Introduction to Symbolic AI

A summary for the lecture unfortunately known as "AI I"

florian.guthmann@fau.de

March 12, 2025

1 Mathematical Prolegomena

1.1 Set Theory

Set theory is usually defined in terms of first-order logic, a topic which is covered in more depth in section 4.2.

The foundational relation between sets is that of membership. We write $x \in A$ if x to express that x is a member of A. The empty set containing no elements is denoted as $\varnothing$.

The usual relations and operations are the following:

Set equality Set equality is *extensional*, i.e. two sets are said to be equal iff they contain the same elements.

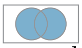
$$A = B \iff (\forall x. x \in A \iff x \in B)$$

Set Inclusion A set A is called a **subset** of a set B iff all elements of A are also elements of B. We write

$$A \subseteq B :\iff (\forall x. x \in A \implies x \in B)$$

A set A is called a **proper subset** of a set B iff $A \subseteq B$ and $A \neq B$. We write $A \subset B$ or $A \subsetneq B$.

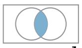
Union



Given two sets A and B we can form a new set, denoted as $A \cup B$, the set that contains all elements of both A and B. Its elements can be characterised as follows:

$$x \in A \cup B \iff x \in A \text{ or } x \in B$$

Intersection

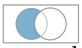


Given two sets A and B we can form a new set, denoted as $A \cap B$, the set that contains those elements which are members of both A and B. Its elements can be characterised as follows:

$$x \in A \cap B \iff x \in A \text{ and } x \in B$$

Two sets A and B are **disjoint** if their intersection $A \cap B$ is empty.

Difference



Given two sets A and B we can form a new set, denoted as $A \setminus B$ (or sometimes $A - B$), the set of all elements of A that are not members of B.

Set Comprehension Given a set A and a formula $P(x)$ over x we can form a new set, denoted as $\{x \in A \mid P(x)\}$, the set of all elements $x \in A$ for which $P(x)$ holds.

Family of sets Given a set I called the **index set**, if we can associate to any $i \in I$ a set A_i we call $(A_i)_{i \in I}$ a **family** of sets indexed over I.

Big union/ Big intersection Given a family $(A_i)_{i \in I}$ we can form a new set, denoted as $\bigcup_{i \in I} A_i$, the set containing all elements of all A_i. Its elements can be characterised as follows:

$$x \in \bigcup_{i \in I} A_i \iff \exists i \in I. x \in A_i$$

Likewise, we can form the set $\bigcap_{i \in I} A_i$ of those elements that are members of all A_i:

$$x \in \bigcap_{i \in I} A_i \iff \forall i \in I. x \in A_i$$

Note how the union and intersection of two sets are just special cases of their big counterparts with a two element index set.

Disjoint Union Let $(A_i)_{i \in I}$ be a family of sets. Then

$$\biguplus_{i \in I} A_i := \{(i, a) \mid i \in I, a \in A_i\}$$

is their **disjoint union**. For a two-element index set $I := \{0, 1\}$ we write $A_0 \uplus A_1$.

Cartesian Product Given two sets A and B we can form a new set, denoted as $A \times B$, of all pairs of elements of A and B.

$$A \times B := \{(x, y) \mid x \in A, y \in B\}$$

Power Set Given a set A, the collection of all subsets of A is also a set, denoted as $\mathcal{P}(A)$.

$$\mathcal{P}(A) := \{B \mid B \subseteq A\}$$

One may therefore use $B \subseteq A$ and $B \in \mathcal{P}(A)$ interchangeably.

Kleene star Given a set A of, the kleene star (or free monoid) A^* is the set of "words" using "characters" of A. The empty word is denoted as $\varepsilon \in A^*$.

1.1.1 Relations and Functions

Def. A (binary) **relation** between two sets A and B is a subset $R \subseteq A \times B$. For $x \in A$, $y \in B$ one may write $x\, R\, y$ instead of $(x, y) \in R$.

Def (Inverse Relation)**.** For any binary relation $R \subseteq A \times A$ there exists the inverse relation

$$R^{-} := \{(y, x) \mid (x, y) \in R\}$$

Def. Given two binary relations $R \subseteq A \times B$, $S \subseteq B \times C$, their **composition** $(S \circ R) \subseteq A \times C$ is given by

$$S \circ R := \{(x, z) \mid \exists y \in B. (x, y) \in R \wedge (y, z) \in S\}$$

Def. Given a relation $R \subseteq A \times A$ we define for $n \in \mathbb{N} \setminus \{0\}$:

$$R^1 := R$$

$$R^{n+1} := R \circ R^n$$

Def (Function)**.** A relation $f \subseteq A \times B$ is called

left total iff for any $x \in A$ there exists a $y \in B$ with $x\, f\, y$

right unique iff for any $x \in A$, $y, z \in B$ with $x\, f\, y$ and $x\, f\, z$ it follows that $y = z$

A relation that is both left total and right unique is called a **function**. We denote such a relation as $f\colon A \to B$. For any $x \in A$ there is a uniquely determined element in B, which we denote $f(x)$, such that $(x, f(x)) \in f$.

