Notation for Logical Operators:

$\top$	always true
$\bot$	always false
$\wedge$	$\ldots$ and $\ldots$
$\vee$	$\ldots$ or $\ldots$
$\rightarrow$	if $\ldots$ then $\ldots$
$\leftrightarrow$	$\ldots$ if-and-only-if $\ldots$
$\forall x{:}\,X\ p(x)$	for all x of type X, $p(x)$
$\exists x{:}\,X\ p(x)$	there exists an x of type X, s.t. $p(x)$
$=$	$\cdots$ is equal to $\cdots$
$p(t_1, \ldots, t_n)$	$t_1, \ldots, t_n$ stand in relation p to each other.

(Formulas of form $p(t_1, \ldots, t_n)$ or $t_1 = t_2$ are called atoms)

The operators $\forall x\colon X$ and $\exists x\colon X$ are called quantifiers.

In the rest of this course, we will nearly always omit the type X.

Examples:

Some birds can fly:

$$\exists x \; [B(x) \wedge CF(x)].$$

Some birds can swim:

$$\exists x \; [B(x) \wedge CS(x)].$$

Some birds can swim and fly:

$$\exists x \; [B(x) \wedge CF(x) \wedge CS(x)].$$

No bird can count:

$$\neg\exists x\ [B(x) \wedge CC(x)].$$

All birds cannot count:

$$\forall x\ [B(x) \rightarrow \neg CC(x)].$$

If something can count, then it is not a bird:

$$\forall x\ [CC(x) \rightarrow \neg B(x)].$$

There exists a bird:

$$\exists x\ B(x).$$

There exist at least two birds:

$$\exists x_1 x_2\ [x_1 \neq x_2 \wedge B(x_1) \wedge B(x_2)].$$

There exist at most one bird:

$$\forall x_1 x_2 [B(x_1) \wedge B(x_2) \rightarrow x_1 = x_2].$$

Examples from Mathematics

The atoms $p(t_1, \ldots, t_n)$ can have form

$3 < 4, \;\; 1 < 1 + 1, \;$ even$(4), \;$ odd(5),

substring$(''cde'', ''abcdefgh'')$.

Examples of formulas are

$\forall x, y{:}\mathbb{N}\; x < y \leftrightarrow x + 1 < y + 1$

$\forall x, y{:}\mathbb{N}\; x \leq y \rightarrow x < y + 1$

$\forall p{:}\mathbb{N}\; \mathrm{prime(p)} \leftrightarrow \neg\exists x{:}\mathbb{N}\; 1 < x \wedge x < p \wedge \mathrm{divides}(x, p)$

$\forall x, y{:}\mathbb{R}\;\; \mathrm{square}(x + y) = \mathrm{square}(x) + \mathrm{square}(y) + 2 * x * y$

Free and Bound Variables

In the formula $\exists x{:}\mathbb{N}\ x > 0$, x acts like a local variable, because it is declared by the $\exists$-quantifier.

We say that the variable x is/occurs bound in the formula $\exists x{:}\mathbb{N}\ x > 0$. The quantifier $\exists x{:}\mathbb{N}$ binds the variable x. Variable occurrences that are not bound, are called free occurrences.

For bound variables, the actual name does not matter. What matters is where and how the variable is declared. $\exists z{:}\mathbb{N}\ z > 0$ denotes the same formula as $\exists x{:}\mathbb{N}\ x > 0$.

QUESTION: Do you know more mathematical operators that are able to bind variables?

ANSWER: For example:

$$\sum_{i=0}^{n} i^2$$

$$\int_0^{\frac{\pi}{2}} sin(x).dx$$

$$\{x \in \mathbb{R} \mid x^2 + 3x - 2 = 0\}$$

First-order, classical logic, What is an order?

- Predicates that speak about simple objects are of 1-st order.
- Predicates that speak about objects of at most i-th order, are by themselves of $(i+1)$-th order.
- Functions that take and return simple objects are of 1-st order.
- Functions that take and return objects of at most i-th order, are by themselves of $(i+1)$-th order.

For example, the induction principle is 2-nd order:
$\forall P{:}\,\mathbb{N} \rightarrow \mathbb{B}\ P(0) \wedge (\forall n{:}\,\mathbb{N}\ P(n) \rightarrow P(n+1)\ ) \rightarrow \forall n{:}\,\mathbb{N}\ P(n)$. For the moment, we consider only first-order logic.

Higher-order, classical logic, what is classical?

When reasoning about physical objects, the following principles are considered valid: $A \vee \neg A$, and $\neg\neg A \rightarrow A$. (law of excluded middle and law of double negation).

When speaking about complicated mathematical objects, the law of excluded middle is problematic.

