3801 Logic Notes

Based on the 2012 autumn lectures by Dr I Strouthos.

1/10/12

MATH 3801: General information

Contact email address(es): strouth@math.ucl.ac.uk
i.strouthos@ucl.ac.uk.

Office Hours (provisional): Mondays 16:15 – 17:00
Wednesday 11:00 – 13:00
Thursday 17:15 – 18:00

Room 712 (7th floor)
Department of Mathematics.
(25 Gordon Street).

Information regarding Wednesday's lecture to be made available (by Tuesday afternoon) at:

www.ucl.ac.uk/~ucahist.

—/—

Some textbooks:

1) Bell and Machover: A Course in Mathematical Logic
2) Boolos, Burgess and Jeffres; Computability and Logic.
3) Fostor; Logic, Induction and Sets.
closest 4) Johnstone; Notes on Logic and set theory.
5) Mendelson; Introduction to Mathematical Logic.
6) van Dalen; Logic and Structure.

—/—

Chapter 0: Preliminary notions

Countability:

"A set is countable if we are able to count it":

Definitions: A set S is countable if S is a finite set or if there is a bijection from S to $\mathbb{N}$, the natural numbers.

There is an equivalent definition:
A set S is countable if there is an injective map from S to $\mathbb{N}$, i.e if there exists $f: S \to \mathbb{N}$, such that f is injective

Examples:

1) Any finite set is countable
2) The set $\mathbb{N}$ is countable.
3) The set $\mathbb{Z}$, of integers, is countable.
   not key "counting" $0, 1, 2, 3, \ldots, -1, -2, -3, \ldots$
   valid "counting" $0, 1, -1, 2, -2, 3, -3, \ldots$
4) the set of pairs of natural numbers $\mathbb{N} \times \mathbb{N}$ is countable:

$$
\begin{array}{lllll}
(4,1) & & & & \\
(3,1), & (3,2), & (3,3) & & \\
(2,1), & (2,2), & (2,3), \ldots & & \\
(1,1), & (1,2), & (1,3), & (1,4), & (1,5), \ldots
\end{array}
$$

Possible "valid counting": $\underline{(1,1)}$, $\underline{(1,2), (2,1)}$, $\underline{(1,3), (2,2), (3,1)}$, $\underline{(1,4)}$

We can also show that $\mathbb{N} \times \mathbb{N}$ is countable using the alternative definition of countability:

Consider the function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

$$(a, b) \to 3^a 5^b$$

Then f is injective.

5) the set $\mathbb{Q}$, of natural numbers, is countable:
for example, we could imagine $\mathbb{Q}$ as lying inside $\mathbb{N} \times \mathbb{N}$, by sending a/b to (a, b).

$0, \frac{1}{1}, -\frac{1}{1}, \frac{1}{2}, -\frac{1}{2}, \frac{2}{1}, \ldots \frac{1}{3}, \cancel{\frac{2}{2}}, \frac{3}{1},$ ignore "repeats".

—/—

Let's now try to show that the set $\mathbb{R}$, of real numbers is not countable.

We'll will use "Cantor's diagonal arguement".

Lets suppose that there exists a list of all real between 0 and 1 (i.e. that we can count $\mathbb{R}$) between 0 and 1.

Lets produce a real number between 0 and 1 which cannot ~~cannot~~ be on the list. (so the list cannot include all real numbers between 0 and 1)

e.g
0.1 3 5 2 4 ...
0. 2 5 6 0 0 ...
0. 1 7 8 9 2 ...
0. 1 2 3 4 5 ...
⋮

Define the number $s = 0, s_1, s_2, s_3, s_4, \dots$

using the rule; $s_i = 5$ if the i^{th} decimal place number in the list is not equal to 5.

$s_i = 2$ if . . . . is equal to 5.

e.g. in this example; $s = 0.5255\dots$

Then, the real numbers satisfies $0 \leq s \leq 1$ and it disagree with i^{th} number in the list at the i^{th} decimal place.

—/—

<u>Overview of the course:</u>
1) Language.
2) Propositional logic ← 2 Questions
3) (First order) predicate logic
4) Computability.

We will try to form a basic language dealing with many kinds of mathematical structures:

If $x^2 = 1$ then $x = \pm 1$ "$\vee$" signifies "or"

$x^2 = 1 \Rightarrow x = \pm 1$

$x \cdot x = 1 \Rightarrow x = +1$ or $x = -1$

$x \cdot x = 1 \Rightarrow x = +1 \vee x = -1$

Our language will contain "unknown" or "varibles" like x, y, a, b, c and special "connecting symbol" like $\Rightarrow$, $\vee$ and also operations like multiplication

We will then study a general version of logic (propositional logic) dealing on a "large scale"

e.g. if we "know" A
and we "know" $A \Rightarrow B$
then we "know" B

Monday is a day
All days are sunny
So Monday is sunny.

We will use truth tables to understand / analyse some of the logical sysmbols / ideas:
where "0" will denote "flase".
"1" will denote "ture".

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

"Something flase imples anything"
"Something ture is implied by anything".

10/10/12

Conclusion of general overview:

Lets consider the following statement:

If we have A, and we have $A \Rightarrow B$, then we have B.

Is the implication "true"? Yes.
Is B true? This depends on whether A is true, and whether or not $A \Rightarrow B$,

Syntactic aspects of logic: related to the structure of mathematical statements, arguements and proofs

Semantic aspects of logic: related to whether or not statement are true or not.

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

We will first describe a language that allows us to express statements such as

$$x^2 = 1 \Rightarrow x = 1 \vee x = -1$$

What kind of symbols / ideas appear here?

- variables e.g x
- relations such as equality
- functions, e.g the "squaring function".
- connectives such as $\Rightarrow$, $\vee$, $\wedge$, $\forall$.

Actually, in our language, we do not "need" $\wedge$ (and), $\vee$ (or)

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

A	B	$A \vee B$
0	0	0
1	0	1
0	1	1
1	1	1

A	B	$A \wedge B$
0	0	0
1	0	0
0	1	0
1	1	1

$\neg$: "not"

A	B	$\neg A$	$(\neg A) \Rightarrow B$
0	0	1	0
1	0	0	1
0	1	1	1
1	1	0	1

So, in a sense, $\neg A \Rightarrow B$ is the "same" as $A \vee B$.

A	B	$\neg B$	$A \Rightarrow \neg B$	$\neg(A \Rightarrow \neg B)$
0	0	1	1	0
1	0	1	1	0
0	1	0	1	0
1	1	0	0	1

So, in a sense, $\neg(A \Rightarrow \neg B)$ is the "same" as $A \wedge B$.

—/—

$x^2 = 1 \Rightarrow x = 1$ or $x = -1$

↖ not clear.

—/—

start of the examinable part

Chapter 1 : Language

In order to be able to study the stucture of mathematical objects and ideas, we will first define a language in which such "things" can be defined and analysed.

The symbols we will use to construct our language consist of:

- A countable infinite set of variable symbols, e.g $\{x_1, x_2, x_3, \ldots\}$ or $\{x, y, z, x', y', z', \ldots\}$
- For each nonnegative integer n, a countably infinite set of predicate symbol e.g $\{P_1, P_2, P_3, \ldots\}$ or $\{P, Q, R, P', Q', R', \ldots\}$, each of which has arity n.

If N is a predicate symbol of arity n, we will say that N is a n-ary predicate symbol.

- The symbols $\neg$, $\Rightarrow$, $\forall$

The set of all strings of symbols from our language is denoted by $\mathcal{L}_{string}$.

Lets now define the subset of $\mathcal{L}_{string}$ consisting of "useful" strings that may be given meaning (later).

Definition: The set of formulae in the first order predicate language denoted by $\mathcal{L}$, and is the subset of $\mathcal{L}_{string}$ defined inductively as follows:

1) If P is an n-ary predicate symbol and variables $x_1, \ldots, x_n$, then $Px_1 \ldots x_n$ is a formula.
2) If α is a formula, then so is $\neg\alpha$
3) If α, β are formulae, then $\Rightarrow \alpha\beta$ is a formula.

4) If α is a formula and x is a variable symbol, then $\forall x \alpha$ is a formula.

<u>Examples</u>:
The following are all in $\mathcal{L}_{string}$ (for a 1-ary predicate P, a 2-ary predicate Q, and variables x, y, z, x_1, x_2, x_3):
x, Px, Pxy, $Qx_1x_2x_3$, Qx, Qxy, Qxx, $\forall x Px$, $\forall \neg x Px$, $P \Rightarrow Q$, $\Rightarrow Px\,Qxy$, $\neg Px$, $\neg \forall \Rightarrow$.

Of these:

Px, $\neg Px$, Qxy, Qxx, $\forall x Px$, $\Rightarrow Px\,Qxy$ are formulae

x, Pxy, $Qx_1x_2x_3$, $\neg\forall\Rightarrow$, $\forall \neg x Px$, $P \Rightarrow Q$, Qx not formulae.

<u>Notes</u>:
1) We will often refer to 1-ary and 2-ary predicates as unary and binary predicates, respectively.

2) Formulae, particularly "simple" one of the form $Px_1, \ldots, x_n$ (for an n-ary predicate P, and variables $x_1, \ldots, x_n$) are often referred to propositional functions.

3) Our language is known as the <u>first order</u> predicate language, because it can deal with statements such as

"For every subset of real numbers, ..."

This still allows us to express lots of mathematical ideas/systems using our language, but it is a deficiency, as we shall indicate towards the end of chapter 3.

—/—

Let's now define the notion of "degree" in our language:

Definition: Let α be a formula (i.e let $\alpha \in \mathcal{L}$). Then, the degree of α, denoted by $deg(\alpha)$ is a nonnegative integer obtained by (starting from 0 and) adding

- 1 for each occurrence of a $\neg$ symbol in α
- 2 ———— $\Rightarrow$ ————
- 1 ———— $\forall$ ————

Examples: Let P be a unary predicate, and x, y be variable symbols

: — Q — binary ————

————

Then $deg(Px) = 0$

$deg(\neg Px) = 1$

$deg(\forall x Qxy) = 1$

$deg(\Rightarrow Px Qxy) = 2$

$deg(\Rightarrow \neg Px \forall x Qxy) = 4$

Note: The degree counts the number of "substructures" in a formula. It "measures" the "complexity" of a formula.

The degree is particulary useful when proving results about the whole of $\mathcal{L}$; it provides a structure / order to the set of formulae.

Note: the only formulae of degree 0 are ones of the form $Px_1 \dots x_n$ for an n-ary predicate P and variables $x_1, \dots, x_n$.

There are two notable absentees from our language, which usually help remove ambiguity from mathematical statements, and "inform us of the order in which things are preformed"

These are the left and right brackets symbols, "(" and ")"

e.g. in "usual" mathematical notation, the statement $\alpha \Rightarrow \beta \Rightarrow \gamma$ is ambigous. this could mean

$$(\alpha \Rightarrow \beta) \Rightarrow \gamma$$
$$\alpha \Rightarrow (\beta \Rightarrow \gamma)$$

But, in our language there is no ambiguity, because:

(*) $(\alpha \Rightarrow \beta) \Rightarrow \gamma$ is written as $\Rightarrow \Rightarrow \alpha \beta \gamma$ in $\mathcal{L}$,

(*) $\alpha \Rightarrow (\beta \Rightarrow \gamma)$ ——— $\Rightarrow \alpha \Rightarrow \beta \gamma$ in $\mathcal{L}$.

different in $\mathcal{L}$

(*)

$(\alpha \Rightarrow \beta) \Rightarrow \gamma$
$\Rightarrow (\alpha \Rightarrow \beta) \gamma$
$\Rightarrow \Rightarrow \alpha \beta \gamma$

(*)

$\alpha \Rightarrow (\beta \Rightarrow \gamma)$
$\Rightarrow \alpha (\beta \Rightarrow \gamma)$
$\Rightarrow \alpha \Rightarrow \beta \gamma$

In general, there is no need to use brackets in $\mathcal{L}$; to show this, we will use the notion of "weight".

<u>Definition</u>: The weight of a string α (i.e of $\alpha \in \mathcal{L}_{string}$), denoted by weight($\alpha$), is an integer obtained by (starting from 0) and adding:

- -1 for each occurrence of a variable symbol in α
- $n-1$ ——— an n-ary predicate symbol ———
- 0 ——— a "$\neg$" symbol ———
- 1 ——— a "$\Rightarrow$" symbol ———
- 1 ——— a "$\forall$" symbol ———

For example: for P a unary pred, Q a binary pred, x, y variables.

Weight$(x) = -1$
Weight$(Px) = -1$ -f
Weight$(\overset{+1-1+0-1}{\forall x P x}) = -1$ -f
Weight$(\neg Qxy) = -1$ -f
Weight$(\Rightarrow Px\, Qxy) = -1$ -f

Weight$(xP) = -1$
Weight$(\forall x) = 0$
Weight$(\Rightarrow \neg Px\, \forall x\, Qxy) = -1$ -f
Weight$(\Rightarrow \forall \neg x) = 1$

-f = is a formula.

Note that every formula in the above list has weight -1.,
(However, not every string of weight -1 is a formula, e.g. weight$(\underbrace{\forall \neg xy \Rightarrow x}_{\text{not formula}}) = -1$)

<u>Proposition</u>: If α is a formula (i.e. if $\alpha \in \mathcal{L}$), then weight$(\alpha) = -1$.

<u>Proof</u>: Let's prove this by induction on degree α.
Suppose that $\deg(\alpha) = 0$. Then, α is of the form $Px_1 \ldots x_n$. for an n-ary predicate P and

variables $x_1, \dots, x_n$.

In this case $\text{weight}(\alpha) = \text{weight}(P x_1 \dots x_n) = (n-1) - n$
$= -1$

So, the result holds for formulae of degree 0.

Let us now assume that the result holds for all formulae of degree smaller than or equal to n (i.e. that every formula of degree smaller than or equal to n has weight -1).

Suppose that α is a formula such that $\deg(\alpha) = n+1$
Then, by definition of $\mathcal{L}$, α must have one of the following forms:

1) α is of the form $\neg\alpha_1$, for some formula α_1.
By considering degrees: $\deg(\alpha) \overset{=n+1}{=} \deg(\neg\alpha_1) = 1 + \deg(\alpha_1)$.
So $\deg(\alpha_1) = n$.

So, using the inductive hypothesis, we deduce that
$\text{weight}(\alpha_1) = -1$

Then $\text{weight}(\alpha) = \text{weight}(\neg\alpha_1) = \text{weight}(\neg) + \text{weight}(\alpha_1)$
$= 0 + \text{weight}(\alpha_1)$
$= -1$

2) α is of the form $\Rightarrow \alpha_1 \alpha_2$ for $\alpha_1, \alpha_2 \in \mathcal{L}$.
By considering degrees: $\deg(\alpha) \overset{=n+1}{=} \deg(\Rightarrow \alpha_1 \alpha_2)$
$= 2 + \deg(\alpha_1) + \deg(\alpha_2)$.

CANNOT WRITE
$\deg(\Rightarrow) + \deg(\alpha_1) + \deg(\alpha_2)$
NO!

So $\deg(\alpha_1) + \deg(\alpha_2) = n - 1$ i.e $\deg(\alpha_1) < n - 1$
i.e $\deg(\alpha_1) < n$, $\deg(\alpha_2) < n$.

Therefore, inductivity, we may assume that $\text{weight}(\alpha_1) = -1$
$\text{weight}(\alpha_2) = -1$.

Then $\text{weight}(\alpha) = \text{weight}(\Rightarrow \alpha_1, \alpha_2) = \text{weight}(\Rightarrow) + \text{weight}(\alpha_1) + \text{weight}(\alpha_2)$
$= 1 - 1 - 1$
$= -1$

i.e $\text{weight}(\alpha) = -1$.

3) α is of the form $\forall x \alpha_1$ for some variable x and formula α_1.

In this case: $\deg(\alpha) \overset{= n+1}{=} \deg(\forall x \alpha_1) = 1 + \deg(\alpha_1)$

So $\deg(\alpha_1) = n$, and, inductively, we may assume that $\text{weight}(\alpha_1) = -1$.

Then $\text{weight}(\alpha) = \text{weight}(\forall x \alpha_1) = 1 + (-1) + \text{weight}(\alpha_1)$
$= \text{weight}(\alpha_1)$
$= -1$

In each case, we have shown that $\text{weight}(\alpha) = -1$. This concludes the proof

$\square$

—/—

Let's now prove a related result:

Proposition: Let α be a formula, that is the concatenation $\beta\gamma$, of two (non empty) strings β and γ. (i.e α may be obtained by writing β followed, on the right, by γ)
Then: weight$(\beta) \geq 0$

In particular, no proper initial segment of α formula is a formula.

Proof: Let's prove this by induction on the degree of α.

Suppose that $\deg(\alpha) = 0$. Then, α is of the from $Px_1 \dots x_n$ for an n-ary predicate P and variables $x_1, \dots, x_n$.

Then β must be of the form $Px_1 \dots x_m$ (for $m < n$).
$n - m > 0$.

Then, weight$(\beta) = (n-1) - m = (n-m) - 1 \geq 0$, as required

—/—

$\Rightarrow \underline{P} \, x \, Q \, x \, y$

$1 + 0 - 1 + 1 - 1 - 1$

$\Sigma_i \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad -1$

—/—

Let us now assume that the result holds for all formulae of degree smaller than or equal to n.
Consider $\alpha \in L$ st $\deg(\alpha) = n+1$

then by definition of $\mathcal{L}$, α must have one of the following forms:

1) $\neg\alpha_1$ for $\alpha_1 \in \mathcal{L}$ Then by considering degrees, as in the earlier proof: $deg(\alpha_1) = n$.

So, inductively, we may assume that, if ε is an proper initial segment of α_1, then $weight(\varepsilon) \geq 0$.

In this case, the proper initial segment β (of α) must have one of the following forms.

1) $\beta : \neg$ $\quad weight(\beta) = weight(\neg) = 0$.
2) $\beta : \neg\varepsilon$ for ε a proper initial segment of α_1,

$$weight(\beta) = weight(\neg) + weight(\varepsilon) = 0 + weight(\varepsilon) \geq 0$$

by assumption.

2) $\Rightarrow\alpha_1\alpha_2$: $\alpha_1, \alpha_2 \in \mathcal{L}$
Then ... $deg(\alpha_1) + deg(\alpha_2) = n-1$, i.e $deg(\alpha_1) < n$, $deg(\alpha_2) < n$.

So, we may assume that the result holds for α_1, α_2.

We have the following possibilities for β:

1) $\beta : \Rightarrow$ $\quad weight(\beta) = 1 \geq 0$

2) $\beta : \Rightarrow\varepsilon$ for ε a proper initial segment of α_1:
$weight(\beta) = weight(\Rightarrow) + weight(\varepsilon) = 1 + weight(\varepsilon) \geq 1$ (*)

3) $\beta: \Rightarrow \alpha_1$ $\quad$ weight(β) = weight$(\Rightarrow)$ + weight(α_1)
$= 1 + (-1)$
$= 0$.