We denote the **domain** $\operatorname{dom}(f) := A$ and the **codomain** $\operatorname{codom}(f) := B$.

Def. Given two functions $f\colon A \to B$ and $g\colon B \to C$ their **composition** $(g \circ f)\colon A \to C$[1] is the function given by

$$(g \circ f)(x) = g(f(x))$$

Def (Image and Preimage)**.** Let $f\colon A \to B$ be a function and $U \subseteq A$ a subset of A. We call the set

$$f(U) := \{f(x) \mid x \in U\}$$

the **image** of U. Now let $W \subset B$ be a subset of B. We call the set

$$f^{-1}(W) := \{x \in A \mid f(x) \in W\}$$

the **preimage** of W.

Def (Properties of functions)**.** Let $f\colon A \to B$ be a function. We call f

injective iff for any $x, y \in A$ with $f(x) = f(y)$ it follows that $x = y$ (i.e. the preimage $f^{-1}(\{y\})$ contains at most one element for any $y \in B$)

[1] Note that some authors use $f\,;\,g$ (or even $f \circ g$) to denote the same function, switching the order of f and g from "applicative" (like in $g \circ f$) to "diagrammatic".

surjective iff for any $y \in B$ there exists a $x \in A$ such that $f(x) = y$ (i.e. $f(A) = B$)

bijective iff it is both injective and surjective

Def ((Co)Restriction)**.** Let $f\colon A \to B$ be a function and $U \subseteq A$ a subset of its domain. The **restriction** $f|_U$ of f to U is the function

$$\begin{aligned} f|_U\colon U &\to B \\ u &\mapsto f(u) \end{aligned}$$

Now let $S \subseteq B$ be a subset of f's codomain such that $f(A) \subseteq S$. Then the **corestriction** $f|^S$ is the function

$$\begin{aligned} f|^S\colon A &\to S \\ a &\mapsto f(a) \end{aligned}$$

Def (Partial Function)**.** A relation $f \subseteq A \times B$ that is right unique is called a **partial function** $f\colon A \rightharpoonup B$. For $x \in A$, if it exists, the unique $y \in B$ such that $(x, y) \in f$ is denoted as $f(x)$.
Equivalently, a partial function $f\colon A \rightharpoonup B$ is a function $f\colon U \to B$ where $U \subseteq A$. The **domain** is then $\text{dom}(f) := U$.

Def (Properties of Relations)**.** Let A be a set and $R \subseteq A \times A$ be a relation. R is called

reflexive iff $x\ R\ x$ for any $x \in A$

symmetric iff for all $x, y \in A$ with $x\ R\ y$ it follows that $y\ R\ x$

transitive iff for all $x, y, z \in A$ with $x\ R\ y$ and $y\ R\ z$ it follows that $x\ R\ z$

antisymmetric iff forall $x, y \in A$ with $x\ R\ y$ and $y\ R\ x$ it follows that $x = y$

Def. A relation $\sim\, \subseteq A\times$ that is reflexive, symmetric and transitive is called an **equivalence**.

Def. A relation $\prec\, \subseteq A\times$ that is reflexive, antisymmetric and transitive is called a **partial order**.

Def (Reflexive Closure)**.** Given a relation $R \subseteq A \times A$, its **reflexive closure** $R \cup \text{id}$ is the smallest reflexive relation containing R.

Def (Symmetric Closure)**.** Given a relation $R \subseteq A \times A$, its **symmetric closure** $R \cup R^{-}$ is the smallest symmetric relation containing R.

Def (Transitive Closure)**.** Given a relation $R \subseteq A \times A$, its **transitive closure** R^+ is the smallest transitive relation containing R. It is given by

$$R^+ := R \cup (R \circ R) \cup (R \circ R \circ R) \cup \cdots = \bigcup_{n=1}^{\infty} R^n$$

Def. A set A is **finite** with **cardinality** $|A| \in \mathbb{N}$ if there is a bijection $\varrho\colon A \to \{n \in \mathbb{N} \mid n < |A|\}$.

Ex (Cardinalities)**.**

- $|\varnothing| = 0$
- $|\{\text{foo}, \text{bar}, \text{baz}\}| = 3$

Def. A set A is **countable** if there is a bijection $\varrho\colon A \to \mathbb{N}$.

1.1.2 Examples: Algebraic Structures

Equipping sets with operations and laws for those operations leads to several natural structures. Functions between those "sets with structure" that behave well (i.e. are "structure preserving") are called **homomorphisms**.[2]

Def. A **magma** $(M, \otimes)$ is a set M with a binary operation $\otimes\colon M \times M \to M$.