If one drops excluded middle, one obtains intuitionistic or constructive logic. If one proves a formula of form $\exists x{:}X\, p(x)$ in intuitionistic logic, then one can always find a witness t, s.t. $p(t)$ holds.

Types of Deduction Systems

The most important types of deduction systems are:

- Natural Deduction: Natural Deduction follows the natural style of reasoning, as it can be found in mathematical textbooks or in spoken arguments. Most of the proof consists of forward reasoning, that is deriving conclusions, deriving new conclusions from these conclusions, etc. Occassionally additional assumptions are introduced or dropped.

- Sequent Calculus: In sequent calculus, conclusions and premisses are treated in the same way. The reasoning proceeds by deriving relations between formulas, instead of deriving only conclusions. This is different from the style found in textbooks, but the resulting calculus is easier to use.

- Axiomatic Method: Axiomatic Methods are historically the oldest proof systems, but they are not important anymore. Their distinguishing feature is that logical operators are defined by axioms. There are usually three deduction rules, modus ponens:

$$\text{If } A \text{ and } A \rightarrow B \text{ are provable, then so is } B,$$

generalization

$$\text{If } A \text{ is provable, then so is } \forall x\ A$$

and formula instantiation:

$$\text{If } A \text{ is provable, then so is } A[X := F].$$

Recursive Definition of Formulas

We assume a set of function symbols $\mathcal{F}$. Each function symbol f has an arity ≥ 0 associated to it.

The arity of a function symbol is the number of arguments that it can be applied on.

Constants vs. Variables

We call the functions with arity 0 either constants or variables dependent on how we use them now or intend to use them later. For the rest, they are the same symbols. When bound by a quantifier $\exists x$ or $\forall x$, we call them variables. Otherwise, we call them constants.

Terms are recursively defined as follows:

- If f is a function with arity n, $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is a term.

We assume a set of predicate symbols $\mathcal{P}$. Like the functions, each predicate symbol has an associated arity ≥ 0.

Atoms are defined as follows:

- If p is a function symbol with arity n, $\quad t_1, \ldots, t_n$ are terms, then $p(t_1, \ldots, t_n)$ is an atom.
- If t_1 and t_2 are terms, then $t_1 = t_2$ is an atom.

Note that the definition of atoms is not recursive.

Formulas are recursively defined as follows:

- If A is an atom, then A is a formula.
- $\perp$ and $\top$ are formulas.
- If F is a formula, then $\neg F$ is a formula.
- If F_1 and F_2 are formulas, then $F_1 \wedge F_2, \quad F_1 \vee F_2, \quad F_1 \rightarrow F_2, \quad F_1 \leftrightarrow F_2$ are formulas.
- If x is a variable, F is a formula, then $\forall x\ F$ and $\exists x\ F$ are formulas.

Induction for Formulas

Because formulas and terms have recursive definitions, one can prove properties of formulas and terms by induction: If P is a property, s.t.

- for all atoms A, $\;P(A), .$
- $P(\bot)$ and $P(\top)$,
- if $P(A)$, then $P(\neg A)$,
- if $P(F_1)$ and $P(F_2)$, then $P(F_1 \wedge F_2)$, $\;P(F_1 \vee F_2)$, $\;P(F_1 \rightarrow F_2)$, $\;P(F_1 \leftrightarrow F_2)$,
- if $P(F)$, then $P(\forall x\ F)$, and $P(\exists x\ F)$,

then for each formula F, $\;P(F)$.

Free

We recursively define when a variable (0-arity function) is free in a term:

- If t is a variable or constant, then x is free in t if and only if $x = t$.
- If $n > 0$ then x is free in $f(t_1, \ldots, t_n)$ if and only if x is free in one of the $t_1, \ldots, t_n$.

We define when a variable is free in an atom.

- x is free in $p(t_1, \ldots, t_n)$ iff x is free in one of the terms $t_1, \ldots, t_n$.
- x is free in $t_1 = t_2$ iff x is free in t_1 or x is free in t_2.

We recursively define when a variable is free in a formula:

- x is not free in $\bot$ or $\top$.
- x is free in $\neg F$ iff x is free in F.
- x is free in $F_1 \vee F_2, \quad F_1 \wedge F_2, \quad F_1 \rightarrow F_2, \quad F_1 \leftrightarrow F_2$ iff x is free in F_1 or x is free in F_2.
- x is free in $\forall y\ F$ iff $x \neq y$ and x is free in F.
- x is free in $\exists y\ F$ iff $x \neq y$ and x is free in F.

Substitution

We write $u[x := t]$ for the substitution that replaces variable x by term t in term u.

- If u is a variable or constant, and $u \neq x$, then $u[x := t] = u$.
- If $u = x$, then $u[x := t]$ is t.
- If $n > 0$, then $f(t_1, \ldots, t_n)[x := t]$ equals $f(t_1[x := t], \ldots, t_n[x := t])$.