4) $\beta: \Rightarrow \alpha_1 \varepsilon$, for ε a proper initial segment of α_2

weight(β) = weight$(\Rightarrow)$ + weight(α_1) + weight(ε)
$= 1 - 1 +$ weight(ε)
$=$ weight$(\varepsilon) \geq 0$ $\quad$ $(\#)$

$(\#)$: because we may assume that weight$(\varepsilon) \geq 0$

3) $\forall x\ \alpha_1$ for $\alpha_1 \in \mathcal{L}$. Then $\deg(\alpha) \overset{=n+1}{=} 1 + \deg(\alpha_1)$
So $\deg(\alpha_1) = n$.

So we may assume the result holds for α,

Then, β must have one of the following forms:

1) $\beta: \forall$ $\quad$ weight$(\beta) = 1 \geq 0$

2) $\beta: \forall x$ weight$(\beta) = +1 - 1 = 0$.

3) $\beta: \forall x \varepsilon$ for ε a proper initial segment of α;

weight(β) = weight$(\forall)$ + weight(x) + weight(ε)
$= 1 \quad - 1 \quad +$ weight(ε)
$=$ weight$(\varepsilon) \geq 0$
by assumption

In each case, weight$(\beta) \geq 0$. This concludes the proof. $\square$

17/10/12

The last two propositions indicate that when "reading" formulae, or formula "within" formulae, no brackets are required (in $\mathcal{L}$); we may use the notion of weight in order to "pick out" formulae.

For example consider $\Rightarrow \neg P x \forall x\, Q x y$ where x, y variables.
P is a unary predicate
Q is a binary predicate

$$\begin{aligned}
\text{Weight}(\Rightarrow) &= +1 \\
\text{''}\ (\Rightarrow \neg) &= +1 + 0 = 1 \\
\text{''}\ (\Rightarrow \neg P) &= 1 \\
\text{''}\ (\Rightarrow \neg P x) &= 0 \\
\text{''}\ (\Rightarrow \neg P x \forall) &= 1 \\
\text{''}\ (\Rightarrow \neg P x\ \forall x) &= 0 \\
\text{''}\ (\Rightarrow \neg P x \forall x Q x) &= 0 \\
\text{''}\ (\Rightarrow \neg P x \forall x Q x y) &= -1 .
\end{aligned}$$

$$\Rightarrow \boxed{\neg P x}\boxed{\forall x\, Q x y}$$

Using the weight, we see that $\Rightarrow \neg P x \forall x\, Q x y$ "means" "$(\neg P x) \Rightarrow (\forall x\, Q x y)$".

that presence of this internal structure within the set of formula $\mathcal{L}$ is quite a nice feature.

However, it might be useful to write down formulae so that we use the symbols as they commonly appear in mathematics.

e.g. to write "$\alpha \Rightarrow \beta$" instead of $\Rightarrow \alpha \beta$.
or to use the symbols for "or" and "and" directly
or to write "$x = y$" instead of "$=xy$".

Furthermore, we would like to be able to describe functions in our language.

Actually, we can already do this in $\mathcal{L}$

Suppose we wish to define a function that "squares": $f(x) = x^2$. We could define a binary predicate, Q say, such that Qxy holds precisely when $x^2 = y$.

e.g. $Q81$ will not hold.
$Q82$ ——
$\vdots$
$Q863$ will not hold.
$Q864$ will hold
$Q865$ will not hold.
$\vdots$

This predicate completely "encapsulate" the "squaring" function.

But, it would be nice to be able to define functions directly.

So let's now describe a more "workable" version of the first order predicate language, which includes functions, as well as common mathematical conventions

—/—

<u>Conventional functional first order predicate language</u>
the symbol we will use to construct our language consist of:

1) A countably infinite set of variables symbols eg $\{x, y, z, x', y', z', \dots\}$.

2) For each non negative integer arity n, a countably infinite set of n-ary predicate symbols e.g $\{P, Q, R, \dots\}$.

3) For each non-negative arity n, a countable infinite set of n-ary functional symbols e.g $\{F_1, F_2, F_3, \dots\}$.

4) the symbol $\neg, \Rightarrow, \forall, (,)$.

Let's now define the corresponding set of formulae:

<u>Definition</u>: The set of formulae in the conventional functional first order predicate language is denoted by L_{math}, and is defined inductively as follows:

0) Every variable symbol is a variable.

1) If F is an n-ary functional symbol, and $x_1, \dots, x_n$

are variables, then $Fx_1 \ldots x_n$ is a variable

2) If P is an n-ary predicate symbol, and $x_1, \ldots, x_n$ are variables, then $Px_1 \ldots x_n$ is a formula.

3) If α is a formula then so is $\neg\alpha$.

4) If α, β are formula, then $\Rightarrow \alpha\beta$ is a formula.

5) If α is a formula and x is a variable symbol then $\forall x \alpha$ is a formula.

In addition, our language will include the following conventions.

- We will allow ourselves to write down:

"$\alpha \Rightarrow \beta$" instead of $\Rightarrow \alpha\beta$

"$\alpha \vee \beta$" $(\neg\alpha) \Rightarrow \beta$, or $\Rightarrow \neg\alpha\beta$ in $\mathcal{L}$.

"$\alpha \wedge \beta$" $\neg(\alpha \Rightarrow (\neg\beta))$, or $\neg \Rightarrow \alpha \neg\beta$ in $\mathcal{L}$.

"$\alpha \Leftrightarrow \beta$" instead of $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

just workings $\left\{ \begin{array}{l} \neg((\alpha \Rightarrow \beta) \Rightarrow \neg(\beta \Rightarrow \alpha)) \\ \neg \Rightarrow (\alpha \Rightarrow \beta) \neg(\beta \Rightarrow \alpha) \end{array} \right.$

or $\boxed{\neg \Rightarrow \Rightarrow \alpha\beta \neg \Rightarrow \beta\alpha}$ in $\mathcal{L}$.

$\boxed{\exists x \alpha \text{ denotes } \neg\forall x \neg\alpha}$ ← Was missing

- For a binary predicate or functional, P say, we will allow ourselves to write down xPy instead of Pxy

eg $x = y$ ⟶ $= xy$

eg $x + y$ ⟶ $+ xy$.

- For an n-ary predicate or functional P, we may write $P(x_1, \dots, x_n)$ instead of $Px_1 \dots x_n$.

Notes:

1) When describing specific mathematical structures using our language, we will often use Π to denote the set of predicates that appear and Ω to denote the set of functionals that appear.

2) A predicte gives rise to a mathematical statement, that we may describe as ture or false in a specific setting. e.g. "$2=6$" → ture modulo 2.
→ flase $\mathbb{Z}$.

A functoinal gives rise to a specific answer, after being given some inputs.
This answer may change depending on the inputs provided; it is a "variable" in some sense.

3) A functoinal of arity 0 takes in no inputs, and gives out an answer. So, the answer is independant of the input values, it is a constant. This will be useful in specific mathematical systems, when we wish to introduce distinguished constants. (eg. we will use a 0-ary functoinal to describe the identity in a group).

Let's now see some examples of specific mathematical system/objects written in the language(s) of chapter1

Examples:

1) Poset

A poset, or partially orded set, is a set X, together with two relations, "=" and "≤" satisfying the following conditions.

1) For every x in X: $x \leq x$.
2) For all x, y in X; if $x \leq y$ and $y \leq x$, then $x = y$.
3) For all x, y, z in X; if $x \leq y$ and $y \leq z$ then $x \leq z$.

e.g if "=" denotes "equals" and "≤" denotes "less than or equal to". then the sets of natural numbers, integers, real numbers form posets, if "=" and "≤" denotes "is the subset of" then the set of subsets of $\mathbb{N}$. forms a poset.

—⟵

Let's write the defining statement of posets:

- We use the predicates "=" and "≤" so $\Pi = \{=, \leq\}$ arity 2 arity 2
- We use no functional, so $\Omega = \emptyset$.

In $\mathcal{L}$math:

1) $(\forall x)(x \leq x)$
2) $(\forall x)(\forall y)(((x \leq y) \wedge (y \leq x)) \Rightarrow (x = y))$
3) $(\forall x)(\forall y)(\forall z)(((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z))$

In $\mathcal{L}$:

1) $\forall x \leq xx$.
2) $\forall x \forall y \Rightarrow \neg \Rightarrow \leq xy \neg \leq xy = xy$.
3) Good luck!

— / —

Examples:

2) Groups

A group consists of a set G, together with an operation $\cdot$, such that:

1) For all x, y in G, xy is in G.
2) There exists an element e of G such that $e \cdot x = x = x \cdot e$ for all x in G.
3) For each x in G, there exists an element y in G such that $x \cdot y = e = y \cdot x$.
4) For each x, y, z in G: $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

Let's try to write the defining statements of groups in $\mathcal{L}_{math}$:

We will use the predicate "$=$" (arity 2) so $\Pi = \{=\}$.
functional M (arity 2), E (arity 0).
so $\Omega = \{M, E\}$.

$\boxed{\exists x \alpha \text{ denotes } \neg \forall x \neg \alpha}$

Then, in $\mathcal{L}_{math}$:

1) ______ (assumed).
2) $(\forall x)((M(E,x) = x) \wedge (M(x, E) = x))$
3) $(\forall x)(\exists y)((M(x,y) = E) \wedge (M(y,x) = E))$
4) $(\forall x)(\forall y)(\forall z)((M(M(x,y), z)) = (M(x, M(y,z))))$

The last two examples indicate that we can use $\mathcal{L}_{math}$ to conveniently express statements related to mathematical objects, e.g. posets, groups, rings, fields etc.

But, next, we will define a simpler version of our language and study the notions of truth and provability using that version (as well as the interplay between these ideas).

Chapter 2: Propositional Logic

In this chapter, we will work with a reduced version of the language(s) from chapter 1. We will "ignore" variables, and so will not (need to) use the "$\forall$" or "$\exists$" symbol.

Also, no functionals will be present, and no predicates of arity greater than zero.

The symbols we will use consist of:

- a countable infinite set of "primitive propositions", denoted by $\mathcal{L}_0^P$.
- the symbol $\neg$, $\Rightarrow$, $($, $)$

Using these symbols, we can obtain all propositions:

Definition: The set of propositions, denoted by $\mathcal{L}_0$, is defined inductively as follows:

1) Every primitive proposition is a proposition, i.e if $\alpha \in \mathcal{L}_0^P$, then $\alpha \in \mathcal{L}_0$.

2) If α is a proposition, then so is $(\neg\alpha)$.

3) If α, β are propositions, then so is $(\alpha \Rightarrow \beta)$

Notes:

1) The notion of degree (from chapter 1) carries over to the setting of propositions. Then, the propositions of degree 0 are precisely the primitive propositions.

2) The set of proposition of degree 0 is countable.
As a result, the set of ———— degree 1 is countable
the set of ———— degree 2 is countable.
and so on.

So, the set of all propositions is countable.

Semantic aspects of propositional logic.

Let's now study the notion of truth, in L_0. We will assign truth/flasehood to the set of all propositions by first choosing the truth/falsehood of primitive propositions and then determining the truth of more complicated propositions using "sensible" rules.

Definition: A valuation v is a function $v: L \to \{0,1\}$ satisfying the following conditions:

1) For a propositional α: $v(\neg\alpha) = 1$ if $v(\alpha) = 0$ and $v(\neg\alpha) = 0$ if $v(\alpha) = 1$ } i.e $v(\neg\alpha) = 1 - v(\alpha)$

2) For proposition α, β: $v(\alpha \Rightarrow \beta) = 0$ if $v(\alpha) = 1$ and $v(\beta) = 0$.

$v(\alpha \Rightarrow \beta) = 1$ otherwise.

Note: We will use "0" to denote the falsehood of a statement and "1" ——— truth ———

Let's show that a valuation is determined by its values on the primitive positions.

<u>Propositions</u>: Let $v: \mathcal{L}_0 \to \{0,1\}$ and $v': \mathcal{L} \to \{0,1\}$. be two valuations such that

$$v(\alpha) = v'(\alpha) \quad \text{for any } \alpha \in \mathcal{L}_0^P.$$

Then, for any proposition (i.e. for any $\alpha \in \mathcal{L}_0$): $v(\alpha) = v'(\alpha)$.

<u>Proof</u>: We prove this by induction on the degree of α.

Suppose that $\deg(\alpha) = 0$. Then α is a primitive proposition.

So $v(\alpha) = v'(\alpha)$ by assumptions

Let's assume that the result holds for all posposition of degree smaller than or equal to n.

Suppose that $\alpha \in \mathcal{L}_0$, and $\deg(\alpha) = n+1$. Then, α must have one of the following forms:

• $\alpha = \neg\alpha_1$ for some proposition α_1. Then
$\deg(\alpha)^{=n+1} = \deg(\neg\alpha_1) = 1 + \deg(\alpha_1)$, so $\deg(\alpha_1) = n$.

So inductively we may assume that $v(\alpha_1) = v'(\alpha_1)$.

Then: $\left.\begin{array}{l} v(\alpha) = v(\neg\alpha_1) = 1 - v(\alpha_1) \\ v'(\alpha) = v'(\neg\alpha_1) = 1 - v'(\alpha_1) \end{array}\right\}$ since $v(\alpha_1) = v'(\alpha_1)$ we obtain $v(\alpha) = v'(\alpha)$, as required.

• $\alpha = \alpha_1 \Rightarrow \alpha_2$ for some propositions α_1, α_2.

Then $\deg(\alpha)^{=n+1} = 2 + \deg(\alpha_1) + \deg(\alpha_2)$, so
$\deg(\alpha_1) < n$ and $\deg(\alpha_2) < n$.

So inductively, we assume that $v(\alpha_1) = v'(\alpha_1)$ and $v(\alpha_2) = v'(\alpha_2)$.

But then,

$$v(\alpha) = v(\alpha_1 \Rightarrow \alpha_2) = \begin{cases} 0 & \text{if } v(\alpha_1)=1 \text{ and } v(\alpha_2)=0 \\ 1 & \text{otherwise} \end{cases}$$

$$v'(\alpha) = v'(\alpha_1 \Rightarrow \alpha_2) = \begin{cases} 0 & \text{if } v'(\alpha_1)=1 \text{ and } v'(\alpha_2)=0 \\ 1 & \text{otherwise} \end{cases}$$

Since $v(\alpha_1) = v'(\alpha_1)$ and $v(\alpha_2) = v'(\alpha_2)$, we deduce that $v(\alpha) = v'(\alpha)$
This concludes the proof

$\square$

<u>22/10/12</u>

Last time: we defined a valuation, and showed that. If, for valuations v, v': $v(\alpha) = v'(\alpha)$ for all $\alpha \in \mathcal{L}_0^P$ then $v(\alpha) = v'(\alpha)$ for all α ($\in \mathcal{L}_0$)

i.e "if two valuations agree on the primitive propositions then they agree on all propositions".

—/—

Let's now give a related result:

<u>Proposition</u>: Consider a function $f: \mathcal{L}_0^P \to \{0,1\}$. Then, there exists a valuation $v: \mathcal{L}_0 \to \{0,1\}$ such that $v(\alpha) = f(\alpha)$ for all $\alpha \in \mathcal{L}_0^P$.

<u>Proof</u>: By induction on the degree of a proposition. Let's try to construct the valuation v by defining it on any proposition α.

If $\deg(\alpha) = 0$, then α is a primitive proposition. In this case, set $v(\alpha) = f(\alpha)$, as required.

Suppose, now, that the result holds for all propositions of degree smaller than or equal to n, i.e that we have successfully defined the valuation v on such propositions.

Consider a proposition α, of degree $n+1$.

Then, α must have one of the following forms:

- $\alpha = \neg\alpha_1$ for some $\alpha_1 \in \mathcal{L}_0$.

Then since $\deg(\alpha) = n+1$; $\deg(\alpha_1) = n$, so we may inductively assume that $v(\alpha_1)$ is defined. Now, set $v(\alpha) = 1 - v(\alpha_1)$.

- $\alpha = \alpha_1 \Rightarrow \alpha_2$ for some $\alpha_1, \alpha_2 \in \mathcal{L}_0$.

Then, $\deg(\alpha_1) < n$ and $\deg(\alpha_2) < n$, so we may inductively assume that $v(\alpha_1)$, $v(\alpha_2)$ have been already defined.

Set $v(\alpha) = \begin{cases} 0 & \text{if } v(\alpha_1) = 1 \text{ and } v(\alpha_2) = 0 \\ 1 & \text{otherwise.} \end{cases}$

This concludes the proof ; by construction, v is a valuation such that $v(\alpha) = f(\alpha)$ for all $\alpha \in \mathcal{L}_0^P$.

So, the last two results have shown that "a valuation is defined by its values on the primitive propositions, and any values will do".

We now give some of the terminology related to valuations.

Let $\alpha \in \mathcal{L}_0$: If, for some valuation v, $v(\alpha) = 1$, then we say that

α is true in v

or v is a model of α

If S is a set of propositions ($S \subset \mathcal{L}_0$) and v is a valuation such that $v(\alpha) = 1$ for all $\alpha \in S$, then we say that :

v is a model of S.

If α is a proposition, such that, for any possible valuation v, $v(\alpha) = 1$ then we say that α is a tautology.

—/—

Often, if we wish to check whether or not a proposition is a tautology, or to simply check precisely for which kinds of valuations it is true, we write down a table that evaluates the proposition under all possible valuations. This is a truth tables.

When writing down a truth table, we may assume the following rules

- $v(\neg\alpha) = 1$ if $v(\alpha) = 0$, and $v(\neg\alpha) = 0$ if $v(\alpha) = 1$
- $v(\alpha \Rightarrow \beta) = \begin{cases} 0 & \text{if } v(\alpha) = 1 \text{ and } v(\beta) = 0 \\ 1 & \text{otherwise.} \end{cases}$
- $v(\alpha \vee \beta) = 0$ if $v(\alpha) = 0$ and $v(\beta) = 0$, $v(\alpha \vee \beta) = 1$ otherwise.
- $v(\alpha \wedge \beta) = 1$ if $v(\alpha) = 1$ and $v(\beta) = 1$, $v(\alpha \wedge \beta) = 0$ otherwise.

If two propositions α, β have "identical truth table columns", then they are true for precisely the same valuations.

In this case, i.e if, for any valuation $v : L_0 \to \{0, 1\}$ we have $v(\alpha) = v(\beta)$ then we say that α and β are semantically equivalent.

Some examples of truth tables:

- $\alpha \Rightarrow \beta$

α	β	$\alpha \Rightarrow \beta$
0	0	1
1	0	0
0	1	1
1	1	1

- $\alpha \Rightarrow \alpha$

α	$\alpha \Rightarrow \alpha$
0	1
1	1

truth table column of $\alpha \Rightarrow \alpha$ contains only ones.

So, $\alpha \Rightarrow \alpha$ is a tautology.

- $\neg\neg\alpha$

α	$\neg\alpha$	$\neg\neg\alpha$
0	1	0
1	0	1

α and $\neg\neg\alpha$ have identical truth table columns.

So, α and $\neg\neg\alpha$ are semantically equivalent

$(\neg\alpha) \Rightarrow \beta$

α	β	$\neg\alpha$	$(\neg\alpha)\Rightarrow\beta$
0	0	1	0
1	0	0	1
0	1	1	1
1	1	0	1

$(\neg\beta) \Rightarrow \alpha$

α	β	$\neg\beta$	$(\neg\beta)\Rightarrow\alpha$
0	0	1	0
1	0	1	1
0	1	0	1
1	1	0	1

So $(\neg\alpha)\Rightarrow\beta$ and $(\neg\beta)\Rightarrow\alpha$ are semantically equivalent.