A **magma-homomorphism** ϱ between two magmas $(M, \otimes)$, $(N, \oplus)$ is a function $\varrho\colon M \to N$ such that for all $a, b \in M$

$$\varrho(a \otimes b) = \varrho(a) \oplus \varrho(b)$$

Def. A **monoid** $(M, \otimes, e)$ is a magma $(M, \otimes)$ together with a **neutral element** $e \in M$ such that

- $\otimes\colon M \times M \to M$ is **associative**:
  $$\forall x, y, z \in M.\, (x \otimes y) \otimes z = x \otimes (y \otimes z)$$
- $e \in M$ is neutral:
  $$\forall x \in M.\, x \otimes e = x = e \otimes x$$

A **monoid homomorphism** ϱ between two monoids $(M, \otimes, e_M)$, $(N, \oplus, e_N)$ is a magma-homomorphism $\varrho\colon M \to N$ such that

$$\varrho(e_M) = e_N$$

Ex (Monoids)**.**

A^*: For any set A, the kleene-star A^* forms a monoid with word concatenation and the empty word.

strings: In most programming languages strings with string concatenation and the empty string form a monoid. This is in fact a special case of the above example[3] with $A := \text{char}$

endo-functions For any set A, the set of "endo"-functions $A^A := \{f\colon A \to A\}$ on A forms a monoid with function composition and the neutral element

$$\begin{aligned} \text{id}_A\colon A &\to A \\ a &\mapsto a \end{aligned}$$

1.2 Computability Theory

2 Rational Agents

Def. An **agent** is an entity that

- **perceives** (via **sensors**)
- **acts** (via **actuators**)

Def (Agent function)**.** A **percept** is the perceptual input of an agent at some instant.

A **action** is an employment of actuators. Let a be an agent that perceives percepts from a set P and can perform actions from a set A. The **agent function** f_a of a is a function

$$f_a\colon P^* \to A$$

Def. An **agent program** is an algorithm that implements an agent function.

Def. A **performance measure** is a function evaluating a sequence of environments.

An agent acts **rationally** if its choice of actions maximise the expected value of the performance measure.

Def (PEAS)**.** A **task environment** is given by

- **P**erformance measure
- **E**nvironment
- **A**ctuators
- **S**ensors

[2] This would quite naturally lead to a discussion of category theory, but that is beyond the scope of this lecture and summary

[3] shying away from any unicode shenanigans

Environments An environment E of an agent a is called

- **fully observable** if a's sensors have access to the complete state of E (else **partially observable**).
- **deterministic** if the next state of E is completely determined by its current state and a's action (else **stochastic**).
- **episodic** if E can be divided into *atomic* (where a perceives and performs a single action) episodes (else **sequential**).
- **dynamic** if E can change without a performing an action, **semidynamic** if only the performance measure changes (else **static**)
- **discrete** if the set of states of E and the set of actions of a are countable (else **continuous**)
- **single agent** if only one agent acts on the environment (else **multi-agent**)

2.1 Agent Types

Simple reflex agent An agent that bases its actions only on the last percept. The agent function reduces to $f_a: P \to A$.

Model-based agent A reflex agent that maintains a world model to determine its actions. The agent function depends on

- a set S of states
- a sensor model $\varrho: S \times P \to S$ that determines the next state given the current state and a percept
- a transition model $\tau: S \times A \to S$
- an action function $f: S \to A$

The agent function is then given by $p \mapsto f(\tau(\varrho(s,p),a))$.

Goal-based agent A model-based agent with a transition model $T: S \to S$ and a set $G \subseteq S$ of goals. Its goal function f selects an action to best reach G.

Utility-based agent An agent with a world model and a utility function that evaluates states. The agent chooses actions to maximise the expected utility.

Def. A state representation is

- **atomic** if it has no internal structure
- **factored** if each state is characterized by attributes and their values
- **structured** if each state includes representations of objects and their relationships

3 Solving Problems by Searching

Def (Search Problem)**.** A **search problem** is a tuple (S, A, τ, I, G) where

- S is a set of **states**
- A is a set of **actions**
- $\tau: A \times S \to \mathcal{P}(S)$ is a **transition model** that assigns to an action and a state a set of successor states
- $I \subseteq S$ is a set of **initial states**
- $G \subseteq S$ is a set of **goal states**

A **solution** to a search problem (S, A, τ, I, G) is a sequence $a_1, a_2, \ldots, a_n$ of actions such that there exists a sequence s_0, s_1, s_n of states where

- $s_0 \in I$
- $\tau(a_i, s_{i-1}) \neq \varnothing$ for all $1 \leq i < n (a_i$ is **applicable** to $s_{i-1})$
- $s_i \in \tau(a_i, s_{i-1})$ for all $1 \leq i < n$
- $s_n \in G$

Def. Let $\Pi := (S, A, \tau, I, G)$ be a search problem. A **cost function** is a function $\mathrm{c}: A \to \mathbb{R}^+$ that assigns a cost to an action. The cost of a solution $a_1, a_2, \ldots, a_n$ is given by

$$\sum_{i=1}^{n} \mathrm{c(a_i)}$$

Def. A search problem (S, A, τ, I, G) is called **deterministic** if

- there is exactly one initial state, $I = \{s_0\}$
- $\tau(a, s)$ contains at most one successor state

Def. Let $\Pi := (S, A, \tau, I, G)$ be a search problem. A **heuristic** for Π is a function $h: S \to \mathbb{R}^+ \cup \{\infty\}$ so that $h(s) = 0$ for all $s \in G$.

