Definitions

- A is a set of atomic objects
- C is a set of constructors (thery support arity op $ar(op) \in \mathbb{N}$ defining the amount of operator)
- T is a type of elements which define the set T_n with $T_0 = A$.
- V is the set of sentences
- F is the set of propositional formulas on V
- $F_0 = F_{\{\top,\bot,\neg,\land,\lor,\Rightarrow,\Leftrightarrow\}}$ refers to the set of all well-formed propositional formulas constructed (from the given constants and connectors)
- $\mathcal{T}$ is a set of types

1) Induction

2) Polish notation

3) Proposition formulas The only symbols usable are:

- $\top, \bot, \neg, \land, \lor, \Rightarrow, \Leftrightarrow$
- `\top, \bot, \neg, \land, \lor, \Rightarrow, \Leftrightarrow`
- Tautology, Contradiction, Negation, And, Or, implication, equivalence `Tautology can be seen as "true" and contradiction as "false".`

4) A valuation is a function $v : V \to \{true, false\}$

5) Given a valuation v, the truth assignment function $||_v$ is:

- $|\top|_v = true$
- $|\bot|_v = false$
- Given $x \in V$, $|x|_v = v(x)$

6) A **tautology** is a proposition x where, for any valuation v, $|x|_v = true$. If it always false, then x is sais to be an **antilogy**. If there exists a valuation v such that $|x|_v = true$, then x is said to be **satisfiable**. In other words:

- If it is always true, it is a *8tautology**.
- If it is always false, it is an **antology**.
- If it is sometimes true, it is **satisfiable**.

7) Two propositions x and y are equivalent if, $\forall v, |x|_v = |y|_v$. Then, we write $x \equiv y$

8) Truth Table

A	B	$A \land B$
0	0	0
0	1	0
0	1	0
1	1	1

A	B	$A \lor B$
0	0	0
0	1	1
1	0	1
1	1	1

A	B	$A \Rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

9) Detemining whether a given propositional formula x is satisfiable or not is called the **satisfiability problem (SAT)**

10) A propositional formula x is said to be in **negative normal form (NFF)** if it only contains $\lor$ and $\land$ and if the $\neg$ constructor only appears in front of atomic statements

11) A propositional formula x is said to be in **conjunctive normal form (CFF)** if it is of the form $x = \land_i \lor_j y_{i,j}$ where $y_{i,j}$ is an atomic with a potenial $\neg$ in front of it.

12) `Skipped`.

13) An **axiom** a is a propositional formula φ that is considered true a priori. We write is $\overline{\varphi}^{[a]}$.

14) An **inference rule** r consists in a finite set of **premises** $\{\psi_1, ..., \psi_n\}$ and a conclusion φ. We use the notation $\frac{\psi_1 ... \psi_n}{\varphi}[r]$.
`Please note that an axiom is nothing but an iference rule whose conclusion is the consequence of an empty set of premises`

15) A **Hilbert proof system** P is a set of composed of a finite number of axioms and a possibly infinite number of inference rules.

16) A **propositional substitution** is a function $\sigma : V \to F_0$
Given $\varphi \in F_0$, $\varphi[\sigma]$ is the propositional formula obtained by replacing any instance of a variable $x \in V$ in φ by the formula $\sigma(x)$ if it exists.
In this case, $\varphi[\sigma]$ is called a **subtitution instance** of φ.

17) Deduction in a Hilbert Proof system.
Let T be a tree whose nodes are labelled by F_0. T is a **deduction** under a Hilbert proof system P if:

For any inner node of T labelled by A with n children labelled by $B_1, \dots, B_n$, there exists:
1 - An inference rule $\dfrac{\psi_1 \;\; ... \;\; \psi_n}{[r] \; \varphi}$ in P
2 - A propositional substitution σ

such that:
$A = \varphi[\sigma]$ (the conclusion after substitution)
$B_i = \psi_i[\sigma]$ for all $i \in \{1, \dots, n\}$ (each child matches a substituted premise)

Terminology:
1 The labels $H_1, \dots, H_m$ of T's leaves are called its **hypotheses**
2 The label C of its root is called its **conclusion**

18) Let T be a dedeuction under a Hilbert system P. A leaf of T labelled by a formula $\psi \in F_0$ is said to be **cancelled** if there exists a propositional substitution ψ and an axiom $\overline{\varphi}^{[a]}$ in P such taht $\psi = \varphi[\sigma]$.
If there exists a deduction T under P with uncancelled hypotheses $\{H_1, ..., H_n\}$ and conclusion C, we then write $\{H_1, ..., H_n\} \vdash_P C$.