We might have expected this, because of our convention for "$\vee$":
"$\alpha \vee \beta$" is expressed as $(\neg\alpha)\Rightarrow\beta$
"$\beta \vee \alpha$" is expressed as $(\neg\beta)\Rightarrow\alpha$
and "$\alpha\vee\beta$", "$\beta\vee\alpha$" mean the same.

· $(\neg\neg\alpha) \Rightarrow \alpha$

α	$\neg\alpha$	$\neg\neg\alpha$	$(\neg\neg\alpha)\Rightarrow\alpha$
0	1	0	1
1	0	1	1

So $(\neg\neg\alpha)\Rightarrow\alpha$ is a tautolgy.

· $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

α	β	$\beta\Rightarrow\alpha$	$\alpha\Rightarrow(\beta\Rightarrow\alpha)$
0	0	1	1
1	0	1	1
0	1	0	1
1	1	1	1

So $\alpha\Rightarrow(\beta\Rightarrow\alpha)$ is a tautolgy

· $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$

α	β	γ	$\beta \Rightarrow \gamma$	$\alpha \Rightarrow (\beta \Rightarrow \gamma)$	$(\alpha \Rightarrow \beta)$	$(\alpha \Rightarrow \gamma)$	$(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$	$(\alpha \Rightarrow (\beta \Rightarrow \gamma))$ $\Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$
0	0	0	1	1	1	1	1	1
1	0	0	1	1	0	0	1	1
0	1	0	0	1	1	1	1	1
1	1	0	0	0	1	0	0	1
0	0	1	1	1	1	1	1	1
1	0	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

So, $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ is a tautology.

In fact, $\alpha \Rightarrow (\beta \Rightarrow \gamma)$ and $(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$ are semantically equivalent.

· $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$:

α	β	$\alpha \Rightarrow \beta$	$\beta \Rightarrow \alpha$	$(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$
0	0	1	1	1
1	0	0	1	1
0	1	1	0	0
1	1	1	1	1

The given proposition is not a tautology.

In fact, for any valuation v such that $v(\alpha) = 0$ and $v(\beta) = 1$, then $v((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)) = 0$.

—/—

Let's see some other ways of proving that $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$ is not a tautology.

Claim: The proposition $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$ is not a tautology.

Proof 1: Let v be a valuation such that $v(\alpha) = 0$ and $v(\beta) = 1$. Then $v(\alpha \Rightarrow \beta) = 1$, $v(\beta \Rightarrow \alpha) = 0$, so $v((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)) = 0$. The presence of this indicates that $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$ is not a tautology.

Proof 2: Suppose that, for some valuation v, $v((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)) = 0$. Then $v(\alpha \Rightarrow \beta) = 1$ and $v(\beta \Rightarrow \alpha) = 0$.

Then, since $v(\beta \Rightarrow \alpha) = 0$, it must be the case that $v(\alpha) = 0$ and $v(\beta) = 1$.

We may then verify that if $v(\alpha) = 0$ and $v(\beta) = 1$, then $v(\alpha \Rightarrow \beta) = 1$ and $v(\beta \Rightarrow \alpha) = 0$.

So that $v((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)) = 0$, as required. Hence $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$ is not a tautology

Let's also see how we might have proven that $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ is a tautology, using a similar method.

Claim: $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ is a tautology.

Proof: Let's try to show this contradiction.

Suppose that, for some valuation v, $v(\alpha \Rightarrow (\beta \Rightarrow \alpha)) = 0$.

Then $v(\alpha) = 1$ and $v(\beta \Rightarrow \alpha) = 0$.

In such a case, since $v(\beta \Rightarrow \alpha) = 0$, it must be the case that $v(\beta) = 1$ and $v(\alpha) = 0$.

So, overall $v(\alpha) = 1$ and $v(\alpha) = 0$. This is not possible, it contradicts the definition of a valuation.

So, as required, $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ is a tautology. (there is no valuation v such that $v(\alpha \Rightarrow (\beta \Rightarrow \alpha)) = 0$)

We will now see a method that "diagrammatically imitates" the arguement used in the previous two proofs, where we try to check if a proposition can ever be false by breaking it down into simpler and simpler parts.

This method is known as the semantic tableaux method.

Let's see some of the "simple diagrams" that we may use as building blocks which we may read as:

$$\begin{array}{c} \alpha \vee \beta \\ / \quad \backslash \\ \alpha \qquad \beta \end{array}$$

" $\alpha \vee \beta$ is true for any valuation for which α is true (one branch)

or for any valuation for which β is true (the other branch)".

Similarly, $\alpha \wedge \beta$ is true precisely when α and β are both true, leading to:

$$\begin{array}{c} \alpha \wedge \beta \\ | \\ \alpha, \beta . \end{array}$$

Let's see a complete list of the "building blocks" we may use in the semantic tableaux method:

$$\begin{array}{c} \neg\neg\alpha \\ | \\ \alpha . \end{array}$$

$$\begin{array}{c} \alpha \vee \beta \\ / \quad \backslash \\ \alpha \qquad \beta \end{array} , \quad \begin{array}{c} \neg(\alpha \vee \beta) \\ | \\ \neg\alpha, \neg\beta \end{array} , \quad \begin{array}{c} (\alpha \wedge \beta) \\ | \\ \alpha, \beta \end{array} , \quad \begin{array}{c} \neg(\alpha \wedge \beta) \\ / \quad \backslash \\ \neg\alpha \qquad \neg\beta . \end{array}$$

$$\begin{array}{c} \alpha \Rightarrow \beta \\ / \quad \backslash \\ \neg\alpha \qquad \beta \end{array} \quad \begin{array}{c} \neg(\alpha \Rightarrow \beta) \\ | \\ \alpha, \neg\beta \end{array} \quad \begin{array}{c} \alpha \Leftrightarrow \beta \\ / \quad \backslash \\ \alpha, \beta \qquad \neg\alpha, \neg\beta \end{array} \quad \begin{array}{c} \neg(\alpha \Leftrightarrow \beta) \\ / \quad \backslash \\ \alpha, \neg\beta \qquad \neg\alpha, \beta . \end{array}$$

α	β	$\alpha \Rightarrow \beta$
0	1	1
1	1	1
0	0	1
1	0	0

24/10/11

Description of semantic tableaux method algorithm. Suppose we are given a proposition α, and we wish to determine whether or not α is a tautology (and, if it is not a tautology, we wish to determine for which kinds of valuation α fails to be true).

Then, as in the proofs from last time, we use the idea that α is a tautology precisely if $\neg\alpha$ is never true. i.e α is true for every valuation if and only if $\neg\alpha$ is not true for any valuation.

We may achieve this using a (propositional) semantic tableaux as follows:

1) Consider $\neg\alpha$.

2) Break down $\neg\alpha$ into simpler and simpler proposition using the basic rules of semantic tableaux (that we saw last time), until we are down to the "indecompose" parts of α.

3) Study each "branch" of the resulting tableaux (all the way from the bottom edge of the branch to the top of the tableaux).

- if a branch contains both δ and $\neg\delta$, for some "indecomposable" δ, then we say the branch is closed
- if a branch does not contain an "indecomposable" proposition and its negation, then we say the branch is open.

4) If every branch is closed, then α is a tautology (i.e $\neg\alpha$ can never be true). If there exists an open branch, then α is not a tautology; by studying the "undecomposable" propositions of an open branch, we can describe a valuation v st $v(\neg\alpha) = 1$, i.e st $v(\alpha) = 0$.

Basic rules of propositional semantic tableaux:

$$\neg\neg\alpha \;\to\; \alpha \qquad \alpha \vee \beta \;\to\; \alpha \;\mid\; \beta \qquad \alpha \wedge \beta \;\to\; \alpha, \beta \qquad \alpha \Rightarrow \beta \;\to\; \neg\alpha \;\mid\; \beta \qquad \alpha \Leftrightarrow \beta \;\to\; \alpha, \beta \;\mid\; \neg\alpha, \neg\beta.$$

$$\neg(\alpha \vee \beta) \;\to\; \neg\alpha, \neg\beta \qquad \neg(\alpha \wedge \beta) \;\to\; \neg\alpha \;\mid\; \neg\beta \qquad \neg(\alpha \Rightarrow \beta) \;\to\; \alpha, \neg\beta \qquad \neg(\alpha \Leftrightarrow \beta) \;\to\; \neg\alpha, \beta \;\mid\; \alpha, \neg\beta.$$

<u>Examples of semantic tableaux:</u>

- Is $\alpha \Rightarrow \alpha$ is a tautology?

Consider $\neg(\alpha \Rightarrow \alpha)$, and form a semantic tableaux for this:

$$\neg(\alpha \Rightarrow \alpha)$$
$$|$$
$$\neg\alpha, \alpha$$

We have a single closed branch, so $\alpha \Rightarrow \alpha$ is a tautology.

- $\alpha \Rightarrow \beta$: Consider a tableaux for $\neg(\alpha \Rightarrow \beta)$

$$\neg(\alpha \Rightarrow \beta)$$
$$|$$
$$\alpha, \neg\beta.$$

There is a single open branch, so $\alpha \Rightarrow \beta$ is not a tautology. In fact, $v(\alpha \Rightarrow \beta) = 0$ for any valuation v satisfying $v(\alpha) = 1$ and $v(\beta) = 0$ since the open branch contains α and $\neg\beta$.

- $\neg(\alpha \Rightarrow \beta)$: Consider the tableaux for $\neg\neg(\alpha \Rightarrow \beta)$

$$\begin{array}{ccc} & \neg\neg(\alpha \Rightarrow \beta) & \\ & | & \\ & \alpha \Rightarrow \beta & \\ / & & \backslash \\ \neg\alpha & & \beta \\ \underline{\text{open}} & & \underline{\text{open}} \end{array}$$

Since there exist open branches, $\neg(\alpha \Rightarrow \beta)$ is <u>not</u> a tautogy.

In fact: $v(\neg(\alpha \Rightarrow \beta)) = 0$ for any valuation v st $v(\alpha) = 0$ or for any valuation v st $v(\beta) = 1$.

- $\alpha \Rightarrow (\beta \Rightarrow \alpha)$: Consider $\neg(\alpha \Rightarrow (\beta \Rightarrow \alpha))$

$$\neg(\alpha \Rightarrow (\beta \Rightarrow \alpha))$$
$$|$$
$$\alpha, \neg(\beta \Rightarrow \alpha)$$
$$|$$
$$\beta, \neg\alpha$$

closed

The branch is closed, so we deduce that $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ is a tautology.

- $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$ Consider $\neg((\beta \Rightarrow \alpha) \Rightarrow (\beta \Rightarrow \alpha))$

$$\neg((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha))$$
$$|$$
$$(\alpha \Rightarrow \beta), \neg(\beta \Rightarrow \alpha)$$
$$\swarrow \quad \searrow$$
$$\neg\alpha \qquad \beta$$
$$| \qquad |$$
$$\beta, \neg\alpha \qquad \beta, \neg\alpha.$$

open open

$$\neg(\beta \Rightarrow \alpha)$$
$$|$$
$$\beta, \neg\alpha$$

So, since there are open branches, $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$ is not a tautology. $v((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)) = 0$ for any valuation v such that $v(\alpha) = 0$ and $v(\beta) = 1$.

Alternative tableax: decompose "$\neg(\beta \Rightarrow \alpha)$" before "$\alpha \Rightarrow \beta$" (final conclusion is the same) . . .

$$\neg((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha))$$

$$(\alpha \Rightarrow \beta), \neg(\beta \Rightarrow \alpha)$$

$$\beta, \neg\alpha$$

$\neg\alpha$ β

open   open

Same conclusion as earlier.

• $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$:

Consider a tableaux for $\neg((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \beta)))$

$$\neg((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)))$$

$$\alpha \Rightarrow (\beta \Rightarrow \gamma), \neg((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$$

$$(\alpha \Rightarrow \beta), \neg(\alpha \Rightarrow \gamma)$$

$$\alpha, \neg\gamma$$

$\neg\alpha$ β

$\neg\alpha$ $\beta \Rightarrow \gamma$ $\neg\alpha$ $\beta \Rightarrow \gamma$

closed $\neg\beta$ γ closed $\neg\beta$ γ

no need   closed   closed.

Every branch is closed: $((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$

is tautology.

29/10/12.

Some more examples of semantic tableaux:

- Is $((\alpha \wedge \beta) \Rightarrow \gamma) \Rightarrow ((\alpha \wedge \beta) \Rightarrow \gamma)$ a tautology?

Consider a tableaux for the negation of this proposition:

$$\neg(((\alpha \wedge \beta) \Rightarrow \gamma) \Rightarrow ((\alpha \wedge \beta) \Rightarrow \gamma)) \quad ①$$

$$④ \ (\alpha \wedge \beta) \Rightarrow \gamma \ , \ \neg((\alpha \wedge \beta) \Rightarrow \gamma) \quad ②$$

$$③ \ \alpha \wedge \beta \ , \ \neg\gamma$$

$$\alpha \qquad \beta$$

$$⑤ \ \neg(\alpha \wedge \beta) \qquad \gamma \qquad \neg(\alpha \wedge \beta) \qquad \gamma$$

closed　　closed

$$\neg\alpha \qquad \neg\beta \qquad \neg\alpha \qquad \beta$$

closed　open.　open　closed

Since there exist open branches, the original proposition is not a tautology.

In fact, $v(((\alpha \wedge \beta) \Rightarrow \gamma) \Rightarrow ((\alpha \wedge \beta) = \gamma)) = 0$ for any valuation v such that

- $v(\alpha) = 1$, $v(\beta) = 0$, $v(\gamma) = 0$.

or any valuation v such that:

$$v(\alpha) = 0, \quad v(\beta) = 1, \quad v(\gamma) = 0$$

— / —

Is $(\neg\alpha \Rightarrow \neg\beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha)$ a tautology?

Consider the following tableaux.

$$\neg((\neg\alpha \Rightarrow \neg\beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha))$$

$$(\neg\alpha \Rightarrow \neg\beta), \ \neg((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha)$$

$$\neg\alpha \Rightarrow \beta, \ \neg\alpha$$

$\neg\neg\alpha$ | α | closed

$\neg\beta$:
- $\neg\neg\alpha$ | α | closed.
- β closed

Since every branch is closed, we deduce that $(\neg\alpha \Rightarrow \neg\beta) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha)$ <u>is</u> a tautology.

Let's extend our notion of truth of include ideas such as "a proposition is true whenever some other proposition are true".

<u>Definition</u>: Let S be a set of proposition ($S \subset L_0$), and α be a proposition ($\alpha \in L_0$). Then, we say that S <u>semantically implies</u>, or <u>entails</u>, α if for any valuation v such that $v(s) = 1$ for all $s \in S$ we must also have $v(\alpha) = 1$.

If S entails α, we write $S \models \alpha$.

<u>Note</u>: If $\emptyset \models \alpha$, then α is always true. i.e. α is a tautology.

We will often simply this to: $\models \alpha$.

Eg: $\models (\alpha \Rightarrow \alpha)$
$\models (\neg\neg\alpha \Rightarrow \alpha)$

<u>Examples</u>. $\models (\neg\neg\alpha) \Rightarrow \alpha$
$\models (\alpha \Rightarrow \alpha)$
$\{\beta\} \models (\alpha \Rightarrow \beta)$ since $v(\alpha \Rightarrow \beta) = 1$ whenever $v(\beta) = 1$
$\{\neg\alpha\} \models (\alpha \Rightarrow \beta)$ since $v(\alpha \Rightarrow \beta) = 1$ whenever $v(\alpha) = 0$.

We may determine whether or not an entailment $S \models a$ holds by using semantic tableaux method.

<u>General idea</u>: $S \models \alpha$ holds

$\Updownarrow$

For any valuation v st $v(s) = 1$ $\forall s \in S$, $v(\alpha) = 1$

$\Updownarrow$

There is no valuation v s.t. $v(s) = 1$ $\forall s \in S$ and $v(\alpha) = 0$

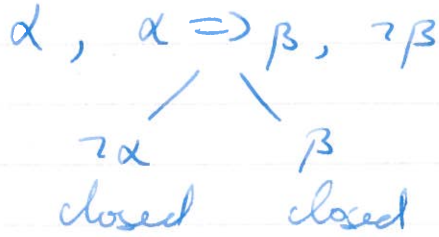
$\Updownarrow$

There is no valuation v s.t. everything in $S \cup \{\neg\alpha\}$ is true for v.

So, to check if $S \models \alpha$, we may apply the semantic tableau method (starting with $S \cup \{\neg\alpha\}$)

<u>Examples</u>: $\{\alpha, \alpha \Rightarrow \beta\} \Rightarrow \beta$?

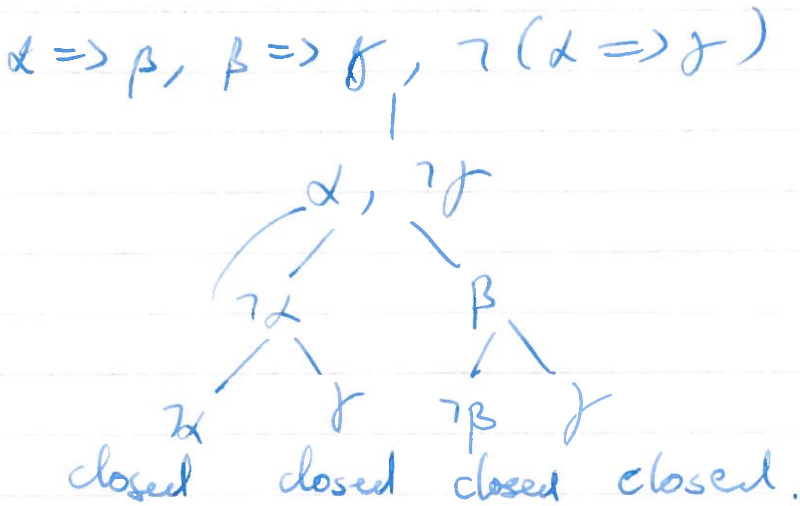
Consider the tableaux starting with $\alpha, \alpha \Rightarrow \beta, \neg\beta$.



Every branch is closed, so the semantic entailment $\{\alpha, \alpha \Rightarrow \beta\} \models \beta$ holds.

- Does $\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \models (\alpha \Rightarrow \gamma)$ hold?

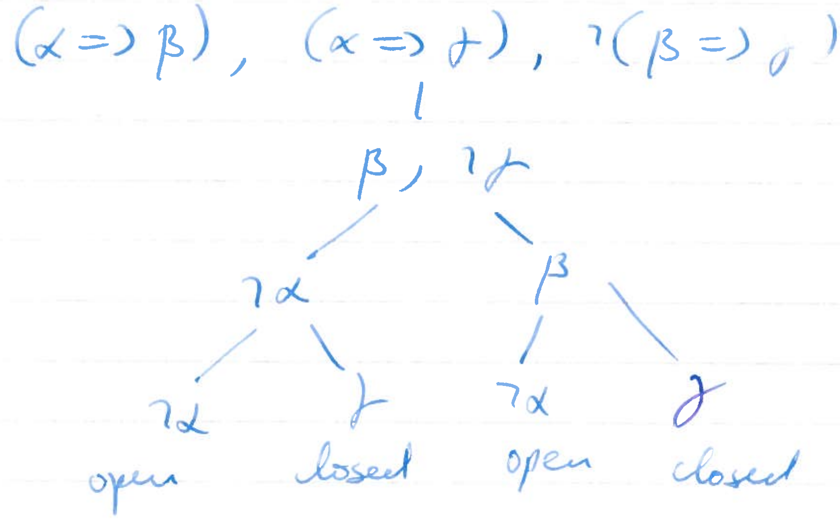
Consider the following tableaux:



Every branch is closed, so the semantic entailment holds.