Def. Let $\Pi := (S, A, \tau, I, G)$ be a search problem. Then the **goal distance function** $\mathrm{h}^\star: S \to \mathbb{R}^+ \cup \{\infty\}$ maps a state s to the cost of the cheapest path from from s to some goal state.

Def. Let $\Pi := (S, A, \tau, I, G)$ be a search problem and $h: S \to \mathbb{R}^+ \cup \{\infty\}$ a heuristic for Π. h is called **admissible** if it always underestimates, i.e.

$$\forall s \in S.\, h(s) \leq \mathrm{h}^\star(s)$$

3.1 Adversarial Search

3.2 Constraint Satisfaction

Def (Constraint Satisfaction Problem)**.** A **constraint satisfaction problem** $(V, (D_v)_{v \in V}, C)$ consist of

- a set of **variables** V
- a **domain** D_v for each variable $v \in V$
- a set C of "constraints" (a proposition containing finitely many variables)

Def. Constraints are classified by the number of constraint variables they involve:

- **Unary** constraints involve a single variable
- **Binary** constraints involve two variables
- **Higher-Order** constraints involve more than two variables

A constraint network is callled **binary** iff all of its constraints are binary.

Prop. Any higher-order constraint can be equivalently expressed by a finite set of binary constraints by introducing additional variables.

Def. Given a binary CSP, a **constraint network** $(V, (D_v)_{v \in V}, C)$ consist of

- a set V of variables
- a domain D_v for each variable $v \in V$
- a set of constraints

$$C := \{C_{u,v} \subseteq D_u \times D_v \mid u, v \in V, u \neq v\}$$

Def. Let $\gamma := (V, (D_v)_{v \in V}, C)$ be a constraint network. A **variable assignment** is a partial function $\varphi: V \rightharpoonup \bigcup_{v \in V} D_v$ such that $\varphi(v) \in D_v$ for all $v \in \mathrm{dom}(\varphi)$. If φ is left total, we call it a **total** variable assignment.

Def. Let $\gamma := (V, (D_v)_{v \in V}, C)$ be a constraint network and $\varphi: V \rightharpoonup \bigcup_{v \in V} D_v$ a variable assignment.
φ **satisifies** a constraint $C_{u,v}$ iff $u, v \in \mathrm{dom}(\varphi)$ and $(\varphi(u), \varphi(v)) \in C_{u,v}$
φ is **consistent** with γ iff it satisfies all constraints in γ.

Def. Let φ, ϱ be variable assignments. φ **extends** ϱ iff $\mathrm{dom}(\varrho) \subseteq \mathrm{dom}(\varphi)$ and $\varphi|_{\mathrm{dom}\,\varrho} = \varrho$ (i.e. ϱ agrees with the restriction of φ to ϱ's domain)

Def. A **solution** of a constraint-network γ is a consistent (total) variable assignment.

3.2.1 Constraint Propagation

Def. Two constraint networks $\gamma := (V, (D_v)_{v \in V}, C)$ and $\gamma' := (V, (D'_v)_{v \in V}, C')$ are **equivalent** iff they have the same solutions. We write $\gamma \equiv \gamma'$

γ' is **tighter** than γ iff

- $D'_v \subseteq D_v$ for all $v \in V$
- $C'_{u,v} \notin C$ or $C'_{u,v} \subseteq C_{u,v}$ for all $u, v \in V$, $u \neq v$ and $C'_{u,v} \in C'$

We write $\gamma' \sqsubseteq \gamma$.

Prop. Let γ, γ' be constraint networks such that $\gamma' \sqsubseteq \gamma$ and $\gamma \equiv \gamma'$. Then γ' has the same solutions, but fewer consistent assignments than γ.

Def (Forward Checking)**.** Let $\gamma := (V, (D_v)_{v \in V}, C)$ be a constraint network, $u \in V$ a variable and φ be a variable assignment for γ such that $u \in \mathrm{dom}(\varphi)$. The process of obtaining an equivalent constraint network $\gamma' := (V, (D'_v)_{v \in V}, C)$ where

$$D'_v = \{d \in D_v \mid C_{u,v} \in C \implies (\varphi(u), d) \in C_{u,v}\}$$

is called **forward checking**.