For an atom, we define:

- $p(t_1, \ldots, t_n)[x := t] = p(t_1[x := t], \ldots, t_n[x := t])$.
- $(t_1 = t_2)[x := t] = t_1[x := t] = t_2[x := t]$.

Substitution in a Formula

- $\bot[x := t] = \bot, \qquad \top[x := t] = \top.$
- $(\neg F)[x := t] = \neg(F[x := t]).$
- $(F_1 \vee F_2)[x := t] = F_1[x := t] \vee F_2[x := t]$. The cases for $\wedge, \rightarrow$, and $\leftarrow$ are analogous.
- If x is not free in $\forall y\ F$, then $(\forall y\ F)[x := t] = \forall y\ F$.
- If x is free in $\forall y\ F$, and y is not free in t, then $(\forall y\ F)[x := t] = \forall y\ (F[x := t])$.
- If x is free in $\forall y\ F$, and y is free in t, then let z a variable which is not free in F and not free in t. Then

  $$(\forall y\ F)[x := t] = \forall z\ (F[y := z][x := t]).$$

  The case for $\exists y$ is analogous.

Renaming, α-Equivalence

Using substitution, one can define when two formulas are renamings of each other. This notion is also called α-equivalence.

We recursively define when $F_1 \equiv_\alpha F_2$:

- $\top \equiv_\alpha \top, \quad \bot \equiv_\alpha \bot$.
- For atoms $F_1, F_2, \quad F_1 \equiv_\alpha F_2$ iff $F_1 = F_2$.
- $(\neg F_1) \equiv_\alpha (\neg F_2)$ iff $F_1 \equiv_\alpha F_2$.
- $(F_1 \vee G_1) \equiv_\alpha (F_2 \vee G_2)$ iff $F_1 \equiv_\alpha F_2$ and $G_1 \equiv_\alpha G_2$. The cases for $\wedge, \rightarrow$ and $\leftrightarrow$ are analogous.
- $(\forall y_1 \ F_1) \equiv_\alpha (\forall y_2 \ F_2)$ iff for some variable z, which is not free in F_1 or F_2, $\quad (F_1[y_1 := z]) \equiv_\alpha (F_2[y_2 := z])$. The case for $\exists y$ is analogous.

Substitution is surprisingly hard to define and implement. When we study the semantics of predicate logic, we will encounter some trivial substitution lemmas with very unpleasant proofs.

One possible way out is using De Bruijn indices.

Replace bound variables in formulas by integers that indicate how many quantifiers one needs to skip in order to find the quantifier that binds the variable. $\#1$ means first quantifier, $\#2$ means second quantifier, etc.

$\forall x \exists y\ p(x, y)$ would be replaced by $\forall\ \exists\ p(\#2, \#1)$.

This also solves the problem of testing for α-equivalence. Formulas with the Bruijn indices are α-equivalent iff they are syntactically equal.

Sequent Calculus for Classical Logic

A multiset is a set that can distinguish how often an element occurs in it, (or alternatively it is a list that cannot see the order of its elements).

Examples:

$$A \vee B,\ A \wedge B,\ A \wedge B,$$

$$A \vee B,\ A \wedge B,\ C \rightarrow D,$$

$$A \wedge B,\ A \vee B,\ A \wedge B.$$

The first and the last multiset are equal.

A sequent is an object of form $\Gamma \vdash \Delta$, in which Γ and Δ are multisets of formulas. We assume that the formulas can be freely replaced by α-variants.

The meaning is: Whenever all of the Γ are true, then at least one of the Δ is true.

Propositional Rules:

$$(\text{axiom}) \quad \frac{}{\Gamma, A \vdash \Delta, A}$$

$$(\text{cut}) \quad \frac{\Gamma, A \vdash \Delta \qquad \Gamma \vdash \Delta, A}{\Gamma \vdash \Delta}$$

Structural Rules:

$$(\text{weakening left}) \quad \frac{\Gamma \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad (\text{weakening right}) \quad \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, A}$$

$$(\text{contraction left}) \quad \frac{\Gamma, A, A \vdash \Delta}{\Gamma, A \vdash \Delta} \qquad (\text{contraction right}) \quad \frac{\Gamma \vdash \Delta, A, A}{\Gamma \vdash \Delta, A}$$

Rules for the truth constants:

$$(\top\text{-left}\ )\ \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta} \qquad (\top\text{-right}\ )\ \frac{}{\Gamma \vdash \Delta, \top}$$

$$(\bot\text{-left}\ )\ \frac{}{\Gamma, \bot \vdash \Delta} \qquad (\bot\text{-right}\ )\ \frac{\Gamma \vdash \Delta}{\Gamma \vdash \Delta, \bot}$$