· Does $\{\alpha \Rightarrow \beta, \alpha \Rightarrow \gamma\} \models (\beta \Rightarrow \gamma)$ holds?

Consider the following tableaux:



Since there exist open branches, we deduce that semantic implication does not hold.

In fact $v(\alpha \Rightarrow \beta) = 1$, $v(\alpha \Rightarrow \gamma) = 1$ but $v(\beta \Rightarrow \gamma) = 0$ for any valuation v st $v(\alpha) = 0$, $v(\beta) = 0$, $v(\gamma) = 0$.

Let's now move from "truth" to "proof".

<u>Syntactic aspect of propositional logic.</u>
We begin by seeing how we can define the notion of "proof".

To construct proofs, we will use the following <u>axioms</u>:

<u>Axioms 1</u>: $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ for any $\alpha, \beta \in L_0$
<u>Axioms 2</u>: $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ for

any $\alpha, \beta, \gamma \in \mathcal{L}_0$

Axiom 3: $((\neg\alpha) \Rightarrow \neg\beta) \Rightarrow (((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$ for any $\alpha, \beta, \gamma \in \mathcal{L}_0$.

We will also use the following rule of deduction, known as modus ponens: If we have $\alpha \Rightarrow \beta$, then we can deduce β.

Modus ponens: From α and $(\alpha \Rightarrow \beta)$, we may deduce β (for any propositions α, β).

—/—

We may then define the notion of proof

Definition: Let S be a set of propositions, and let α be a proposition.

A proof of α from S is a finite, ordered sequence of propositions, or lines $t_1, \dots, t_n$ say, such that t_n is the proposition α, and such that for each $1 \leq i \leq n$, t_i is either:

- an (occurrence of an) axioms or
- an element of S (also known as a hypothesis) or
- is deduced by modus ponens from two preceding proposition γ, and t_j is the proposition $\gamma \Rightarrow \delta$.

Notes:

1) All our axioms are instances of tautolgies: (have seen this already).
2) Each axiom given above corresponds to an infinite collection of propositions (they all

"work" for all proposition α, β, γ where relevant). They are often referred to as axiom schemes.

Let's now give the notion corresponding to "entailment" in the setting of proofs:

Definition: If $S \subset L_0$ and $\alpha \in L_0$, then we say that S syntactically implies, or proves α, and write $S \vdash \alpha$; if there exists a proof of α from S.

Note: If $\emptyset \vdash \alpha$, i.e if we can prove α without using any hypothesis, then we say that α is a theorem, and may write $\vdash \alpha$

Examples of proofs. $\{\alpha, \alpha \Rightarrow \beta\} \vdash \beta$
The following is a proof of the implication:

1. α hypothesis
2. $\alpha \Rightarrow \beta$ hypothesis
3. β modus ponens on lines 1, 2.

· Lets' write down a proof that shows $\{\neg\alpha \Rightarrow \neg\beta, \neg\alpha \Rightarrow \beta\} \vdash \alpha$

1. $(\neg\alpha) \Rightarrow (\neg\beta)$ hypothesis
2. $(\neg\alpha) \Rightarrow \beta$ hypothesis
3. $((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow (((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$ Axioms 3
4. $((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha$ modus ponens on lines 1, 3.
5. α modus ponens on lines 2, 4.

- Let's write down a proof of
$\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \vdash (\alpha \Rightarrow \gamma)$.

1. $\alpha \Rightarrow \beta$ — hypothesis
2. $\beta \Rightarrow \gamma$ — hypothesis
3. $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ — Axiom 2.
4. $(\beta \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow (\beta \Rightarrow \gamma))$ — Axiom 1
5. $\alpha \Rightarrow (\beta \Rightarrow \gamma)$ — modus ponens on lines 2, 4.
6. $(\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)$ — modus ponens on lines 3, 5.
7. $\alpha \Rightarrow \gamma$ — modus ponens on lines 1, 6.

- Let's prove that $\alpha \Rightarrow$ is a theorem i.e let's show that $\vdash (\alpha \Rightarrow \alpha)$.

We use the following proofs:

1. $(\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)) \Rightarrow ((\alpha \Rightarrow (\alpha \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha))$ — Axiom 2.
2. $\alpha \Rightarrow ((\alpha \Rightarrow \alpha) \Rightarrow \alpha)$ — Axiom 1
3. $(\alpha \Rightarrow (\alpha \Rightarrow \alpha)) \Rightarrow (\alpha \Rightarrow \alpha)$ — modus ponens on lines 1, 2.
4. $\alpha \Rightarrow (\alpha \Rightarrow \alpha)$ — Axiom 1
5. $\alpha \Rightarrow \alpha$ — modus ponens on lines 3, 4.

31/10/12.

The examples of proofs we have seen may indicate that even proofs of relatively simple results may appear to be complicated.

Let's now see a result that will help us in the setting of proofs.

Appears in exams before

Theorem: (Deduction Theorem for propositional logic) Let S be a set of propositions, and let α, β be propositions. Then

$$S \vdash (\alpha \Rightarrow \beta) \text{ if and only if } S \cup \{\alpha\} \vdash \beta.$$

Proof: We will prove the two "directions" separately. Let's first assume that $S \vdash (\alpha \Rightarrow \beta)$, i.e there is a sequence of proposition $t_1, \ldots, t_n$ say such that t_n is $\alpha \Rightarrow \beta$ and each t_i $(1 \leq i \leq n)$ is either an axiom, or an element of S, or deduced by modus ponens on two earlier lines (so, there is a proof of $\alpha \Rightarrow \beta$ from S).

Then, if we simply add the two lines:
$(t_n : \alpha \Rightarrow \beta)$

t_{n+1} : α hypothesis (in $S \cup \{\alpha\}$)
t_{n+2} : β modus ponens on lines t_n, t_{n+1}

Then, we obtain a proof of β from $S \cup \{\alpha\}$, as required, i.e. $S \cup \{\alpha\} \vdash \beta$.

Let's now assume that $S \cup \{\alpha\} \vdash \beta$, and let's consider a proof of β from $S \cup \{\alpha\}$, i.e a sequence of propositions $t_1, \dots, t_m$, such that t_m is β, and each t_i $(1 \leq i \leq m)$ is either an axiom, or an element of $S \cup \{\alpha\}$, or deduced by modus ponens on two earlier lines.

We will try to obtain a proof $\alpha \Rightarrow \beta$ from S by replacing each line of the given proof, t_i say, by $\alpha \Rightarrow t_i$ (and by also trying to avoid using α as a hypothesis).

There are three possibilities to consider, for t_i, $1 \leq i \leq m$:

1) t_i is an axiom; we may still assume t_i; and can obtain $\alpha \Rightarrow t_i$ by using the following lines:

t_i	axiom
$t_i \Rightarrow (\alpha \Rightarrow t_i)$	Axiom 1
$\alpha \Rightarrow t_i$	Modus ponens.

So, we can valid replace $t_i \Rightarrow \alpha \Rightarrow t_i$.

→ then it remains a hypothesis.

2) t_i is a hypothesis, in $S \cup \{\alpha\}$

If $t_i \in S$, then we may proceed as above; we may replace t_i by the following lines:

t_i	hypothesis
$t_i \Rightarrow (\alpha \Rightarrow t_i)$	Axiom 1
$\alpha \Rightarrow t_i$	Modus ponens.

If t_i is the proposition, then it may no longer by a hypothesis (it may not be in S)

But, $\alpha \Rightarrow \alpha$ is a theorem, so we may obtain it without using any hypothesis, i.e. to replace α by $\alpha \Rightarrow \alpha$, we can simply write down a proof of $\alpha \Rightarrow \alpha$ instead of the α. (e.g. the proof we saw last time). Note that we do not need to) use α as a hypothesis in this case.

3) t_i is some proposition, δ say, which is deduced by modus ponens on two previous lines, e.g. say that for $j, k < i$, t_j is some $\gamma \in L_0$ and t_k is $\gamma \Rightarrow \delta$.

We may inductively assume that t_j and t_k have already been sucessfully replace by $\alpha \Rightarrow t_j$, $\alpha \Rightarrow t_k$ i.e. by $\alpha \Rightarrow \gamma$ and $\alpha \Rightarrow (\gamma \Rightarrow \delta)$. Then, the following sequence of lines will lead to $\alpha \Rightarrow \delta$.

$S \cup \{\alpha\} \vdash \beta$	$S \vdash (\alpha \Rightarrow \beta)$
$t_j : \gamma$	$\rightsquigarrow \alpha \Rightarrow \gamma$
$t_k : \gamma \Rightarrow \delta$	$\rightsquigarrow \alpha \Rightarrow (\gamma \Rightarrow \delta)$
$t_i : \delta$	$\alpha \Rightarrow \delta$

$$\left[\begin{array}{c} \vdots \\ \alpha \Rightarrow \gamma \\ \vdots \\ \alpha \Rightarrow (\gamma \Rightarrow \delta) \end{array}\right]$$

$(\alpha \Rightarrow (\gamma \Rightarrow \delta)) \Rightarrow ((\alpha \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \delta))$ Axiom 2

$(\alpha \Rightarrow \gamma) \Rightarrow (\alpha \Rightarrow \delta)$ Modus ponens.

$\alpha \Rightarrow \delta$ Modus ponens.

—/—

In each of the cases, we have shown how we can replace E_i by $\alpha \Rightarrow E_i$. So, in the proof of β from $S \cup \{\alpha\}$, we may replace each line E_i by $\alpha \Rightarrow E_i$, "without using α as a hypothesis.

In particular, the last line $E_m : \beta$ may be successfully replace by $\alpha \Rightarrow E_m$ i.e. $\alpha \Rightarrow \beta$. So, the above arguement shows that there is a proof of $\alpha \Rightarrow \beta$ from S. i.e. $S \vdash (\alpha \Rightarrow \beta)$ as required.

This concludes the proof of the theorem $\square$.

Let's see some examples of how this result may be used. Earlier, we gave a direct proof of $\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma\} \vdash (\alpha \Rightarrow \gamma)$. By Deduction Theorem, it suffices to show that

$$\{\alpha \Rightarrow \beta, \beta \Rightarrow \gamma, \alpha\} \vdash \gamma$$

as we do below:

1. $\alpha \Rightarrow \beta$ hypothesis
2. $\beta \Rightarrow \gamma$ hypothesis
3. α hypothesis
4. β Modus ponens on lines 1,3.
5. γ Modus ponens on lines 2,4.

Let's now use the Deduction Theorem to show that $\vdash (\neg\neg\alpha) \Rightarrow \alpha$.

Note: In the proofs that follow, we will sometimes assume the validity of theorems already shown.

By the Deduction Theorem, it suffices to prove $\{\neg\neg\alpha\} \vdash \alpha$.

We give a proof below:

1. $\neg\neg\alpha$ — hypothesis
2. $(\neg\alpha \Rightarrow \neg\neg\alpha) \Rightarrow ((\neg\alpha \Rightarrow \neg\alpha) \Rightarrow \alpha)$ — Axiom 3
3. $\neg\neg\alpha \Rightarrow (\neg\alpha \Rightarrow \neg\neg\alpha)$ — Axiom 1.
4. $(\neg\alpha) \Rightarrow (\neg\neg\alpha)$ — Modus ponens on lines 1, 3.

Reading week exercise $\vdash (\alpha \Rightarrow \neg\neg\alpha)$.

12/11/12

Note : In the proofs that follow, we will be able to assume the validity of theorems that we have already shown, and use them in proofs, we will justify these as "theorems".

Similiarly, in general, if we have already shown that $S \vdash \alpha$ (for $S \subset \mathcal{L}_0$, $\alpha \in \mathcal{L}_0$) then we may write down α in any proof that includes the elements of S as hypothesis and may write "assumed" to justify the use of such an α.

Such a convention works, essentially due to this result:

Proposition : Let $S \subset \mathcal{L}_0$ and $\alpha \in \mathcal{L}_0$. Then, if $S \vdash \alpha$, for any $\beta \in \mathcal{L}_0$

$$S \cup \{\alpha\} \vdash \beta \text{ if and only if } S \vdash \beta.$$

Proof : Let's first suppose that $S \vdash \beta$. Then any proof of β from $S \cup \{\alpha\}$, i.e $S \cup \{\alpha\} \vdash \beta$ as required.

Now suppose that $S \cup \{\alpha\} \vdash \beta$, and consider a proof of β from $S \cup \{\alpha\}$.

Since $S \vdash \alpha$, we may simply replace any occurrence of α in the above proof by a proof of α from S.

This gives a proof of β which does not use α as a hypothesis, i.e. it gives a proof of β from S. So $S \vdash \beta$ as required

$\square$.

Let's now give proofs of some theorems, using the Deduction Theorems.

$\vdash (\neg\neg\alpha) \Rightarrow \alpha$ By the Deduction Theorem, it is enough to show that $\{\neg\neg\alpha\} \vdash \alpha$, and we do so below:

1. $\neg\neg\alpha$ — hypothesis
2. $(\neg\alpha \Rightarrow \neg\neg\alpha) \Rightarrow ((\neg\alpha \Rightarrow \neg\alpha) \Rightarrow \alpha)$ — Axiom 3
3. $(\neg\neg\alpha) \Rightarrow (\neg\alpha \Rightarrow \neg\neg\alpha)$ — Axiom 1
4. $(\neg\alpha \Rightarrow \neg\neg\alpha)$ — Modus ponens on lines 1,3
5. $(\neg\alpha \Rightarrow \neg\alpha) \Rightarrow \alpha$ — Modus ponens on lines 2,4
6. $\neg\alpha \Rightarrow \neg\alpha$ — "theorem"
7. α — Modus ponens on lines 5,6

$\vdash \alpha \Rightarrow (\neg\neg\alpha)$ By the Deduction Theorem, it is enough to show that $\{\alpha\} \vdash (\neg\neg\alpha)$ and we do so below:

1. α — hypothesis
2. $(\neg\neg\neg\alpha \Rightarrow \neg\alpha) \Rightarrow ((\neg\neg\neg\alpha \Rightarrow \alpha) \Rightarrow \neg\neg\alpha)$ — Axiom 3
3. $(\neg\neg\neg\alpha) \Rightarrow (\neg\alpha)$ — "theorem"
4. $(\neg\neg\neg\alpha \Rightarrow \alpha) \Rightarrow \neg\neg\alpha$ — Modus ponens on lines 2,3
5. $\alpha \Rightarrow (\neg\neg\neg\alpha \Rightarrow \alpha)$ — Axiom 1
6. $\neg\neg\neg\alpha \Rightarrow \alpha$ — Modus ponens on lines 1,5
7. $\neg\neg\alpha$ — Modus ponens on lines 4,6.

$\underline{\vdash \neg\alpha \Rightarrow (\alpha \Rightarrow \beta)}$: By the Deduction Theorem, it suffices to prove $\{\neg\alpha\} \vdash (\alpha \Rightarrow \beta)$. By another use of the Deduction theorem, it suffices to prove $\{\neg\alpha, \alpha\} \vdash \beta$ and we do so below:

1.	α	hypothesis
2.	$\neg\alpha$	hypothesis.
3.	$(\neg\beta \Rightarrow \neg\alpha) \Rightarrow ((\neg\beta \Rightarrow \alpha) \Rightarrow \beta)$	Axiom 3.
4.	$\neg\alpha \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$	Axiom 1.
5.	$(\neg\beta \Rightarrow \neg\alpha)$	Modus ponens on lines 2,4
6.	$((\neg\beta) \Rightarrow \alpha) \Rightarrow \beta$	Modus ponens on lines 3, 5.
7.	$\alpha \Rightarrow (\neg\beta \Rightarrow \alpha)$	Axiom 1.
8.	$(\neg\beta) \Rightarrow \alpha$	Modus ponens on lines 1,7
9.	α	Modus ponens on lines 6,8.

Let's complete this section by noting that a result similar to the Deduction Theorem holds for semantic implication(s):

<u>Proposition</u>: Let $S \subset L_0$ and $\alpha, \beta \in L_0$. Then:

$$S \models (\alpha \Rightarrow \beta) \text{ if and only if } S \cup \{\alpha\} \models \beta.$$

<u>Proof</u>: We prove the direction separately:

- Suppose that $S \models (\alpha \Rightarrow \beta)$

Then, for any valuation v st $v(\alpha) = 1$ for all $s \in S$, it must be the case that $v(\alpha \Rightarrow \beta) = 1$

Consider a valuation v s.t. $v(s) = 1$ for all $s \in S$ and $v(\alpha) = 1$.

Then, since $S \models (\alpha \Rightarrow \beta)$ by assumption, we may deduce that $v(\alpha \Rightarrow \beta) = 1$. So, since $v(\alpha) = 1$ and $v(\alpha \Rightarrow \beta) = 1$, it must be the case that $v(\beta) = 1$. (by the definition of a valuation: if $v(\beta) = 0$, then $v(\alpha \Rightarrow \beta) = 0$.) So, for such a valuation, v, satisfying $v(s) = 1$ for all $s \in S$ and $v(\alpha) = 1$, it must be the case that $v(\beta) = 1$. Therefore, $S \cup \{\alpha\} \models \beta$, as required.

· Let's now suppose that $S \cup \{\alpha\} \models \beta$. Then, for any valuation v such that $v(s) = 1$ for all $s \in S$ and $v(\alpha) = 1$, we have $v(\beta) = 1$. Consider a valuation v st $v(s) = 1$ for all $s \in S$. We consider the two cases for $v(\alpha)$.

1) $\underline{v(\alpha) = 1}$: then, since $v(s) = 1$ $\forall s \in S$ and $v(\alpha) = 1$, we deduce that $v(\beta) = 1$, by assumption.

Then, since $v(\alpha) = 1$ and $v(\beta) = 1$: $v(\alpha \Rightarrow \beta) = 1$ as required.

2) $\underline{v(\alpha) = 0}$: Then, by the definition of a valuation $v(\alpha \Rightarrow \beta) = 0$ (irrespective of the value of $v(\beta)$).

In either case $v(\alpha \Rightarrow \beta) = 1$ so $S \vdash (\alpha \Rightarrow \beta)$, as required.

$\square$

The last result, together with the Deduction theorem, suggest a deep and underlying connection between the semantic and syntactic implication, which we study in the next section (coming up... now).

<u>Completeness Theorem for propositional logic:</u>
Here, we will try to show that semantic and syntactic implication are equivalent, in sense that, for $S \subset \mathcal{L}_0$ and $\alpha \in \mathcal{L}_0$.

$S \models \alpha$ if and only if $S \vdash \alpha$ ← completeness theorem.