Def (Arc Consistency)**.** Let $\gamma := (V, (D_v)_{v \in V}, C)$ be a constraint network. A variable $u \in V$ is **arc consistent** relative to $v \in V$ if either $C_{u,v} \notin C$ or for every $d \in D_u$ there exists a $t \in D_v$ such that $(d, t) \in C_{u,v}$. γ is arc consistent if every variable $u \in V$ is arc consistent to every variable $v \in V$.

The process of obtaining an equivalent constraint network $\gamma' := (V, (D'_v)_{v \in V}, C)$ where

$$D'_v = \bigcap_{u \in V} \{d \in D_v \mid C_{v,u} \in C \implies \exists d' \in D_u.\, (d, d') \in C_{v,u}\}$$

is called **arc consistency**.

4 Logic

4.1 Propositional Logic

The set $\mathrm{P}(\mathcal{V})$ of formulae of propositional logic are given by

$A, B :=$	X	variable
$\mid$	$\top$	truth
$\mid$	$\bot$	falsity
$\mid$	$\neg A$	negation
$\mid$	$A \wedge B$	conjunction
$\mid$	$A \vee B$	disjunction
$\mid$	$A \implies B$	implication
$\mid$	$A \iff B$	equivalence

where $X \in \mathcal{V}$ is in the set of variables $\mathcal{V}$.

Def. A **model** $(\mathcal{D}, [\![-]\!]_{\_})$ for propositional logic consist of

- a **universe** $\mathcal{D}$ (typically the two-element boolean algebra)
- an interpretation function $[\![-]\!]$ that assigns meaning to all connectives
- a family of **value functions** $[\![-]\!]_\varphi : \mathrm{P}(\mathcal{V}) \to \mathcal{D}$ where $\varphi\colon \mathcal{V} \to \mathcal{D}$ is a **variable assignment**.

  It is defined recursively using the interpretation function:

$$[\![X]\!]_\varphi = \varphi(X)$$
$$[\![\neg A]\!]_\varphi = [\![\neg]\!]\left([\![A]\!]_\varphi\right)$$
$$[\![A \wedge B]\!]_\varphi = [\![\wedge]\!]\left([\![A]\!]_\varphi, [\![B]\!]_\varphi\right)$$
$$\vdots$$

Two formulae A and B are called **equivalent** iff $[\![A]\!]_\varphi = [\![B]\!]_\varphi$ for all assignments φ.

$\wedge$	$\bot$	$\top$
$\bot$	$\bot$	$\bot$
$\top$	$\bot$	$\top$

$\vee$	$\bot$	$\top$
$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$

$\implies$	$\bot$	$\top$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$

Def (Entailment)**.** Let φ be a variable assignment, A a propositional formula. We write $\varphi \vDash A$ for $[\![A]\!]_\varphi = \top$.

Now let B be a propositional formula. If it holds that for all φ such that $\varphi \vDash A$ it is also the case that $\varphi \vDash B$, then we write $A \vDash B$.

Def. Let $\mathcal{M} := (\mathcal{D}, [\![-]\!]_{\_})$ be a model. A formula A is called

- **true under** φ if $[\![A]\!]_\varphi = \top$
- **false under** φ if $[\![A]\!]_\varphi = \bot$
- **satisfiable** in $\mathcal{M}$ if there exists a φ such that $[\![A]\!]_\varphi = \top$
- **valid** in $\mathcal{M}$ if $[\![A]\!]_\varphi = \top$ for all φ
- **falsifiable** in $\mathcal{M}$ if there exists a φ such that $[\![A]\!]_\varphi = \bot$
- **unsatisfiable** in $\mathcal{M}$ if $[\![A]\!]_\varphi = \bot$ for all φ

Def (Deduction)**.** A relation $\vdash_C \subseteq \mathcal{P}(\mathrm{P}(\mathcal{V})) \times \mathrm{P}(\mathcal{V})$ is called a **derivation relation** iff

- $\Gamma \vdash_C A$ if $A \in \Gamma$
- if $\Gamma \vdash_C A$ and $\Gamma' \cup \{A\} \vdash_C B$ then $\Gamma \cup \Gamma' \vdash_C B$
- if $\Gamma \vdash_C A$ and $\Gamma \subseteq \Gamma'$ then $\Gamma' \vdash_C A$

Def. A formula A is called a **theorem** in a calculus C if there exists a **proof** $\vdash_C A$.

Def (Inference Rule)**.** Derivation relations are typically defined inductively, i.e. via a set C of **inference rules** like

$$\frac{\Gamma \vdash_C A \qquad \Gamma \vdash_C A \implies B}{\Gamma \vdash_C B}$$

An inference rule $\dfrac{\Gamma \vdash A_1 \ldots \Gamma \vdash A_n}{C}$ is called **derivable** in a calculus $\vdash_C$ if there is a derivation $A_1, \ldots, A_n \vdash_C C$.

