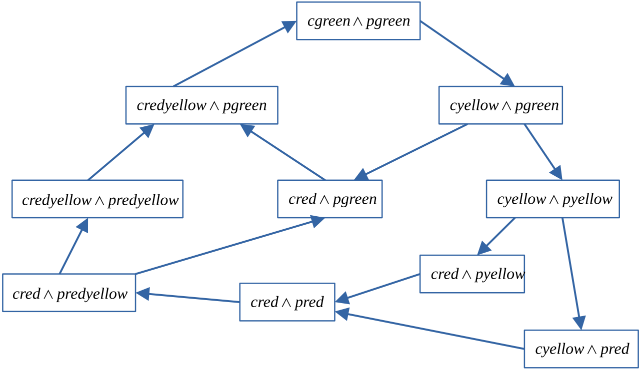

Figure: The state transition graph for the traffic lights.

Example V

Control of traffic lights

Additional formulas ensure that each traffic light can only be in one of the four states *green*, *yellow*, *red*, or *redyellow*:

$$
\begin{aligned}
&(\mathit{cred} \wedge \neg \mathit{credyellow} \wedge \neg \mathit{cgreen} \wedge \neg \mathit{cyellow}) \\
\vee&(\neg \mathit{cred} \wedge \mathit{credyellow} \wedge \neg \mathit{cgreen} \wedge \neg \mathit{cyellow}) \\
\vee&(\neg \mathit{cred} \wedge \neg \mathit{credyellow} \wedge \mathit{cgreen} \wedge \neg \mathit{cyellow}) \\
\vee&(\neg \mathit{cred} \wedge \neg \mathit{credyellow} \wedge \neg \mathit{cgreen} \wedge \mathit{cyellow})
\end{aligned}
$$

and analogously for the pedestrian trafic light,

Example VI
Control of traffic lights

Model checking:
Add the negated specification

$$\neg s \equiv ((pyellow \vee pred \vee predyellow) \wedge cgreen)$$

one after the other to the formulas valid for each state and generate the empty clause using resolution.

Example VII

Control of traffic lights

Let the state be $cred \wedge pred$. We have to prove that the set of formulas

$$\left.\begin{array}{l} cred \\ pred \end{array}\right\} \text{the current state}$$

$$\left.\begin{array}{l} pyellow \vee pred \vee predyellow \\ cgreen \end{array}\right\} \text{the negated spec}$$

$$\left.\begin{array}{l} \ldots \\ \ldots \end{array}\right\} \text{each traffic light can only be in one state}$$

leads to an empty clause $\emptyset$, i.e. a contradiction. The dots indicate the clauses expressing that each traffic light can only be in one state (see above).

Further applications

Some examples

- Automatic program verification: Increasingly complex software systems are now taking over tasks of more and more responsibility and security relevance.
- Software reuse. Programmers specify the state before and after a program was run and compare it to a software data base to select a module which fits the requirements.
- Automatic theorem proving in mathematics.
- Control system and safety system verification.
- ...

Chap. 3 First order logic

Chap. 4 Applications and limitations of logic in AI

Chap. 5 Probability theory and probabilistic logic

Chap. 6 Bayesian Networks

Chap. 7 Further Approaches