Let's start by proving one direction of this result, let's show that our proofs are "sound".

<u>Soundness theorem for propositional logic:</u> Let $S \subset \mathcal{L}_0$, and $\alpha \in \mathcal{L}_0$.

If $S \vdash \alpha$ then $S \models \alpha$.

<u>Proof:</u> Suppose that $S \vdash \alpha$, i.e. that there exists a finite sequence of propositions $t_1, \dots, t_n$ say such that t_n is α, and each proposition t_i, $1 \leq i \leq n$, is either an axiom or a hypothesis (an element of S) or deduced by modus ponens on two earlier lines.

Consider a valuation v s.t. $v(s) = 1$ for all $s \in S$. We will show that $v(\alpha) = 1$, by showing that $v(t_i) = 1$ for every t_i $(1 \leq i \leq n)$.

There are three cases to consider:

1) t_i is an axiom: all our axioms are tautologies, so, by definition, $v(t_i)=1$ in these case (for any valuation v).

2) t_i is a hypothesis, i.e $t_i \in S$: $v(t_i)=1$ by assumption ($v(s)=1$ for all $s \in S$).

3) t_i is deduced by modus ponens on two earlier lines, t_j and t_k say ($j, k < i$).

So, t_i is some proposition δ

t_j ——— γ

t_k ——— $\gamma \Rightarrow \delta$.

We may inductively assume that we have already shown that $v(t_j) = v(\gamma) = 1$ and $v(t_k) = v(\gamma \Rightarrow \delta) = 1$. (since t_j, t_k are "earlier lines".)

Then, by definition of a valuation, it must be the case that $v(\delta)=1$, i.e $v(t_i)=1$, as required. So either case, $v(t_i)=1$. So $v(t_i)=1$ for each $1 \le i \le n$. In particular, $v(t_n) = v(\alpha) = 1$, so $S \models \alpha$

□.

The other direction of the Completeness Theorem is a bit more "tricky"; possibly (partly) due to the following reasons...

1) The set S may be infinite and yet we still have

to "boil down" $S' \vDash \alpha$ to a proof, a finite sequence of lines.

2) Our proofs only use three (types of) axioms; not all tautologies are axioms.

Let's assume that S' is finite and that all tautologies are axioms (i.e. if $\vDash \gamma$, then γ is an axiom, so $\vdash \gamma$).

Then, it would be relatively easy to show that $S \vdash \alpha$ if $S' \vDash \alpha$. Say $S' = \{s_1, \ldots, s_n\}$.

Then, since $S \vDash \alpha$: $\{s_1, \ldots s_n\} \vDash \alpha$.
$\{s_2, \ldots, s_n\} \vDash (s_1 \Rightarrow \alpha)$
$\{s_3, \ldots, s_n\} \vDash s_2 \Rightarrow (s_1 \Rightarrow \alpha)$ Using earlier proposition
$\vdots$
$\vDash s_n \Rightarrow ((s_{n-1} \Rightarrow \ldots (s_2 \Rightarrow (s_1 \Rightarrow \alpha))$
$\vdash s_n \Rightarrow ((s_{n-1} \Rightarrow \ldots (s_2 \Rightarrow (s_1 \Rightarrow \alpha))$

Since we assume all tautologies are axioms.

$\{s_n\} \vdash s_{n-1} \vdash (\ldots \Rightarrow (s_2 \Rightarrow (s_1 \Rightarrow \alpha)))$ by the
$\{s_n, s_{n-1}\} \vdash s_{n-2} \Rightarrow \ldots$ Deduction
$\{s_1, \ldots, s_n\} \vdash \alpha$ theorem,

$S \vdash \alpha$

—/—

But, we cannot "make the assumptions given above here, so we not try to give a general of "If $S \models \alpha$ then $S' \vdash \alpha$"

A <u>key idea</u> is that of <u>consistency</u>:

- A set of proposition S' ($S' \subset L_0$) is consistent if there is no proposition α ($\in L_0$) such that $S' \vdash \alpha$ and $S' \vdash (\neg\alpha)$.

Crucially, if a set is consistent we may extend it indefinitely to a (larger) consisent set:

<u>Proposition</u>: Let S' be a consisent of proposition then for any $\alpha \in L_0$, at least one of $S' \cup \{\alpha\}$, $S \cup \{\alpha\}$ is consistent.

14/11/12

From last time: A set of propositions S is consistent if there is no proposition α such that $S \vdash \alpha$ and $S \vdash \neg\alpha$. Let's now show that we may "extend" consistent sets indefinitely:

Proposition: Suppose that a set of propositions ($S \subset L_0$) is consistent. Then, for any proposition α, at least one of $S \cup \{\alpha\}$, $S \cup \{\neg\alpha\}$ is consistent.

Proof: Consider $\alpha \in L_0$, and consider $S \cup \{\neg\alpha\}$. Then if $S \cup \{\neg\alpha\}$ is consistent, we are done.

Suppose that $S \cup \{\neg\alpha\}$ is not consistent, i.e. there is some proposition β such that $S \cup \{\neg\alpha\} \vdash \beta$ and $S \cup \{\neg\alpha\} \vdash (\neg\beta)$

We will show that, in this case, $S \cup \{\alpha\}$ is consistent, by showing that "$S \cup \{\alpha\}$ proves the same thing as S"

Since $S \cup \{\neg\alpha\} \vdash \beta$ and $S \cup \{\neg\alpha\} \vdash \neg\beta$, we may use the Deduction Theorem to deduce that $S \vdash (\neg\alpha) \Rightarrow \beta$ and $S \vdash (\neg\alpha) \Rightarrow (\neg\beta)$ respectively. Then $S \vdash \alpha$ using the following arguement/proof:

$(\neg\alpha) \Rightarrow \beta$	"assumed" $S \vdash (\neg\alpha) \Rightarrow \beta$
$(\neg\alpha) \Rightarrow \neg\beta$	"assumed" $S \vdash (\neg\alpha) \Rightarrow (\neg\beta)$
$((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow (((\neg\alpha) \Rightarrow (\beta)) \Rightarrow \alpha)$	Axiom 3
$((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$	using modus ponens
α	using modus ponens

So, $S \vdash \alpha$.

Since $S \vdash \alpha$, we may deduce (as shown earlier) that, for any γ : $S \vdash \gamma$ if and only if $S \cup \{\alpha\} \vdash \gamma$. Therefore, $S \cup \{\alpha\}$ inherits consistency from S.

[If $S \cup \{\alpha\}$ were inconsistent, then, for some $\gamma \in \mathcal{L}_0$: $S \cup \{\alpha\} \vdash \gamma$ and $S \cup \{\alpha\} \vdash \neg\gamma$. Then $S \vdash \gamma$ and $S \vdash (\neg\gamma)$; this is a contradiction, since S is consistent.]

$\square$

We will use this result in the proof of the "main result" that will lead us to the Completeness Theorem:

Theorem: Let S be a set of propositions: If S is consistent, then S has a model.

Proof: Suppose S is consistent set. We will try to define a valuation $v : \mathcal{L}_0 \to \{0,1\}$ such that v is a model of S, i.e such that $v(s) = 1$ for all $s \in S$. Lets try to construct such a function.

We start by setting $v(s)=1$ for all $s \in S$ and try to extend this function defined for any proposition in $\mathcal{L}_0$.

Note that, as mentioned earlier $\mathcal{L}_0$ is countable, so we may "list" the elements of $\mathcal{L}_0$, say:
$\mathcal{L}_0 = \{\alpha_1, \alpha_2, \alpha_3, \dots, \alpha_n, \dots\}$.

Consider α_1: By the previous result, at least one of $S \cup \{\alpha_1\}$ and $S \cup \{\neg\alpha_1\}$ is consistent (since S is consistent)

If $S' \cup \{\alpha_1\}$ is consistent, then set $S_1' = S' \cup \{\alpha_1\}$. and set $v(\alpha_1) = 1$, $v(\neg\alpha_1) = 0$. Otherwise, $S \cup \{\neg\alpha_1\}$ is consistent, then set $S_1 = S' \cup \{\neg\alpha_1\}$ and set $v(\alpha_1) =$ $v(\neg\alpha_1) = 1$.

In either case, S_1 is consistent.

· Consider α_2. Since S_1' is consistant, $S_1' \cup \{\alpha_2\}$ or $S_1' \cup \{\neg\alpha_2\}$ is consistent.

If $S_1' \cup \{\alpha_2\}$ is consistent, then set $S_2' = S' \cup \{\alpha_2\}$ and set $v(\alpha_2) = 1$, $v(\neg\alpha_2) = 0$.

Otherwise, set $S_2' = S_1' \cup \{\neg\alpha_2\}$ and $v(\neg\alpha_2) = 1$, $v(\alpha_2) = 0$. In either case, S_2 is consistent.

We may proceed similarly to obtain a consistent S_n, for any n. Note that $S \subset S_1 \subset S_2 \subset S_3 \subset \ldots \subset S_n \subset \ldots$

Consider the "infinite union" $\bar{S}' = \bigcup_{n \in \mathbb{N}} S_n$

By construction, $\bar{S}'$ contains either α or $\neg\alpha$, for any proposition α.

Let's show that $\bar{S}'$ is consistent. Suppose not. If $\bar{S}$ is inconsistent, then, for some $\beta \in \mathcal{L}_0$: $\bar{S}' \vdash \beta$ and $\bar{S} \vdash (\neg\beta)$. Since proofs are finite, proofs of β and $\neg\beta$ from $\bar{S}$ will use only finitely many propositions from $\bar{S}'$.

As a result, if $\bar{S}' \vdash \beta$, and $\bar{S}' \vdash (\neg\beta)$, there must be natural numbers m, n such that $S_m \vdash \beta$ and $S_n \vdash (\neg\beta)$ there must be natural numbers m, n such that $S_m \vdash \beta$ and $S_m \vdash (\neg\beta)$.

Then, for some large enough number k (e.g. k = maximum of m and n)

$$S_k \vdash (\beta) \text{ and } S_k \vdash (\neg\beta).$$

Then S_k is inconsistent, but this contradicts the consistency of S_k, for any k. So, $\bar{S}$ is a consistent set

—/—

Also, $\bar{S}'$ is deductively closed, i.e. if $\bar{S}' \vdash \beta$ then $\beta \in \bar{S}'$ (for any $\beta \in \mathcal{L}_0$). Let's show this! Suppose not; suppose there is a $\beta \in \mathcal{L}_0$ such that $\bar{S}' \vdash \beta$, but $\beta \notin \bar{S}'$. Then, since $\bar{S}'$ contains either α or $\neg\alpha$ (for any $\alpha \in \mathcal{L}_0$), $\bar{S}'$ must contain $(\neg\beta)$. i.e $\neg\beta \in \bar{S}$. As a result, $\bar{S} \vdash \neg\beta$, i.e $\bar{S} \vdash \beta$ and $\bar{S} \vdash (\neg\beta)$. i.e $\bar{S}$ is inconsistent. This contradicts the consistency of $\bar{S}'$! $\bar{S}$ is deductively closed.

is a <u>theorem</u>.

So $\bar{S} \vdash (\neg\alpha) \Rightarrow (\alpha \Rightarrow \beta)$
By modus ponens $\bar{S} \vdash (\alpha \Rightarrow \beta)$
So, $\alpha \Rightarrow \beta \in \bar{S}$ (since $\bar{S}$ is deductively closed).
i.e $v(\alpha \Rightarrow \beta) = 1$ as required.

$\rightarrow$ Finally, suppose that $v(\alpha) = 1$, $v(\beta) = 0$. We need to show that $v(\alpha \Rightarrow \beta) = 0$, and we argue by contradiction.

Suppose that $v(\alpha \Rightarrow \beta) = 1$. Then $\alpha \Rightarrow \beta \in \bar{S}$. So $\bar{S} \vdash (\alpha \Rightarrow \beta)$.
Also, since $v(\alpha) = 1$, $\alpha \in \bar{S}$. So $\bar{S} \vdash \alpha$.

Then, using modus ponens, $\bar{S} \vdash \beta$ i.e $\beta \in \bar{S}$ (since $\bar{S}$ is deductively closed). So $v(\beta) = 1$.

This contradicts assumption that $v(\beta) = 0$.

So it must be the case that $v(\alpha \Rightarrow \beta) = 0$. So, our function $v : \mathcal{L}_0 \rightarrow \{0, 1\}$ is a <u>valuation</u>.
Also, by construction, $v(s) = 1$ for all $s \in S$.
So, as required, v is a model of S. $\square$

The above result plays an important role in the proof of:

<u>Adequacy Theorem</u>: Let $S \subset \mathcal{L}_0$ and $\alpha \in \mathcal{L}_0$: If $S \models \alpha$ then $S \vdash \alpha$.

Proof: We assume that $S' \models \alpha$, i.e. that for any valuation v such that $v(s)=1$ for all $s \in S'$, it must be the case that $v(\alpha)=1$ (i.e that $v(\neg\alpha)=0$).

In other words: there is no valuation $v: \mathcal{L}_0 \to \{0,1\}$ such that $v(s)=1$ for all $s \in S'$ and $v(\neg\alpha)=1$.

So $S' \cup \{\neg\alpha\}$ does not have a model.
Therefore, using the previous theorem, we deduce that $S' \cup \{\neg\alpha\}$ is inconsistent.

So, for some $\beta \in \mathcal{L}_0$: $S \cup \{\neg\alpha\} \vdash \beta$ and $S' \cup \{\neg\alpha\} \vdash (\neg\beta)$.

Hence, by the Deduction Theorem: $S' \vdash (\neg\alpha) \Rightarrow \beta$ and $S' \vdash (\neg\alpha) \Rightarrow (\neg\beta)$. Then, we can show $S' \vdash \alpha$ using the following argument.

$\neg\alpha \Rightarrow (\beta)$ (assumed)
$(\neg\alpha) \Rightarrow (\neg\beta)$ (assumed)
$((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow ((\neg\alpha \Rightarrow \beta) \Rightarrow \alpha)$ Axiom 3.
$(\neg\alpha \Rightarrow \beta) \Rightarrow \alpha$ Modus Ponens
α Modus Ponens.

So, we have shown that (if $S' \models \alpha$ then) $S' \vdash \alpha$, as required. $\square$

By combining the Soundness and Adequency Theorems, (for propositional logic), we obtain the Completeness Theorem:

19/11/12

From last time:

Theorem: If $S' \subset \mathcal{L}_0$ is consistent, then S' has a model.

Proof so far: (in brief): Try to contruct valuation $v: \mathcal{L}_0 \to \{0, 1\}$ satisfying $v(s)=1$ for all $s \in S'$.

We construct(ed) function $v: \mathcal{L}_0 \to \{0,1\}$ inductively.

$\mathcal{L}_0 = \{\alpha_1, \alpha_2, \dots, \alpha_n \dots\}$

- $v(s) = 1$ for all $s \in S'$ $\quad = S_1$
- $v(\alpha_1) = 1$, $v(\neg\alpha_1)$ if $S' \cup \{\alpha_1\}$ $\quad = S_1$
  $v(\alpha_1) = 0$, $v(\neg\alpha_1) = 1$ otherwise (if $S' \cup \{\alpha_1\}$ consistent)

⋮

$v(\alpha_{n+1}) = 1$, $v(\alpha_{n+1}) = 0$ if $S_n \cup \{\alpha_{n+1}\}$ consistent $v(\alpha_{n+1}) = 0$, $v(\alpha_{n+1}) = 1$ otherwise

⋮

Define function $v: \mathcal{L}_0 \to \{0,1\}$ this way and also obtained a "really big" consistent set $\overline{S'} = \bigcup_{n \in \mathbb{N}} S_n$

$\overline{S'}$ is consistent

$\overline{S}$ is deductively closed (If $\overline{S} \vdash \alpha$ then $\alpha \in \overline{S'}$)

Let's now complete the above proof:

It remains to check that a function $v : \mathcal{L}_0 \to \{0,1\}$ $v : \mathcal{L} \to \{0,1\}$ is a (well-defined) valuation (that models $\bar{S}$).

- Let's check that, for any $\alpha \in \mathcal{L}_0$
  $v(\neg\alpha) = 1$, if $v(\alpha) = 0$ and $v(\neg\alpha) = 0$ if $v(\alpha) = 1$.

This is ensured by construction of v.

- We also need to verify that, for any $\alpha, \beta \in \mathcal{L}_0$:

$v(\alpha \Rightarrow \beta) = 0$ if $v(\alpha) = 1$ and $v(\beta) = 0$.
$v(\alpha \Rightarrow \beta) = 0$ otherwise.

Lets verify this

→ Suppose $v(\beta) = 1$. Then, by construction: $\beta \in \bar{S}$.
So, $\bar{S} \vdash \beta$.

Now, note that $\beta \Rightarrow (\alpha \Rightarrow \beta)$ is an axiom
so $\bar{S} \vdash \beta \Rightarrow (\alpha \Rightarrow \beta)$

By modus ponens: $\bar{S} \vdash (\alpha \Rightarrow \beta)$

Since $\bar{S}$ is deductively closed: $\alpha \Rightarrow \beta \in \bar{S}$
So $v(\alpha \Rightarrow \beta) = 1$ as, required.

→ Suppose that $v(\alpha) = 0$. Then, $v(\neg\alpha) = 1$. So $\neg\alpha \in \bar{S}$
Then $\bar{S} \vdash (\neg\alpha)$.

Now, note that (as shown earlier): $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$

Completeness Theorem for propositional logic:

Let S be a set of propositions ($S \subset \mathcal{L}_0$) and α be a proposition ($\alpha \in \mathcal{L}_0$). Then

$$S \models \alpha \quad \text{if and only if} \quad S \vdash \alpha$$

We complete this chapter with two consequences of the Completeness Theorem:

Compactness theorem for propositional logic

Suppose that $\alpha \in \mathcal{L}_0$, and that S is a (possibly infinite) set of propositions such that $S \models \alpha$.

Then, there exist a finite subset of S, S' say, such that $S' \models \alpha$

Proof: Suppose $S \models \alpha$. Then, by the Completeness Theorem $S \vdash \alpha$.

But a proof is a finite sequence of proposition, and so a proof of α from S uses only finitely many hypotheses from S.
i.e there is a finite subset of S, S' say - such that $S' \vdash \alpha$.

So, by the Completeness Theorem, it must be the case that $S' \models \alpha$ (and S' is finite).

Decidability Theorem for propositional logic.

Given a finite set of proposition S' and any propositional α, there is an algorithm that (in a finite number of steps) decides whether or not $S' \vdash \alpha$.

Proof: By the Completeness Theorem, deciding if $S' \vdash \alpha$ is equivalent to deciding if $S' \models \alpha$.

We may decide this by using a finite truth table or a semantic tableaux that necessarily ends in a finite number of steps (since α, as a proposition, is made up of finitely many primitive propositions, and S' is finite). $\square$

Chapter 3: First order predicate logic.

In this chapter, we will study the notions of "truth" and "proof" in the general setting of first order predicate logic.

So, we will now deal with formulae, which may includes:

- variables
- the "$\forall$" symbol.
- any predicate of any given arity
- any functional of any given arity.

Semantic aspects of first order predicate logic.

In chapter 2, it was possible to "easily" define a valuation on all propositions.