An inference rule is called **admissible** in a calculus C if its addition does not produce new theorems.

Def. Let $\vdash_C$ be a derivation relation. There are two ways to relate deduction and entailment:

Soundness $\vdash_C$ is **sound** if whenever $A \vdash_C B$ then $A \vDash B$.

Completeness $\vdash_C$ is **complete** if whenever $A \vDash B$ then $A \vdash_C B$

4.1.1 Propositional Natural Deduction

A bracketed formula like $[A]$ indicates that its proof is in **context**. A context is a set of formulae that we currently assume to be true. Taking the introduction rule for implication as an example, we can see that this means that to prove $A \implies B$ we must provide a proof of B, assuming A.

Sequent Style We can make this more explicit by switching to "sequent-style" natural deduction. This introduces the operator $\vdash$, which takes as its left argument a context and as its right argument a formula. $\Gamma \vdash A$ asserts that A can be proven using only the context Γ. We can change most natural deduction rules that do not involve contexts quite easily, i.e. $\wedge_\mathrm{I}$ becomes

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \wedge B}\ \wedge_\mathrm{I}$$

For notational convenience, we write Γ, A for the context obtained "extending" a context Γ with a formula A, i.e. $\Gamma \cup \{A\}$. For a singleton context $\{A\}$ we will omit the curly braces and just write A.

Those rules where we previously used bracketed formulae to indicate assumptions are changed as follows:

$$\frac{\Gamma \vdash A \lor B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{C} \lor_E$$

$$\frac{\Gamma, A \vdash B}{A \implies B} \implies_I \qquad \frac{\Gamma, A \vdash C \quad \Gamma, A \vdash \neg C}{\neg A} \neg_I$$

Figure 1: Sequent-style propositional rules of natural deduction

Since we want to be more explicit about the use of contexts, we add a separate rule for proving a formula that is in the current context:

$$\frac{P \in \Gamma}{\Gamma \vdash P} \text{ctx}$$

Fitch Style This is another notation used prominently in the *GLoIn* lecture[4]. The calculus used is the same, namely natural deduction, but the proof tree is "linearized" such that it can be written down more easily. Consider the two proofs in Figure 2, done in fitch/sequent style: The proofs look similar, ignoring the obvious difference that the fitch "proof tree" grows from left to right, not bottom to top. The main difference is in how we deal with assumptions. Starting a subproof with an assumption in fitch-style corresponds to adding the formula to the current context in a new branch of the sequent-style proof tree. Using a formula from the context is implicit in the fitch proof.

Def (Test calculus)**.** One can exploit the fact that

$$A \text{ valid} \iff \neg A \text{ unsatisfiable}$$

This means that to prove a formula A valid, it suffices to show that $\neg A \vdash_T \bot$.

4.1.2 Propositional Tableau

$$\frac{(A \land B)^\top}{\begin{matrix}(A)^\top \\ (B)^\top\end{matrix}} \qquad \frac{(A \land B)^\bot}{(A)^\bot \quad (B)^\bot}$$

$$\frac{(\neg A)^\top}{(A)^\bot} \qquad \frac{(\neg A)^\bot}{(A)^\top}$$

$$\frac{(A \implies B)^\top}{(A)^\bot \quad (B)^\top} \qquad \frac{(A \implies B)^\bot}{\begin{matrix}(A)^\top \\ (B)^\bot\end{matrix}}$$

$$\frac{(A \lor B)^\top}{(A)^\top \quad (B)^\top} \qquad \frac{(A \lor B)^\bot}{\begin{matrix}(A)^\bot \\ (B)^\bot\end{matrix}}$$

$$\frac{\begin{matrix}(A)^\alpha \\ (A)^\beta \quad \alpha \neq \beta\end{matrix}}{\bot}$$

Figure 3: Rules of the analytical tableau calculus

[4]This section is mainly intended to help students that have taken that lecture carry over their intution. Fitch style is not actually covered in the lecture at hand.

Def (Tableau)**.** A tree produces by the above inference rules of is called a **tableau**. A tableau is **saturated** if no rule adds new material. A branch is **closed** if it ends in $\bot$. A tableau is **closed** if all of its branches are.

4.1.3 Resolution

Resolution is a test calculus that operates on formulae in conjunctive normal form. The calculus then consist of just two rules:

$$\frac{(A)^\top \lor C \quad (B)^\bot \lor D \quad \sigma = \text{mgu}(A, B)}{\sigma(C) \lor \sigma(D)}$$

$$\frac{A^\alpha \lor B^\alpha \lor C \quad \sigma = \text{mgu}(A, B)}{\sigma(A) \lor \sigma(C)}$$