19) A **proof** under a Hilbert system P is a deduction T under P whose leaves are all cancelled. Its conclusion C is then called a **theorem** of A deduction with an empty set of uncancelled hypotheses, so to speak. P and we write $\vdash_P C$.
We write $\theta(P)$ the set of theorems under the system P.

Hilbert calculus is a Hilbert proof system H containing a single inference rule

$$\frac{Premise_1 ... Premise_n}{Conclusion}[Condition]$$

- The premises are formulas or other derivations.
- The conclusion is derived from them.
- The [condition] specifies constraints (e.g., variable substitutions).

The condition is either an **premises** or one of the **axiom** of Hilbert systems:

Exercises page 23

Natural Deduction is a proof system with hypotheses N

We use one of the symbols $\{\top, \bot, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$ in to feature **introduction** or **elimination**.

$$\frac{}{A \Rightarrow A \vee B}[\mathbf{1}][\vee_1] \quad \frac{}{B \Rightarrow A \vee B}[\mathbf{2}][\vee_2] \quad \frac{}{A \vee B \Rightarrow (A \Rightarrow C) \Rightarrow (B \Rightarrow C) \Rightarrow C}[\mathbf{3}][\vee_3]$$

$$\frac{}{A \wedge B \Rightarrow A}[\mathbf{4}][\wedge_1] \quad \frac{}{A \wedge B \Rightarrow B}[\mathbf{5}][\wedge_2] \quad \frac{}{A \Rightarrow B \Rightarrow A \wedge B}[\mathbf{6}][\wedge_3]$$

$$\frac{}{\bot \Rightarrow A}[\mathbf{7}][\bot] \quad \frac{}{A \Rightarrow B \Rightarrow A}[\mathbf{8}][\Rightarrow_1] \quad \frac{}{(A \Rightarrow B \Rightarrow C) \Rightarrow (A \Rightarrow B) \Rightarrow A \Rightarrow C}[\mathbf{9}][\Rightarrow_2]$$

$$\frac{}{(A \Rightarrow \bot) \Rightarrow \neg A}[\mathbf{10}][\neg_1 \quad \frac{}{A \Rightarrow \neg A \Rightarrow \bot}[\mathbf{11}][\neg_2] \quad \frac{}{A \vee \neg A}[\mathbf{12}][\mathit{Excluded\ middle}]$$

Figure 1: Hilbert Axioms

Example

An *introduction*:

$$\frac{A \qquad B}{A \Rightarrow B}[\Rightarrow]$$

An *elimination*:

$$\frac{A \Rightarrow B \qquad A}{B}[\Rightarrow]$$

Canceling a Leaf

If we can derive a conclusion B under the assumption X, we can discharge X and infer $X \Rightarrow B$ (or even $X \wedge B$)

$$\begin{array}{c} [X]^1 \\ \vdots \\ B \\ X \implies B \end{array}$$

Note that a premise of an inference rule and its matching hypothesis may be matched to the very same leaf.

$$\frac{\overline{A}^1}{A \Rightarrow A}[\Rightarrow]^1$$

Moreover, different rules may cancel the same leaf.

$$\frac{\frac{\overline{A}^{1,2}}{A \Rightarrow A}[\Rightarrow]^1}{A \Rightarrow A \Rightarrow A}[\Rightarrow]^2$$

Exercises page 38

Lambda Calculus

Lambda calculus is a model for functional programming.

- We model functions anonymously: we do not necessarily name them, but we use the symbol λ to state that a term is a function. $\lambda x \cdot M$ stands for a function of argument x and body M
- A function may have more than one argument: $\lambda xy \cdot M$ *Example* $\lambda fx \cdot Plus \cdot x(fx)$ models `let func x f = x + f(x)`.

This model follows the following language:

```
<term> := <variable>|<function>|<application>
<variable> := x   V
<function> :=  <variable> · <term>
<application> := (<term><term>)
```

Note that a variable can appear in different scope and the most-left scope applies. *Example*

$$x\lambda x \cdot (x\lambda x \cdot xx) \equiv x\lambda y \cdot (y\lambda z \cdot zz)$$

Please note that $f^n x \equiv f(f(...(fx)))$ where f is applied n times.