Here, some formulae cannot be interpreted as ture or flase, e.g. if F is a unary "squaring" functorial then Fx $(= \ldots "x^2")$ is simply an expression.

Furthermore, some formulae may be ture and flase, in different cases. e.g statement: "for some binary functorial F, with identity E, every element has an inverse".

$$(\forall x)(\exists y)\,((F_{xy} = E) \wedge (F_{yx} = E))$$

Then:
If we work in $\mathbb{N} \cup \{0\}$, and F repersents addition
E ——— zero

then the statement is <u>flase</u>.

If we work with $\mathbb{Z}$, and F ——— addition
E ——— zero.

then the statement is <u>ture</u>.

If we work with $\mathbb{Z}$, and F ——— multiplication
E ——— one.

then the statement is <u>flase</u>.

So, we need some "structure", like $\mathbb{N}$ or $\mathbb{Z}$ above, to "interprate" formulae:

Definition: Let Π and Ω be given sets of predicates and functionals respectively. Then an $\mathcal{L}(\Pi, \Omega)$-structure consists of:

1) a non-empty set U.

2) for each n-ary predicate P, an n-ary relation on U P_U.

3) for each n-ary functional F, an n-ary function F_U on U (F_U is a function from U^n to U).

Notes:
1) We often write "U is an $\mathcal{L}(\Pi, \Omega)$-structure" if U is the non-empty mentioned above.

2) In many cases in this chapter (except towards the end), U will be a countable set, so that it "fits in" with our countable language (from chapter 1)

—/—

Let's now move towards "interpreting" formulae in given structure.

We first try to use variables unambiguously!

Definition(s): An occurrence of a variable x in a formula is free if the "x" is not within the scope of a "$\forall x$".

If the "x" is within the scope, the occurrence is bound (the "x" in "$\forall \underline{x}$" is also said to be bound). e.g. if x, y are variables, P unary predicate, Q binary predicate.

$\underbrace{\forall x\, P x}_{\text{bounded}}$ $\qquad$ $\underbrace{\forall x\, Qx}_{\text{bounded}}\underbrace{y}_{\text{free}}$ $\qquad$ $\underbrace{\forall x\, Qxx}_{\text{Bounded}}$

$(\underbrace{\forall x\, Px}_{\text{bounded}}) \vee Q\underbrace{x}_{\text{free}}\underbrace{y}_{\text{free}}$.

A hopefully more familiar example is:

Free

$$\int_0^x \underbrace{f(x)\, dx}_{\text{bound}} \xrightarrow[\text{written as}]{\text{more "properly"}} \int_0^x f(t)\, dt.$$

From now on, we may assume that all our formulae are clean: i.e. that in a given formula, we cannot have both bound and free occurrences of the same variables.

(we may achieve this by renaming the bound occurences of a variable, without changing "meaning").

Using this convention, we may define a <u>sentence</u> as a formula with no free (occurrence of) variables.

Sentences, as we will see, may always be interpreted as "true" or "false".

Let's now consider some "things" that will take "values" when we interprete them:

· The set of <u>terms</u> in a given first order predicate languages is defined inductively as follows:

1) Each variable symbol is a term

2) Each 0-ary functional is a terms.

3) For each n-ary functional F, and terms $\alpha_1, \ldots, \alpha_n$, $F\alpha_1, \ldots \alpha_n$ is a term.

21/11/12

From last time

sentence: a formula containing no free variables.

e.g. $(\forall x)(x \geq 1)$ is a sentence
$(\forall x)(x \geq y)$ is not a sentence
$(\forall x)(\forall y)(x \geq y)$ is a sentence

for a binary predicate "$\geq$" and variables x, y
0-ary functional "1".

—/—

Some other key definitions:

- The set of terms is defined inductively as follows:

1) Every variable is a term.
2) Every 0-ary functional symbol is a term.
3) If F is an n-ary functional symbol, and $t_1, \dots, t_n$ are terms then $F t_1, \dots, t_n$ is a term.

($+$ is a binary functional)

e.g. if $\bar{0}, \bar{1}$ are 0-ary functionals, and x, y are variables, then $x, y, 0, 1, x+1, y+0, (x+y)+1, (x+y)+y, x+y, 1+1$ are terms.

Closed terms, are terms containing no variables:
e.g. in the example above, $0, 1, 1+1$ are closed terms.

—/—

Using these ideas, we may now define the equivalent of a "valuation" for first order predicate logic:

<u>Definition</u>: Let $\mathcal{L}(\Pi, \Omega)$ be a given first order predicate language, and $\mho$ be an $\mathcal{L}(\Pi, \Omega)$-structure.
Then, the <u>interpretation</u> of (some) strings in $\mathcal{L}(\Pi, \Omega)$ is defined as follows:

1) Let F be an n-ary functional, and $t_1, \dots, t_n$ be closed terms. Then, the interpretation of $F(t_1, \dots, t_n)$ is $F_\mho((t_1)_\mho, \dots, (t_n)_\mho) \in \mho$

2) Let P be an n-ary predicate symbol and $t_1, \dots, t_n$ be closed terms. Then, the interpretation of $P(t_1, \dots, t_n)$ is $[P(t_1, \dots, t_n)]_\mho$ and

$$[P(t_1, \dots, t_n)]_\mho = \begin{cases} 1 \text{ if } P_\mho((t_1)_\mho, \dots, (t_n)_\mho) \text{ holds in } \mho \\ 0 \text{ if } P_\mho((t_1)_\mho, \dots, (t_n)_\mho) \text{ does not hold in } \mho. \end{cases}$$

3) If α is a sentence with interpretation 1, i.e. if $\alpha_\mho = 1$ then $\neg\alpha$ is interpreted as 0, and vice versa. So $(\neg\alpha)_\mho = 0$ if $(\alpha)_\mho = 1$ and $(\neg\alpha)_\mho = 1$ if $(\alpha)_\mho = 0$.

4) If α, β are <u>sentences</u> with interpretations $\alpha_\mho, \beta_\mho$ then

$$(\alpha \Rightarrow \beta)_\mho = \begin{cases} 0 \text{ if } \alpha_\mho = 1 \text{ and } \beta_\mho = 0. \\ 1 \text{ otherwise} \end{cases}$$

5) If $(\forall x)\alpha$ is a sentence, then the interpretation $((\forall x)\alpha)_\mho$ is defined as follows. Define a 0-ary predicate $\bar{u}$ for each u in $\mho$ ($\bar{u}_\mho = u$)

(*)

Examples: Let $\Pi = \{=\}$, $\Omega = \{+, 0, 1\}$, x, y variable

binary (=)   binary (+)   0-ary (0, 1)

Consider $(1+1)_U$ If $U = \mathbb{N}$: $(1+1)_{\mathbb{N}} = 1_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} \ldots \boxed{2_{\mathbb{N}}}$

If $U = \underbrace{\mathbb{Z}_2}_{\text{integers modulo } 2}$: $(1+1)_{\mathbb{Z}} = 1_{\mathbb{Z}_2} +_{\mathbb{Z}_2} 1_{\mathbb{Z}_2} \ldots \boxed{0_{\mathbb{Z}_2}}$

Consider $(1+1=0)_U$ if $U = \mathbb{N}$: $(1+1=0) = 0$ "i.e $(1+1=0)_{\mathbb{N}}$ is flase"

Look at $1_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} =_{\mathbb{N}} 0_{\mathbb{N}}$ does not hold

If $U = \mathbb{Z}_2$: $(1+1 = 0)_{\mathbb{F}_2} = 1$

$1_{\mathbb{F}_2} +_{\mathbb{F}_2} 1_{\mathbb{F}_2} =_{\mathbb{F}_2} 0_{\mathbb{F}_2}$ does hold

Then $(\neg((1+1) = 0))_{\mathbb{N}} = 1$
$(\neg(1+1=0))_{\mathbb{F}_2} = 0$.

—/—

$((\forall x)(x \geq 1))_U$ Substitute every $u \in U$ for x into "$x \geq 1$" and check if "$x \geq 1$" is <u>ture</u>.

<u>Notation</u>: If α is a formula, $\alpha[t/x]$ is a formula obtained by replacing every (free) occurrence of x in α by t. (t may be any term).

Let α be $x \geq 1$ Then:

$\alpha[y/x]$ is $y \geqslant 1$
$\alpha[0/x]$ is $0 \geqslant 1$
$\alpha[1/x]$ is $1 \geqslant 1$

—/—

(*) Then if $\alpha[\bar{u}/x]$ holds for each u in U then we say that

$(\forall x)\alpha$ is true $[(\forall x)\alpha]_v = 1$

Otherwise

$(\forall x)\alpha$ is false $[(\forall x)\alpha]_v = 0$.

—/—

e.g $[(\forall x)(x \geqslant 1)]_{\mathbb{N}} = 1$ $(\bar{1} \geqslant 1)_v$ holds $(\bar{2} \geqslant 1)_v$ holds, $(\bar{3} \geqslant 1)$ holds, ...

$[(\forall x)(x \geqslant 1)]_{\mathbb{Z}} = 0$ e.g. consider $-\bar{2}$ st $-\bar{2}_{\mathbb{Z}} = -2$
$(-\bar{2} \geqslant 1)_{\mathbb{Z}}$ does not hold

the actual integer -2

—/—

Notes: Not all strings have interpretations
e.g for a variable x, 0-ary functional $\bar{1}$, binary functional "+".

x, $x + \bar{1}$ have no interpretation

2) In part (5) of the above definition, we needed to define a 0-ary functional $\bar{u}$ for each $u \in U$. Technically, since our "language is countable", we should only be able to achieve this if the set U is countable

—/—

Consider $\alpha \in \mathcal{L}(\Pi, \Omega)$ and an $\mathcal{L}(\Pi, \Omega)$-structure U.

Then if $\alpha_U = 1$ then we say that α is <u>true</u> in U or that α holds in/for U.

or that U is a <u>model</u> of α
U <u>models</u> α.

Similarly, if for a set of formulae S, $(s)_U = 1$ for all $s \in S$. then S holds in/for U (S is true in U) or, equivalently, U is a <u>model</u> of S.

—/—

Let's now apply these constructions/use these ideas in specific mathematical systems:

<u>Definition</u>: A <u>theory</u> T in a specified $\mathcal{L}(\Pi, \Omega)$ first order predicate language is a set of <u>sentences</u> in $\mathcal{L}(\Pi, \Omega)$

A theory consists of the set of rules that we use to define mathematical objects.

⋮ See this next time.

26/11/12

A *sentence* is a formula that does not contain *free* variables. Given a (specified) first order predicate language $\mathcal{L}(\Pi, \Omega)$ a theory (in $\mathcal{L}(\Pi, \Omega)$) is a *set of sentences* in $\mathcal{L}(\Pi, \Omega)$.

Consider a sentence α in $\mathcal{L}(\Pi, \Omega)$. If, for some $\mathcal{L}(\Pi, \Omega)$-structure U, $\alpha_U = 1$ (i.e if the interpretation of α is 1 in U), we say that α is *true* in U, or, equivalently, that α *holds* in U.

If, for some theory T, $\alpha_U = 1$ for each $\alpha \in T$, then we say that T holds in U.

Equivalently if $\alpha_U = 1$ we say that U is a *model* of α (U models α)

$\alpha_U = 1$ if $\alpha_U = 1$ for all $\alpha \in T$, we say that U is a *model* of T (U *models* T).

e.g. $\mathbb{N}$ is a model for "$(\forall x)(x \geq 1)$"
$\mathbb{Z}$ is not a model for/of "$(\forall x)(x \geq 1)$"

Examples of theories

We first note that for theories including the equality predicate "$=$", we will always add some sentances that describe some of the main properties of equality:

1) $(\forall x)(x = x)$ Reflexivity
2) $(\forall x)(\forall y)((x = y) \Rightarrow (y = x))$ Symmetry

3) $(\forall x)(\forall y)(\forall z)(((x=y)\wedge(y=z))\Rightarrow(y=z))$

transitivity.

If other predicates and functionals are present, which have positive arities, we will also include some <u>substitutivity</u> properties:

e.g: Suppose a theory includes the binary predicates "$\geq$" and the binary functional "+" together with equailty. Then we will use the following sentences.

$$(\forall x)(\forall y)(\forall z)((x=y)\Rightarrow((x\geq z)\Rightarrow(y\geq z)))$$

$$(\forall x)(\forall y)(\forall z)((x=y)\Rightarrow((z\geq x)\Rightarrow(z\geq y)))$$

$$(\forall x)(\forall y)(\forall z)((x=y) = ((z+x)=(z+y)))$$

1) <u>Theory of groups</u>: Let $\Pi=\{=\}$ and $\Omega=\{\cdot, E\}$

where "=" has arity 2

"$\cdot$" has arity 2

"E" has arity 0

Then, the following theory has all groups as models:

$$\left.\begin{array}{l}
\cdot(\forall x)(((x\cdot E)=x)\wedge((E\cdot x)=x))\\
\cdot(\forall x)(\exists y)(((x\cdot y)=E)\wedge((y\cdot x)=E))\\
\cdot(\forall x)(\forall y)(\forall z)(((x\cdot y)\cdot z)=(x\cdot(y\cdot z)))\\
\cdot(\forall x)(x=x)\\
\cdot(\forall x)(\forall y)((x=y)\Rightarrow(y=x))\\
\cdot(\forall x)(\forall y)(\forall z)(((x=y)\wedge(y=z))\Rightarrow(x=z))
\end{array}\right\}(*)$$

$$\left.\begin{array}{l}
\cdot(\forall x)(\forall y)(\forall z)((x=y)\Rightarrow((x\cdot z)=(y\cdot z)))\\
\cdot(\forall x)(\forall y)(\forall z)((x=y)\Rightarrow((z\cdot x)=(z\cdot y)))
\end{array}\right\}(*)$$

(*) Basic equality sentences

(*) Substitutivity sentences.

<u>Possibles problem</u>: We usually would like equality to <u>only</u> identify non-distinct elements of a structure. Given a structure U, with distinct elements a and b then we do not want $a=b$ to hold.

From now on, we will assume that all our models satisfy this i.e that $a=b$ in a structure U if and only if a and b are the same element in U.

Such "nice" models are known as <u>normal models</u>

Then, U is a <u>normal</u> model of the given theory if and only if U forms a <u>group</u>.

2) <u>Theory of posets</u> $\Pi = \{=, \leq\}$, $\Omega = \emptyset$.

- $(\forall x)(x \leq x)$
- $(\forall x)(\forall y)(((x \leq y) \wedge (y \leq x)) \Rightarrow (x = y))$
- $(\forall x)(\forall y)(\forall z)(((x \leq y) \wedge (y \leq z)) \Rightarrow (x \leq z))$
- $(\forall x)(x = x)$
- $(\forall x)(\forall y)((x = y) \Rightarrow (y = x))$
- $(\forall x)(\forall y)(\forall z)(((x = y) \wedge (y = z)) \Rightarrow (x = z))$
- $(\forall x)(\forall y)(\forall z)((x = y) \Rightarrow ((x \leq z) \Rightarrow (y \leq z)))$
- $(\forall x)(\forall y)(\forall z)((x = y) \Rightarrow ((z \leq x) \Rightarrow (z \leq y)))$

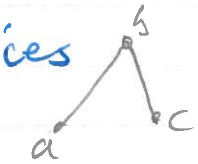
So, a structure U is a <u>normal</u> model of this theory if and only if U is a poset

—/—

—/

3) A graph consist of a (non empty) set of vertices and a (possibly empty) set of edges.

An edge may connect two distinct vertices



· An edge may not connect a vertex with itself.

· It is not necessary for two vertices to be conneted by an edge.

Lets define a theory of graphs:

$$\Pi = \{ \sim \}$$

binary predicate saying "$x \sim y$" if there is an edge between x and y

1) $(\forall x)(\neg(x \sim x))$

2) $(\forall x)(\forall y)((x \sim y) \Rightarrow (y \sim x))$

—/—

Lets now try to make theories more specific

E.g. How can we write down a theory that models groups of order 3 (only)?

We may extend our set of functionals so that it "identifies" 3 constants: $\Pi = \{=\}$,

$\Omega = \{\circ, E, A, A_2\}$

arity 0

and then add the following sentences to the earlier to the earlier theory of groups:

· $(\forall x)((x = E) \vee (x = A_1) \vee (x = A_2))$

← Up to this point, we model all groups of order up to 3

· $\neg(E = A_1)$
· $\neg(E = A_2)$
· $\neg(A_1 = A_2)$

So E, A_1, A_2 are distinct in a given structure.

So, overall, our theory models groups of order 3 (only).

· $(\neg(E = A_1)) \wedge (\neg(E = A_2)) \wedge (\neg(A_1 = A_2))$.

Similarly, for any natural number n, we may produce theories that models all groups of order n, or of order up to n.

Can we find a theory that models all finite groups, but not any infinite groups, within a first order predicate language?

No, as we will see later on!

<u>Syntactic aspects of first order predicate logic</u>:
To define proofs in this setting, we use the following axioms:

<u>Axiom 1</u>: $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ for all formulae α, β

<u>Axiom 2</u>: $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ for all formulae α, β, γ

<u>Axiom 3</u>: $(\neg\alpha \Rightarrow \neg\beta) \Rightarrow (((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$ for all formulae α, β

<u>Axiom 4</u>: $(\forall x)\alpha \Rightarrow \alpha[t/x]$ for a variable x, formula α, and term t such that no free variable of t appears bound in $(\forall x)(\alpha)$.

e.g $(\forall x)\overbrace{(x>y)}^{\alpha}$

$t = z \hookrightarrow z > y$
$t = 2 \hookrightarrow 2 > y$
$t = \frac{1}{2}y \hookrightarrow \frac{1}{2}y > y$

Axiom 5. $((\forall x)(\alpha \Rightarrow \beta)) \Rightarrow (\alpha \Rightarrow (\forall x)\beta)$
assuming no free occurrences of x in α.

e.g. $(\forall x)((y=1) \Rightarrow (x \geq y)) \Rightarrow (y=1) \Rightarrow (\forall x)(x \geq y)$ ✓

$(\forall x)((x>3) \Rightarrow (x>2)) \Rightarrow ((x>3) \Rightarrow (\forall x)(x>2))$ ✗

28/11/12.

For all formula α, β, γ and any variable x.

Axiom 1 $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

Axiom 2 $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$

Axiom 3 $((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow (((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$

Axiom 4 $(\forall x)\alpha \Rightarrow \alpha[{}^{t}/_{x}]$ t is a term such that no free variable in t is bound in α.

Axiom 5 $(\forall x)(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow ((\forall x)\beta)$ if α contains no free occurrences of x

Justification for the "caveats" in axioms 4 & 5

$(\forall x)(\forall y)(x+y = y+x) \Rightarrow (\forall x)(z+y = y+z)$

Set $t = z$ and use Axiom 4.

Similarly, if $\bar{z}$ is a 0-ary functional, then setting $t = \bar{z}$.