Figure 4: Rules of the resolution calculus

Unification

$$S \cup \{x \doteq x\} \to S \qquad \text{(delete)}$$

$$S \cup \{f(E_1, \ldots, E_n) \doteq f(D_1, \ldots, D_n)\} \to S \cup \{E_1 \doteq D_1, \ldots, E_n \doteq D_n\} \qquad \text{(decomp)}$$

$$S \cup \{f(E_1, \ldots, E_n) \doteq g(D_1, \ldots, D_m)\} \to \bot \qquad \text{(conflict)}$$

$$S \cup \{E \doteq x\} \to S \cup \{x \doteq E\} \qquad \text{(orient)}$$

$$S \cup \{x \doteq E\} \to \begin{cases} \bot & x \in \text{free}(E), x \neq E \\ S\ [E/x] \cup \{x \doteq E\} & x \notin \text{free}(E), x \in \text{free}(S) \end{cases} \qquad \text{(occurs/elim)}$$

4.2 First-Order Logic

Def (Signature)**.** A signature is a tuple $\Sigma \coloneqq (\Sigma^f, \Sigma^p, \text{ar})$ where

- Σ^f is a set of **function symbols**
- Σ^p is a set of **predicate symbols**
- $\text{ar}\colon \Sigma^f \uplus \Sigma^p \to \mathbb{N}$ is a function assigning each symbol an **arity**, i.e. the number of arguments it takes.

We write Σ^f_n and Σ^p_n for the sets of n-ary function and predicate symbols, respectively.

Def (Terms)**.** Let $\mathcal{V}$ be a set of variables, Σ a signature. The set of **terms** $\text{wf}_\iota(\mathcal{V}, \Sigma)$ is defined by

- $\mathcal{V} \subseteq \text{wf}_\iota(\mathcal{V}, \Sigma)$
- if $f \in \Sigma^f_n$ and $A_1, \ldots A_n \in \text{wf}_\iota(\mathcal{V}, \Sigma)$ then $f(A_1, \ldots, A_n) \in \text{wf}_\iota(\mathcal{V}, \Sigma)$

Def (Propositions)**.** Let $\mathcal{V}$ be a set of variables, Σ a signature. The set of **propositions** $\text{wf}(\mathcal{V}, \Sigma)$ is defined by

- if $P \in \Sigma^p_n$ and $A_1, \ldots, A_n \in \text{wf}_\iota(\mathcal{V}, \Sigma)$ then $P(A_1, \ldots, A_n) \in \text{wf}(\mathcal{V}, \Sigma)$
- if $A, B \in \text{wf}(\mathcal{V}, \Sigma)$ then $\neg A, A \land B, A \lor B, A \implies B \in \text{wf}(\mathcal{V}, \Sigma)$
- $\top, \bot \in \text{wf}(\mathcal{V}, \Sigma)$
- if $v \in \mathcal{V}$ and $A \in \text{wf}(\mathcal{V}, \Sigma)$ then $\forall v.\, A, \exists v.\, A \in \text{wf}(\mathcal{V}, \Sigma)$

Def (Free Variables)**.** Given a formula A, the set $\text{free}(A) \subset \mathcal{V}$ of **free** variables of A contains those variables

1. $A \lor B$	Premise
2. A	Assumption
3. $B \lor A$	$\lor_{\mathrm{Ir}}$
4. B	Assumption
5. $B \lor A$	$\lor_{\mathrm{Il}}$
6. $B \lor A$	$\lor_{\mathrm{E}}$

$$\dfrac{\dfrac{}{(A \lor B) \vdash A \lor B}\,\text{ctx} \qquad \dfrac{\dfrac{}{(A \lor B), A \vdash A}\,\text{ctx}}{(A \lor B), A \vdash B \lor A}\,\lor_{\mathrm{Ir}} \qquad \dfrac{\dfrac{}{(A \lor B), B \vdash B}\,\text{ctx}}{(A \lor B), B \vdash B \lor A}\,\lor_{\mathrm{Il}}}{(A \lor B) \vdash B \lor A}\,\lor_{\mathrm{E}}$$

Figure 2: Comparison of Fitch and Sequent Styles

in A that are not **bound** by a quantifier.

$\mathrm{free}(v) = \{v\}$

$\mathrm{free}(f(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathrm{free}(A_i)$

$\mathrm{free}(P(A_1, \ldots, A_n)) = \bigcup_{i=1}^{n} \mathrm{free}(A_i)$

$\mathrm{free}(\bot) = \mathrm{free}(\top) = \varnothing$

$\mathrm{free}(A \land B) = \mathrm{free}(A \lor B) = \mathrm{free}(A) \cup \mathrm{free}(B)$

$\mathrm{free}(\forall v.\, A) = \mathrm{free}(\exists v.\, A) = \mathrm{free}(A) \setminus \{v\}$

Def (Substitution)**.** A **substitution** is a function $\sigma\colon \mathcal{V} \to \mathrm{wf}_\iota(\mathcal{V})$ with finite **support** (i.e the set $\{x \mid x \neq \sigma(x)\}$ is finite). We denote by $[A/X]$ the substitution that maps the variable X to the term A and behaves like the identity function on all other variables. A perhaps preferable notation would be $[X := A]$, but we will stick with the above.