Rules for $\neg$:

$$(\neg\text{-left}\ )\ \frac{\Gamma \vdash \Delta, A}{\Gamma, \neg A \vdash \Delta} \qquad (\neg\text{-right}\ )\ \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \Delta, \neg A}$$

Rules for $\wedge$ and $\vee$:

$$(\wedge\text{-left})\ \frac{\Gamma, A, B \vdash \Delta}{\Gamma, A \wedge B \vdash \Delta} \qquad (\wedge\text{-right})\ \frac{\Gamma \vdash \Delta, A \quad \Gamma \vdash \Delta, B}{\Gamma \vdash \Delta, A \wedge B}$$

$$(\vee\text{-left})\ \frac{\Gamma, A \vdash \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \vee B \vdash \Delta} \qquad (\vee\text{-right})\ \frac{\Gamma \vdash \Delta, A, B}{\Gamma \vdash \Delta, A \vee B}$$

(one can see from this, that premisses and conclusions are treated in completely the same way)

Rules for $\rightarrow$ and $\leftrightarrow$:

$$(\rightarrow\text{-left})\ \frac{\Gamma \vdash \Delta, A \qquad \Gamma, B \vdash \Delta}{\Gamma, A \rightarrow B \vdash \Delta} \qquad\qquad (\rightarrow\text{-right})\ \frac{\Gamma, A \vdash \Delta, B}{\Gamma \vdash \Delta, A \rightarrow B}$$

$$(\leftrightarrow\text{-left})\ \frac{\Gamma, A \rightarrow B,\ B \rightarrow A \vdash \Delta}{\Gamma, A \leftrightarrow B \vdash \Delta}$$

$$(\leftrightarrow\text{-right})\ \frac{\Gamma \vdash \Delta, A \rightarrow B \qquad \Gamma \vdash \Delta, B \rightarrow A}{\Gamma \vdash \Delta, A \leftrightarrow B}$$

Rules for the quantifiers:

$$(\forall\text{-left }) \ \frac{\Gamma, P[x := t] \vdash \Delta}{\Gamma, \forall x \ P \vdash \Delta} \qquad (\forall\text{-right }) \ \frac{\Gamma \vdash \Delta, P}{\Gamma \vdash \Delta, \forall x \ P}$$

$$(\exists\text{-left }) \ \frac{\Gamma, P \vdash \Delta}{\Gamma, \exists x \ P \vdash \Delta} \qquad (\exists\text{-right }) \ \frac{\Gamma \vdash \Delta, P[x := t]}{\Gamma \vdash \Delta, \exists x \ P}$$

The t is an arbitrary term (of the right type, when we consider types) It must be the case that x is not free in Γ or Δ.

rules for equality:

$$(\text{refl-left})\ \frac{\Gamma,\ t = t \vdash \Delta}{\Gamma \vdash \Delta} \qquad\qquad (\text{refl-right})\ \frac{}{\Gamma \vdash \Delta,\ t = t}$$

$$(\text{repl})\ \frac{t_1 = t_2,\ \Gamma[t_1] \vdash \Delta[t_1]}{t_1 = t_2,\ \Gamma[t_2] \vdash \Delta[t_2]}$$

The last rule means: If $t_1 = t_2$ occurs among the premisses, then arbitrary free occurrences of t_1 in other formulas can be replaced by t_2.

rules for equality (2)

The rule repl can be made precise by means of substitution:

$$(\mathrm{repl})\ \frac{t_1 = t_2,\ \Gamma_1 \vdash \Delta_1}{t_1 = t_2,\ \Gamma_2 \vdash \Delta_2}$$

There must exist Γ, Δ and a variable x, s.t.

$\Gamma_1 = \Gamma[x := t_1], \quad \Delta_1 = \Gamma[x := t_1],$
$\Gamma_2 = \Delta[x := t_2], \quad \Delta_2 = \Delta[x := t_2].$

Preconditions in the Quantifier Rules

1. Can you give an example of a wrong derivation, in case that the condition 'y is not free in Γ, Δ' is dropped from $\forall$-right?

2. Suppose that substitution would not take into account the possible capture of variables. Can you give an example of a wrong derivation using rule $\exists$-right?

1. For example:

$$\dfrac{\dfrac{P(x) \vdash P(x)}{\exists x\ P(x) \vdash P(x)}}{\exists x\ P(x) \vdash \forall x\ P(x)}$$

2. For example:

$$\frac{\vdash \forall y\ P(y,y)}{\vdash \exists x \forall y\ P(t(x),y)}$$

In order to see that this is nonsense, take $P(x,y) := (x = y)$.