$(\forall y)(\bar{z} + y = y + \bar{z})$

constant

set $x = \bar{z}$

Consider $(\forall x)(\exists y)(x \cdot y = E) \Rightarrow (\exists x)(\bar{z} \cdot y = E)$ ✓

$(\forall x)(\exists y)(x \cdot y = E) \Rightarrow (\exists y)(y \cdot y = E)$ ✗

not allowed, since here $t = y$, and y is bound in α
$\alpha = (\exists y)(x \cdot y = E)$

As for Axiom 5, consider:

$$(\forall x)(\ y=1 \Rightarrow x+y=x+1\ )$$
$$\Downarrow$$
$$(y=1) \Rightarrow ((\forall x)(x+y=x+1)) \qquad \underline{\text{allowed}}$$

However, the following is <u>not</u> an instance of <u>Axiom 5</u>

$$(\forall x)((x=0) \Rightarrow (y+x=y))$$
$$\Downarrow$$
$$(x=0) \Rightarrow (\forall x)((y+x=y))$$

free variable

bound x's

We will also use the following <u>rules of deduction</u>:

1) <u>Modus ponens</u>: For formulae α, β: From α and $\alpha \Rightarrow \beta$, we may deduce β.

2) <u>Generalisation</u>: From a formula $\underline{\alpha}$, we may deduce $\underline{(\forall x)\alpha}$.

if no free occurrences of x appear in the premises/hypotheses used to obtain α

—/—

<u>Then</u>, given a set of formulae S' ($S' \subseteq L$), and a formula α. A proof of α from S' consist of a finite, ordered sequence of formulae $t_1, \dots, t_n$

say, such that t_n is α and for each t_i, $(1 \leq i \leq n)$, t_i is either:

1) **an axiom**
2) an element of S' (a hypothesis)
3) deduced by modus ponens on two earlier lines i.e for $j, k < i$: t_j is some formula β
   t_k is some formula $\beta \Rightarrow \gamma$
   and t_i is γ.
4) t_i is deduced by/using generalisation on an earlier line, i.e. for for $j < i$: t_j is a formula δ and t_i is the formula $(\forall x)\delta$.

(assuming no free occurrence of the variable x was used to obtain δ)

Examples of proofs

Let T be the theory of groups, with $\Pi = \{=\}$ (arity 2) and $\Omega = \{\cdot, E\}$ ($\cdot$: arity 2, E: arity 0.)

Let's show that $T \vdash (y = y)$; consider the following proof:

1.	$(\forall x)(x = x)$	hypothesis
2.	$(\forall x)(x = x) \Rightarrow (y = y)$	Axiom 4.
3.	$y = y$	Modus ponens on lines 1, 2

- Let's show that $\{T, z=E\} \vdash E=z$.

1. $(\forall x)(\forall y)((x=y) \Rightarrow (y=x))$ hypothesis
2. $(\forall x)(\forall y)((x=y) \Rightarrow (y=x)) \Rightarrow (\forall y)((z=y) \Rightarrow (y=z))$ Axiom 4
3. $(\forall y)((z=y) \Rightarrow (y=z))$ Modus ponens on lines 1, 2.
4. $(\forall y)((z=y) \Rightarrow (y=z)) \Rightarrow ((z=E) \Rightarrow (E=z))$ Axiom 4
5. $(z=E) \Rightarrow (E=z)$ Modus ponens on lines 3, 4.
6. $z=E$ hypothesis
7. $E=z$ Modus ponens on lines 5, 6.

In the first example, we may also add the line

4. $(\forall y)(y=y)$ Generalisation.

In the second example, we cannot write:

"8. $(\forall z)(E=z)$"

This is not allowed, since a free occurrence of z appeared in our hypothesis, namely in "$z=E$".

We note that a result analogous to the Deduction Theorem for propositional logic holds in this setting too:

Deduction Theorem for first order predicate logic
Given a first order predicate language $\mathcal{L}(\Pi, \Omega)$ and $S \subset \mathcal{L}(\Pi, \Omega)$, $\alpha, \beta \in \mathcal{L}(\Pi, \Omega)$:

$$S \vdash (\alpha \Rightarrow \beta) \text{ if and only if } S \cup \{\alpha\} \vdash \beta$$

—/—

Notes the syntax of first order predicate logic:

1) Axioms 4, and 5 and generalisation, are the "tools allowing us to deal with variables.

2) Axioms 4 and 5 correspond to tautologies when suitably interpreted in a given setting.

3/12/12

Completeness Theorem for first order predicate logic

Just as in the case of propositional logic, first order predicate logic is complete, i.e. "$S \models \alpha$ if and only if $S \vdash \alpha$" when the elements of the set S, and α, are "nice" formulae which may be interpreted as true or false, i.e when we are dealing with sentences.

In this setting, we may prove the Soundness Theorem for first order predicate logic (the proof is similar to the proof of the corresponding result in chapter 2).

Soundness Theorem: Let S be a set of sentences in first order predicate language $\mathcal{L}(\Pi, \Omega)$ say, and let α be a sentence in $\mathcal{L}(\Pi, \Omega)$:

If $S \vdash \alpha$ then $\underbrace{S \models \alpha}$.

for every $\mathcal{L}(\Pi, \Omega)$-structure U
for which $s_U = 1$ for all $s \in S$
we must have $\alpha_U = 1$

Furthermore, as in chapter 2, we may use the notion of consistency to prove the "other direction" of the theorem:

A set of sentences S in a first order predicate language, $\mathcal{L}(\Pi, \Omega)$ say is consistent if there is no sentence α in $\mathcal{L}(\Pi, \Omega)$ such that $S \vdash \alpha$ and $S \vdash (\neg\alpha)$

Otherwise, we say that S is inconsistent.

Using this idea (and some work) we may prove the following key result:

<u>Theorem</u>: Let S be a set of sentences in a first order predicate language.

If S is <u>consistent</u>, the S has a <u>model</u>.

This theorem the leads to:

<u>Adequacy Theorem for first order predicate logic</u>.
Let S be a set of <u>sentences</u> in a first order predicate language $\mathcal{L}(\Pi, \Omega)$ and α be a sentences in $\mathcal{L}(\Pi, \Omega)$:

If $S \vDash \alpha$ then $S \vdash \alpha$:

Combining the soundness and Adequacy Theorem, we obtain:

Let S be a set of <u>sentences</u> in a first order predicate language, $\mathcal{L}(\Pi, \Omega)$ say, and α be a <u>sentence</u> in $\mathcal{L}(\Pi, \Omega)$

$S \vDash \alpha$ if and only if $S \vdash \alpha$.

We may restate the Completeness Theorem as follows:

S has a <u>model</u> <u>if and only if</u> S is <u>consistent</u>.

An actual $\mathcal{L}(\Pi, \Omega)$-structure U such that U models S (i.e $s_U = 1$ for each $s \in S$).

Lets consider some consequences of the Completeness Theorem.

<u>Compactness Theorem:</u> (possibly infinite)
Let S be a set of sentences in some first order predicate language, and α be a sentence in that language.

Then $S \vdash \alpha$ if and only if $S' \vdash \alpha$ when S' is a finite subset of S.

<u>Proof</u>: Similar to the one in chapter 2.

Alternative form of the Compactness Theorem:

· Let S be a (possibly infinite) set of sentences in the language $\mathcal{L}(\Pi, \Omega)$. If every <u>finite</u> subset of S has a model, then so does S.

<u>Proof</u>: Consider the set S, and assume that S does <u>not</u> have a model (i.e. we will prove this result by contradiction). By the Completeness Theorem, we deduce that S is <u>inconsistent</u>, i.e there is a sentence α in $\mathcal{L}(\Pi, \Omega)$ such that $S \vdash \alpha$ and $S \vdash (\neg\alpha)$.

Since, proofs are <u>finite</u> (sequence of formulae), proofs of $\alpha, \neg\alpha$ from S will only finitely many elements of S. i.e there must exist finite subsets of S, S', S'' say, such that $S' \vdash \alpha$ and $S'' \vdash (\neg\alpha)$.

Then, the finite subset $S' \cup S''$ satisfies $S' \cup S'' \vdash \alpha$ and $S' \cup S'' \vdash (\neg \alpha)$

Then $S' \cup S''$ is inconsistent

So, by the Completeness Theorem, $S' \cup S''$ does not have a model.

This contradicts the assumption that every finite subset of S has a model. So, we deduce that S does have a model as required.

Let's consider an important consequence of the Compactness Theorem:

Upward Lowenheim - Skolem Theorem.
Let T be a theory in a first order predicate language $\mathcal{L}(\Pi, \Omega)$ such that T has arbitrarily large finite models (i.e such that, for any natural number $n \in \mathbb{N}$, there exist an $\mathcal{L}(\Pi, \Omega)$-structure U with at least n elements which is a model of T)

Then, T also has a infinite model (countable infinite).

Proof: We construct an infinite model for T by "extending" the language $\mathcal{L}(\Pi, \Omega)$ and using the Compactness Theorem.

We first extend the set of functionals, Ω, so that it includes infinitely many constants.

Set $\Omega' = \Omega \cup \{c_1, c_2, \dots c_n, \dots\}$

i.e $\Omega' = \Omega \cup \{c_i : i \in \mathbb{N}\}$.

where $c_1, c_2, \dots$ are functionals of arity 0.

We extend our theory to one where "the constants are all distinct".

Set $T' = T \cup \{\neg(c_1 = c_2), \neg(c_1 = c_3), \neg(c_2 = c_3), \neg(c_1 = c_4), \dots\}$.

i.e set $T' = T \cup \{\neg(c_i = c_j) : i, j \in \mathbb{N} ; i \neq j\}$.

Now, consider T' as a theory in $\mathcal{L}(\Pi, \Omega')$

Consider S', a finite subset of T'.

Since S' is finite, it inculdes only finitely many of the constants $c_1, c_2, \dots$ and only finitely many sentences of the form $\neg(c_i = c_j)$ for $i, j \in \mathbb{N}$

Note that T has arbitrarily large finite models, so there must be a model of S'.

(e.g. if S' mentions n constants, then any model of T that contains at least n elements will be a model of S').

This works for any finite subset of T'.

So, by the Compactness theorem, T' has a model. Such a model is necessarily (countably) infinite, so T' has an infinite model.

Since the original theory, T, is a subset of T', any model of T' will also be a model of T.

So, T also has an infinite model, as required. $\square$

If we extend our original first order predicate language to one dealing with uncountable many symbols, we may similarly prove an "uncountable" version of the above theorem:

<u>Upward Lowenheim - Skolem Theorem</u> (uncountable version).

Let T be a theory in a first order predicate language $\mathcal{L}(\Pi, \Omega)$ such that T has a <u>countable infinite</u> model.

Then, T also has an <u>uncountably infinite</u> model.

Let's now try to define the set of natural numbers, $\mathbb{N} = \{1, 2, 3, \ldots\}$ within a first order predicate language:

S: sucessor function "$S(x) = x+1$"

We set $\Pi = \{=\}$, $\Omega = \{0, S\}$

binary (for $=$), arity 0 (for 0), arity 1 (for S).

Consider the following theory, the theory of (weak) <u>Peano arithmetic</u>.

PA 1) $(\forall x)(x=x)$
PA 2) $(\forall x)(\forall y)((x=y) \Rightarrow (y=x))$
PA 3) $(\forall x)(\forall y)(\forall z)(((x=y) \wedge (y=z)) \Rightarrow (x=z))$
PA 4) $(\forall x)(\forall y)((x=y) \Rightarrow (s(x)=s(y)))$
PA 5) $(\forall x)(\forall y)((s(x)=s(y)) \Rightarrow (x=y))$
PA 6) $(\forall x)(\neg(s(x)=1))$
PA 7) $(\forall y_1)(\forall y_2)\ldots(\forall y_n)((p[1/x] \wedge (p \Rightarrow p[s(x)/x])) \Rightarrow (\forall x)p)$

p is a formula containing free occurrences of $x, y_1, \ldots, y_n$

<u>Notes</u>: • This "weak" version of Peano arithmetic does not include functionals representing "addition" and "multiplication", as well as related sentences eg $(\forall x)(\forall y)((x+y)=(y+x))$.

Including such functionals and sentences leads to "stronger" versions of Peano arithmetic.

—/—

• <u>Notes</u>: The sentance PA 7 is present in order to express the idea of "mathematical induction" (in $\mathbb{N}$)

The variables $y_1, \ldots, y_n$ that appear allow us to "perform" multiple "inductions" ("inductions" within "induction")

e.g. if we wish to prove $(\forall y)(\forall x)((x+y)=(y+x))$

in a version of Peano arithmetic including addition, we may use PA 7 twice as follows:

Firstly, in PA 7 set $y = y$ and $x = x$ and let p be $x+y=y+x$. to obtain $(\forall x)(x+y=y+x)$.

Then, set x to y in PA 7 and let p be $(\forall x)(x+y=y+x)$ to obtain $(\forall y)(\forall x)(x+y=y+x)$.

The set $\mathbb{N}$ of natural numbers is a (normal model of weak Peano arithmetic.

However, the uncountable version of the Upward Löwenheim-Skolem Theorem shows that the theory described also has an uncountable model.

Similarly, any theory that has the natural numbers as a model will also have an uncountable model.

This is, in some senses a "deficiency" of first order predicate logic: no first order predicate theory exist that has the natural numbers as the unique normal model.

2012

5a) <u>Def</u>: The set of recursive (partial) function is defined (inductively) as follows:

1) Any zero function $f: \mathbb{N}_0^k \to \mathbb{N}_0$, $f(n_1, \ldots, n_k) = 0$ is recursive.

2) The successor function $f: \mathbb{N}_0 \to \mathbb{N}$, $f(n) = n+1$, is recursive.

3) Any projection function $f: \mathbb{N}_0^k \to \mathbb{N}_0$, $f(n_1, \ldots, n_k) = n_i$, is recursive for $1 \leq i \leq k$.

4) Applying composition to recursive (partial) function lead to a recursive (partial) function.

5) Applying primitive recursion to recursive (partial) function leads to recursive (partial) function.

6) Applying minimalisation to a recursive (partial) function leads to a recursive, possibly, partial function.

5/12/12

Chapter 4: Computability

In this chapter, we will study (ideas related to) computable and recursive (partial) functions, and the interplay between them

Computable (partial) functions:

The basic object of this section is an abstract, idealised machine.

Definition: A register machine consists of the following:

- a sequence of registers $R_1, R_2, R_3, \ldots$ each of which is capable of being assigned a non-negative integer. (Note: In this chapter, we will use $\mathbb{N}_0$ to denote the set of non-negative integers: $\mathbb{N}_0 = \{0, 1, 2, 3, \ldots\}$)

- a program, which consists of a finite, specified number of states $S_0, S_1, \ldots, S_n$ say such that:
  - each S_i $(1 \leq i \leq n)$ corresponds to an instruction
    - S_1 corresponds to the initial state (to the instruction we perform first).
    - S_0 is the terminal state; on reaching it, the program ends.

There are two types of instructions, that can be associated to a state S_i:

1) An instruction which adds 1 to a register, R_j say, and then moves to a state S_k

$$S_i \xrightarrow{R_j + 1} S_k$$

2) An instruction which:
- If $R_j > 0$, substract 1 from R_j, and moves to state S'_k
- If $R_j = 0$, moves to state S'_l.

$$\begin{tikzcd} S'_i \arrow[rr, bend left, "R_j - 1"] \arrow[dr, dotted] & & S'_k \\ & S'_l & \end{tikzcd}$$

<u>Examples of register machines:</u>

- a register machine that adds 1 to R_1.

$$S_1 \xrightarrow{R_1 + 1} S_0$$

R_1	R_2	R_3	R_4	
1	0	3	2	...

↓

2	0	3	2	...

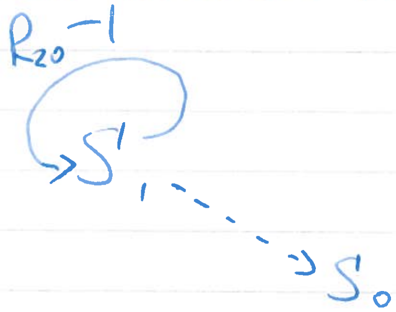
- a register machine that adds 1 to R_j:

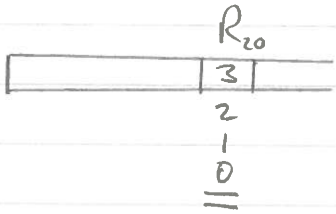
$$S'_1 \xrightarrow{R_j + 1} S'_0$$

- a register machine that adds 3 to R_2:

$$S'_1 \xrightarrow{R_2 + 1} S_2 \xrightarrow{R_2 + 1} S_3 \xrightarrow{R_2 + 1} S_0$$

- a register machine that clears R_{20}:
(i.e. that makes R_{20} hold the value 0 at the end, whatever the original value is):





—/—

We will try to investigate the types of functions that can be expressed using register machines:

<u>Definition</u>: A function $f: \mathbb{N}_0^k \to \mathbb{N}_0$ is <u>computable</u> if there exists a register machine such that, when started with n_1 in R_1, n_2 in R_2, n_k is R_k, and zero values in the remaining registers, the associated program ends (i.e. reaches state S_0) with $f(n_1, \dots, n_k)$ in R_1.

<u>Examples:</u>

1) The function $f: \mathbb{N}_0 \to \mathbb{N}$ is computable
$$n \longmapsto n+1$$

Below is a diagrammatic description of a register machine that computes f:

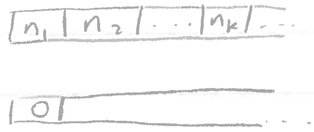
$$S_1 \xrightarrow{R_1+1} S_0$$

2) The function $f: \mathbb{N}_0 \to \mathbb{N}_0$, $f(n) = n+3$, is computable e.g via:

$$S_1 \xrightarrow{R_1+1} S_2 \xrightarrow{R_1+1} S_3 \xrightarrow{R_1+1} S_0$$

3) The function $f: \mathbb{N}_0^k \to \mathbb{N}_0$ "the zero function"

$$(n_1, \dots, n_k) \longmapsto 0$$

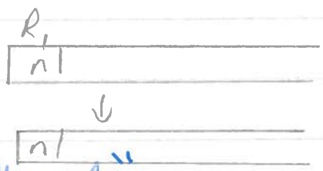


is computable, e.g via:

$$\begin{tikzcd} S'_1 \arrow[loop above, "R_1 - 1"] \arrow[r, dashed] & S_0 \end{tikzcd}$$

4) The identity function $f: \mathbb{N}_0 \to \mathbb{N}_0$, where $f(n) = n$, is computable e.g via:

$$S_1 \xrightarrow{R_5 + 1} S'_0$$

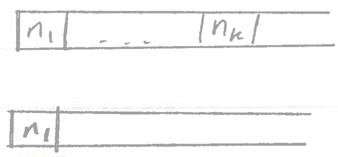


"use any program that leaves R_1 unaltered".