Applying a substitution σ to a term/formula is done via recursion over the syntatic structure:

On terms

$v\,\sigma = \sigma(v)$ (where $v \in \mathcal{V}$)

$f(A_1, \ldots, A_n)\,\sigma = f(A_1\,\sigma, \ldots, A_n\,\sigma)$ (where $f \in \Sigma^{\mathrm{f}}_n$, $A_1, \ldots, A_n \in \mathrm{wf}_\iota(\mathcal{V})$)

On formulae

$P(A_1, \ldots, A_n)\,\sigma = P(A_1\,\sigma, \ldots, A_n\,\sigma)$

(where $P \in \Sigma^{\mathrm{p}}_n$ and $A_1, \ldots, A_n \in \mathrm{wf}_\iota(V, \Sigma)$)

$\bot\sigma = \bot$

$(\neg A)\,\sigma = \neg(A\,\sigma)$

$(A \land B)\,\sigma = A\,\sigma \land B\,\sigma$

$\vdots$

$$(\forall X.\, A)\,\sigma = \begin{cases} \forall X.\, (A\,\sigma) & X \notin \{\mathrm{free}(B) \mid B \in \sigma(\mathcal{V})\} \\ \forall X'.\, ((A\ [X'/X])\,\sigma) & \text{otherwise} \end{cases}$$

(where X' is a fresh variable)

4.2.1 First-Order Natural Deduction

We extend propositional natural deduction with the rules shown in Figure 5.

4.2.2 Free Variable Tableau

This tableau calculus extends the propositional tableau with the rules shown in Figure 6.

$$\dfrac{(\forall X.\, A)^{\top} \qquad Y \text{ fresh}}{(A\ [Y/X])^{\top}}$$

$$\dfrac{(\forall X.\, A)^{\bot} \qquad \{X_1, \ldots, X_k\} = \mathrm{free}(\forall X.\, A) \qquad f \in \Sigma^{\mathrm{sk}}_k \text{ new}}{(A\ [f(X_1, \ldots, X_k)/X])^{\bot}}$$

$$\dfrac{(\exists X.\, A)^{\top} \qquad \{X_1, \ldots, X_k\} = \mathrm{free}(\exists X.\, A) \qquad f \in \Sigma^{\mathrm{sk}}_k \text{ new}}{(A\ [f(X_1, \ldots, X_k)/X])^{\top}}$$

$$\dfrac{(\exists X.\, A)^{\bot} \qquad Y \text{ fresh}}{(A\ [Y/X])^{\bot}}$$

Figure 6: Additional Rules of the Free Variable Tableau

4.2.3 First-Order Resolution

$$\dfrac{\{\forall X.\, A \lor C\} \qquad Z \notin (\mathrm{free}(A) \cup \mathrm{free}(C))}{\{(A\ [Z/X]) \lor C\}}$$

$$\dfrac{\{\exists X.\, A \lor C\} \qquad \{X_1, \ldots, X_k\} = \mathrm{free}(\exists X.\, A) \qquad f \in \Sigma^{\mathrm{sk}}_k}{\{(A\ [f(X_1, \ldots, X_k)/X]) \lor C\}}$$

Figure 7: First-Order CNF-Calculus

4.3 Knowledge Representation

4.3.1 Semantic Networks

Def. A **semantic network** is a directed graph where
nodes represent objects and concepts
edges represent relations between nodes

4.3.2 $\mathcal{ALC}$

4.4 Planning

References

[1] Gerhard Gentzen. "Untersuchungen über das logische Schließen I". In: *Mathematische Zeitschrift* 39 (1935), pp. 176–210.

[2] Stuart Russell and Peter Norvig. *Artificial Intelligence, Global Edition A Modern Approach*. Pearson Deutschland, 2021, p. 1168. ISBN: 9781292401133. URL: https://elibrary.pearson.de/book/99.150005/9781292401171.

$$\frac{\Gamma \vdash A\ [C/X] \qquad C \notin \mathrm{free}(\Gamma)}{\Gamma \vdash \forall X.\, A}\ \forall_{\mathrm{I}} \qquad\qquad \frac{\Gamma \vdash \forall X.\, A}{\Gamma \vdash A\ [B/X]}\ \forall_{\mathrm{E}}$$

$$\frac{\Gamma \vdash A\ [E/X]}{\Gamma \vdash \exists X.\, A}\ \exists_{\mathrm{I}} \qquad\qquad \frac{\Gamma \vdash \exists X.\, A \qquad \Gamma, (A\ [c/X]) \vdash C \qquad c \in \Sigma_0^{\mathrm{sk}}\ \mathrm{new}}{\Gamma \vdash C}\ \exists_{\mathrm{E}}$$

Figure 5: Additional Rules of FO Natural Deduction