Exercises page 46

Alpha equivalence

We say a term is `-equivalent` if we can cange the name of all the function variable and the overall term does not change its meaning. The notation is $X[x/y]$ to say that we replace every instance of x by y in the term X.

Examples

$$\lambda x \cdot y \neq \lambda y \cdot y$$

This is **not -equivalent** as y “does not belong to the same scope” anymore.

$$\lambda x \cdot x \neq \lambda y \cdot y$$

$$(\lambda x \cdot \lambda y \cdot x)x = (\lambda x \cdot \lambda x \cdot x)x$$

Thos are **-equivalent**

Exercises page 50

If we have a term that is not ambiguous (no free **and** bound variable + no double λx), we say that it follows Barendregt’s convention. *Exercise page 51*

Beta reduction

-reduction is the process of applying a function to its argument in lambda calculus. It is the computational step that simplifies expressions by substituting the argument into the function body. In other word, it only means that you replace the variable of the function by the actual arguments used later on in the term.

$$(\lambda x \cdot M)N \rightarrow_\beta M[N/x]$$

Example

$$(\lambda x \cdot x)y \rightarrow_\beta (x)[x/y] \equiv y$$

Exercise page 54

Termination of -reduction

Given a term M, we say it is:

- In its **Normal form** when there is no term N such as $M \rightarrow_\beta N$
- **-normalizable** when there exists a term N such as $M \rightarrow_\beta^* N$
- **strongly -normalizable** when there exists no infinite reduction sequence.

Exercise page 55

Head reduction strategy (H)

To reduce a term made of chain lambda functions please follow this simple rules:

1) -rename bound variables to avoid name clashing ($\equiv_\alpha$)
2) Replace in the left most function the variable(s) by the next function if it exists ($\to_\beta$)\ *Example* $(\lambda x \cdot xx)y \to_H yy$
3) Repeat until normal form.
4) If the term is a single abstraction (e.g., $\lambda x \cdot M$), simplify M by returning to step 1
5) Stop when the term is irreducible

Left most reduction strategy (I)

Another strategy (much simpler) consist of:

1) Performing a single -conversion step on the leftmost $\lambda x \cdot M$ to which an argument can be matched\ *Example* $x(\lambda y \cdot y)x \to_I xx$
2) Repeat

Exercise page 61

Sample Functions & Church integer

Reminder Please note that $f^n x \equiv f(f(...(fx)))$ where f is applied n times. We also call it $\underline{n} = \lambda fx \cdot f^n x$. Here, $\underline{n}$ is a **Church Numeral**. *Example:* $\underline{3} = \lambda fx \cdot f(f(fx))$

- $True = \lambda xy \cdot x$. It simulates the instruction `if B then X else Y`
- $False = \lambda xy \cdot y$. It simulates the instruction `if B then X else Y`
- $Succ = \lambda nfx \cdot f(nfx)$. It simulates $\underline{n+1}$ by applying f to $\underline{n}$ (here the variable is the church numeral *function* and take 2 parameters).
- $Plus = y y Succ $. It simulates $\underline{n+m}$
- $IsZero = \lambda x \cdot (\lambda y \cdot False)True$. It checks that a function does not have more than a single depth
- $Pair = \delta xyf \cdot fxy$
- $First = \delta p \cdot pTrue$. Note that $First(PairAB) \to_\beta^* A$
- $Second = \delta p \cdot pFalse$. Note that $First(PairAB) \to_\beta^* B$

Exercise Simplify $IsZero\underline{0}$ and $IsZero\underline{2}$ (Correction *page 64*) *Exericses page 65*

Recursion

- A **fixed point** of A is a term M such that $AM \leftrightarrow_\beta^* A$.
- A **fixed point combinator** of A is a term M such that $MA \leftrightarrow_\beta^* A(MA)$.
- A **Curry's Y combinator** is the term $Y = \lambda f \cdot (\lambda x \cdot f(xx))(\lambda y \cdot f(yy))$ is a **fixed point combinator**

Any term A admits at least one fixed point (Because $YA \leftrightarrow_\beta^* A(YA)$).

Exercises page 68

Typed systems

Merely a Hilbert System with types. We say that $\sigma \to \tau$ is a function that takes a parameter of type σ and returns a value of type τ

Exercises page 72

Some terms are not **Typable**. *Example*: $\omega = \lambda x \cdot xx$. Note that any **Typeable system** has finite computation.

Also, there is **no** equivalent to $\neg$.

The "typed proof" are denoted as $\vdash_\mathcal{N}$

Exercises page 75