5) The projection $f: \mathbb{N}_0^k \to \mathbb{N}_0$, defined via $f(n_1, \dots, n_k) = n_i$ for some $1 \leq i \leq k$. is computable:

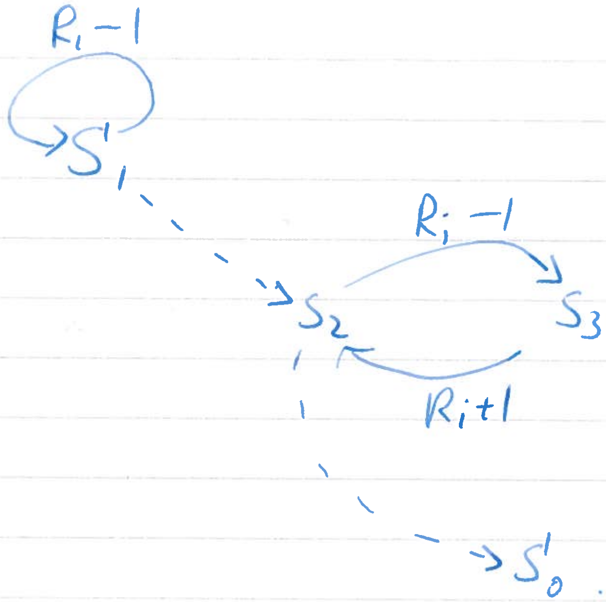
If $i = 1$, we need a program that "doesn't change R_1" e.g use:

$$S'_1 \xrightarrow{R_5 + 1} S'_0$$

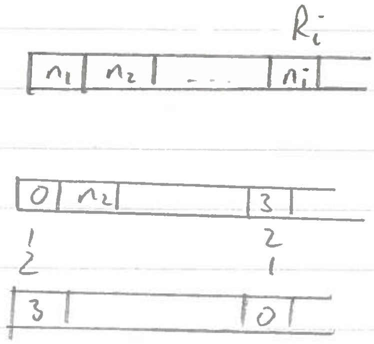


$f(n_1, \dots, n_k) = n_1$

If $i > 1$, we first "clear" R_1 and then add the value of R_i to R_1:



$R_1 - 1$
S'_1
$R_i - 1$
S_2
S_3
$R_i + 1$
S'_0

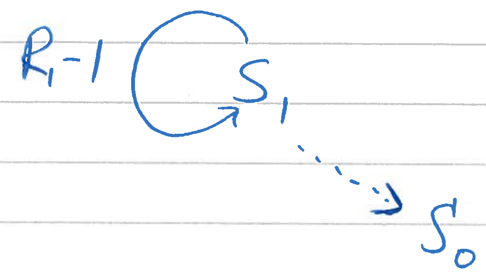


R_i
n_1 n_2 ... n_i
0 n_2 3
1 2
2 1
3 0

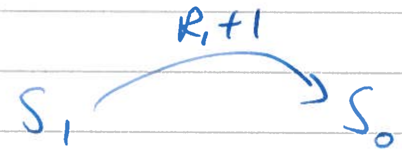
10/12/12.

Examples of Computable functions:

- Zero function ...



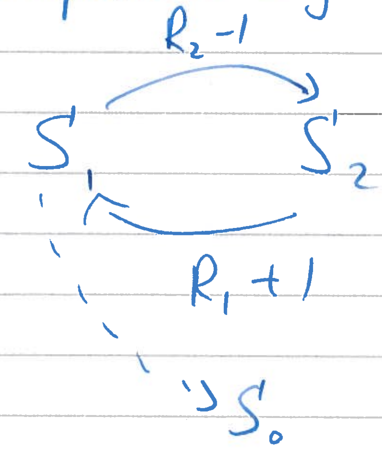
- successor function:

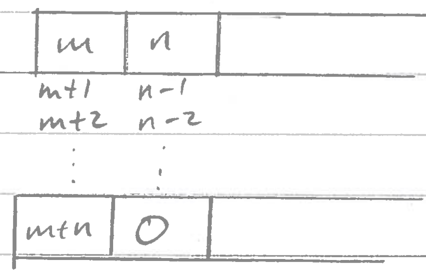


- The addition function $f : \mathbb{N}_0^2 \longrightarrow \mathbb{N}_0$

$$(m,n) \longrightarrow m+n.$$

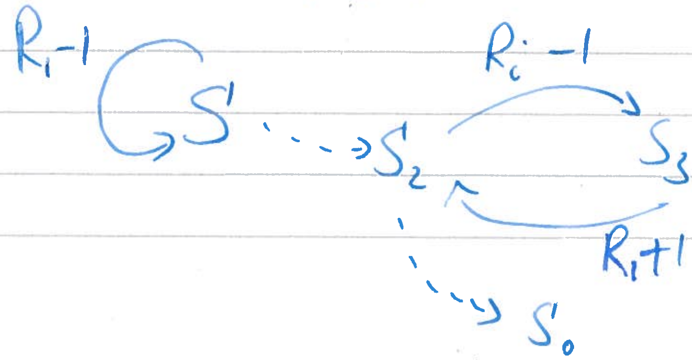
is also computable e.g. via the register function:

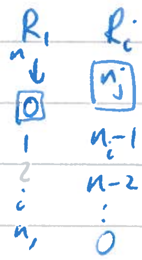




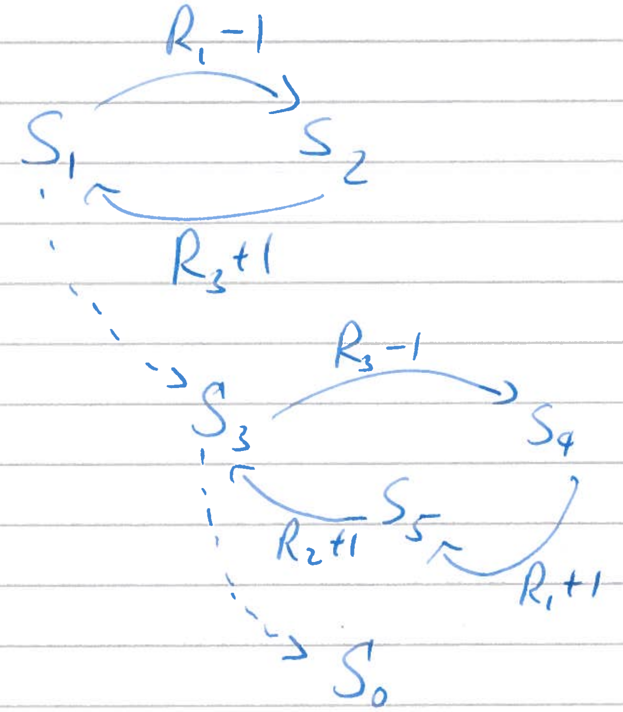
Note that such a register machine is "present" inside the projection function register machine from last time:

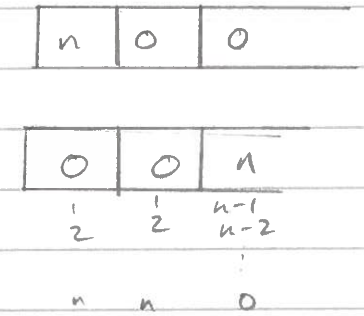
$f(n_1, \dots, n_k) \mapsto n_i \quad i \neq 1$





Lets now show how we can <u>copy</u> a register entry. Suppose we start with $R_1 = n$, $R_2 = 0$ and we want to end with $R_1 = n$, $R_2 = n$.



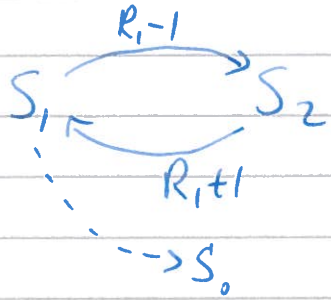


So we can express the "copy operation" in terms of a <u>register machine</u>.

Some reqsister machine (at least given certain inputs) may lead to processes that do not terminate

e.g.: $[S_1 \circlearrowleft R_1 + 1]$

—/—



If this expresses some "f"

$f(0) = 0$

$f(n)$ is undefined for $n \geq 1$

The f given here is an example of a partial function

Definition: A map $f: \mathbb{N}_0^k \to \mathbb{N}_0$ is a partial function if f is defined on some subset A of $\mathbb{N}_0^k$.

i.e when restricted to $A \subset \mathbb{N}_0^k$, f becomes a function
There exists $A \subset \mathbb{N}_0^k$ such that $f|_A : A \to \mathbb{N}_0$ is a function.

e.g. $f: \mathbb{N}_0 \to \mathbb{N}_0$ st $f(0) = 0$.
$f(n)$ is undefined for $n > 0$.

$f: \mathbb{N}_0 \to \mathbb{N}_0$, $n \mapsto \sqrt{n}$ — f is only defined for square numbers.
$f(1) = 1$, $f(2)$ undefined, $f(3)$ undefined, $f(4) = 2$.

$f: \mathbb{N}_0 \to \mathbb{N}_0$, $n \mapsto n/2$ — f only defined on the subset of even numbers (in $\mathbb{N}_0$).

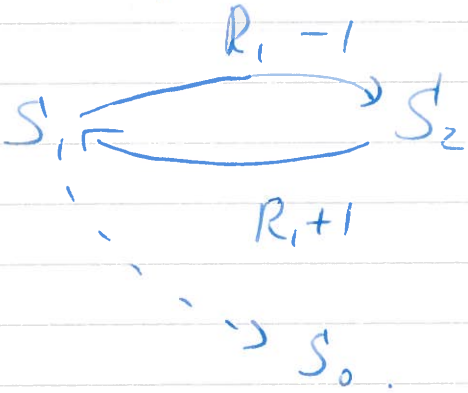
—/—

A partial function $f: \mathbb{N}_0^k \to \mathbb{N}_0$ is computable if there exists a register machine such that, when started with n_1 in $R_1, \ldots, n_k$ in R_k, and zero values in all remaining registers, the register machine:

- ends with $f(n_1, \ldots, n_k)$ in R_1, if $f(n_1, \ldots, n_k)$ is defined
- does not terminate, if $f(n_1, \ldots, n_k)$ is undefined.

e.g the partial function $f : \mathbb{N}_0 \to \mathbb{N}_0$
satisfying $f(0) = 0$
$f(n)$ is undefined for $n \geq 1$

is computable partial function, as shown earlier



We now describe some operations that can be performed on computable (partial) functions to yield other computable (partial) functions:

We start with composition:

Theorem: Let $f : \mathbb{N}_0^k \to \mathbb{N}_0$, $g_i : \mathbb{N}_0^l \to \mathbb{N}_0$ for $1 \leq i \leq k$ be computable (partial) functions. Then, the following is a computable (partial) function.

$$h : \mathbb{N}_0^l \to \mathbb{N}_0 \qquad (n_1, \ldots, n_l) = f(g_1(n_1, \ldots, n_l), \ldots, g_k(n_1, \ldots, n_l))$$

e.g. if $f : \mathbb{N}_0^2 \to \mathbb{N}_0$
$(m, n) \mapsto m+n$.

and $g_1 : \mathbb{N}_0 \mapsto \mathbb{N}_0$, $g_2 : \mathbb{N}_0 \to \mathbb{N}_0$
$n \mapsto n$ $\qquad$ $n \mapsto n^2$.

then the h in the theorem is defined as follows:
$h : \mathbb{N}_0 \to \mathbb{N}_0$.

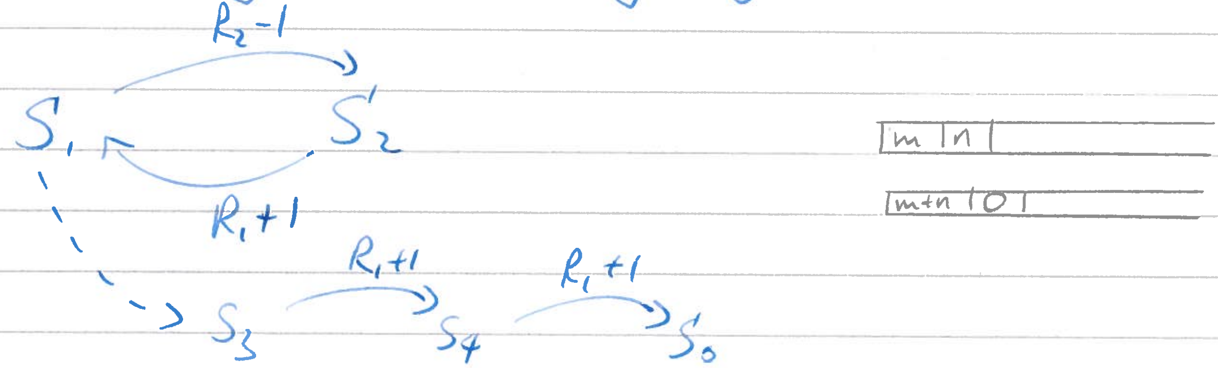
$$h(n) = f(g_1(n), g_2(n))$$
$$= f(n, n^2)$$
$$= n + n^2.$$

Explict example (using a register machine)

$f : \mathbb{N}_0^2 \to \mathbb{N}_0$, $(m,n) \to m+n$ $\qquad$ $g : \mathbb{N}_0 \to \mathbb{N}_0$, $n \longmapsto n+2$ $\qquad$ These are both computable

Then, consider $g \circ f : \mathbb{N}_0^2 \to \mathbb{N}_0$
$(m,n) \longmapsto (m+n)+2$.

The theorem says $g \circ f$ is computable, and we may verify this using the following register machine:



We now consider the process of <u>recursion</u>:

Let's use addition to define multiplication <u>recursively</u>:
Let $f : \mathbb{N}_0^2 \to \mathbb{N}_0$
$(m,n) \mapsto mn$.

$$f(m,0) = 0 \quad \text{computable (zero function)}$$

$$f(m,k+1) = f(m,k) + m \quad \text{computable (addition)}$$

In this recursive input we using the original input, "m".

Let's now define $f : \mathbb{N}_0 \to \mathbb{N}_0$ recursively
$n \mapsto n!$

$$f(0) = 1 \quad (\text{convention } 0! = 1)$$

<u>Recursive step</u>:

$$f(k+1) = (k+1)f(k). \quad \text{for } k \in \mathbb{N}_0$$

In this recursive step, what we do depends on the step itself, there is a dependency on k.

In the type of recursion we will encounter, <u>primitive recursion</u>, we will allow both the original input value(s) and the step (counter) itself to play a role in the recursion.

Crucially, recursion <u>respects</u> computability

Theorem: Let $f: \mathbb{N}_0^k \to \mathbb{N}_0$ and $g: \mathbb{N}_0^{k+2} \mapsto \mathbb{N}_0$ be computable (partial) functions. Then, applying primitive recursion to f and g gives a computable (partial) function, i.e. the following is computable:

$$h: \mathbb{N}_0^{k+1} \longrightarrow \mathbb{N}_0$$

$$h(n_1, \dots, n_k, 0) = f(n_1, \dots, n_k)$$
and for $m \in \mathbb{N}_0$.

$$h(n_1, \dots, n_k, m+1) = g(h(n_1, \dots, n_k, m), n_1, \dots, n_k, m)$$

Finally, the process of minimalisation:

Theorem: Let $f: \mathbb{N}_0^{k+1} \to \mathbb{N}$ be a computable (partial) function. Then the following is a computable, possibly partial, function:

$$g: \mathbb{N}_0^k \to \mathbb{N}_0$$

$$g(n_1, \dots, n_k) = n \text{ if } f(n_1, \dots, n_k, n) = 0.$$
$$f(n_1, \dots, n_k, m) > 0 \text{ for any } m < n.$$

$g(n_1, \dots, n_k)$ is undefined if there is no $n \in \mathbb{N}_0$ st $f(n_1, \dots, n_k, n) = 0$.

e.g consider $f: \mathbb{N}_0^2 \longrightarrow \mathbb{N}_0$.
$(m, n) \longmapsto m+n$.

Let's apply minimalisation to the "second" input:

$g(0) = 0$ since $f(0, \underline{0}) = 0$.

$g(3)$ is undefined since $f(3, n) \neq 0$ for any $n \in \mathbb{N}_0$.

So overall, g is the following computable function:

$$\left.\begin{array}{l} g: \mathbb{N}_0 \mapsto \mathbb{N}_0 . \\ g(0) = 0 . \\ g(m) \text{ undefined for any } m > 0 \end{array}\right\} \text{this is } \underline{\text{computable}} \text{ as shown earlier.}$$

Let's now see how to encode programs i.e. to find, for each program, we will try to find a unique identifier, in natural numbers.

Let's start by encoding single instructions. Suppose we are in state S_i, there are two possible instructions:

- $\xrightarrow{R_j + 1} S_k$ we encode this using e.g $2^j 3^k$
- $\xrightarrow{R_j - 1} S_k$ we encode this using $2^j 3^k 5^{l+1}$
  $\dashrightarrow S_l$.

Then e.g.:

$$\bullet \xrightarrow{R_4+1} S_7 \quad \text{encoded by} \quad 2^4 3^7$$

$$\begin{array}{l} \circ \xrightarrow{R_4+1} S_7 \quad \text{———} \quad 2^4 3^7 5^1 \\ \quad \dashrightarrow S_0 . \end{array}$$

Similarly: $180 = 2^2 \times 3^2 \times 5$
so 180 encodes

$$\begin{array}{l} \xrightarrow{R_2-1} S_2 \\ \dashrightarrow S_0' . \end{array}$$

defines $f : \mathbb{N}_0 \longrightarrow \mathbb{N}_0$

$$f(0) = 0 \ , \ f(n) = n-1 \text{ for } n > 0 .$$

How do we now encode whole program?
A program has a number of states, each associated to an instruction.

Use the following code/natural number.

$$2^{\text{code of state } S_1} , \ 3^{\text{code of state } S_2} \ \text{---} \ p_n^{\text{code of state } S_n}$$

n^{th} prime number.

<u>Note!</u> A program has a finite number of states, the above code will always be a well-defined natural numbers.

e.g let's encode "addition"

$$\begin{tikzcd} S'_1 \arrow[r, bend left, "R_2 - 1"] \arrow[dr, dashed] & S'_2 \arrow[l, bend left, "R_1 + 1"] \\ & S'_0 \end{tikzcd}$$

Code of instruction in S'_1:

$$\begin{tikzcd} \cdot \arrow[r, bend left, "R_2 - 1"] \arrow[dr, dashed] & S'_2 \\ & S'_0 \end{tikzcd} \quad \text{is} \quad 2^2 3^2 5^1$$

Code of instruction in S'_2: $\begin{tikzcd} S_1 & S_2 \arrow[l, bend left, "R_1+1"] \end{tikzcd}$ $2^1 3^1$

So the code of the whole program is $2^{180} 3^6$

12/12/12

Recursive (partial) functions

Definition: The set of recursive (partial) functions is defined (inductively) as follows:

1) Any zero function $f: \mathbb{N}_0^k \to \mathbb{N}_0$, $f(n_1, \ldots, n_k) = 0$, is recursive

2) The successor function $f: \mathbb{N}_0 \to \mathbb{N}_0$, $f(n) = n+1$, is recursive.

3) Any projection function $f: \mathbb{N}_0^k \to \mathbb{N}_0$ $f(n_1, \ldots, n_k) = n_i$, is recursive for $1 \leq i \leq k$.

4) Applying composition to recursive (partial) functions lead to a recursive (partial) function

5) Applying primitive recursion to recursive (partial) functions lead to a recursive (partial) function.

6) Applying minimalisation to a recursive (partial) function leads to a recursive, possibly, partial, function.

—/—

Note that the functions described in parts (1) to (3) are all examples of computable functions (as shown earlier), and the processes described in parts (4) to (6) take computable (partial) functions to computable, possibly partial, functions (as shown earlier). So, we deduce that:

Any recursive (partial) function is a computable (partial) function.

Examples of recursive functions:

1) The constant function $f_1 : \mathbb{N}_0^{\,k} \to \mathbb{N}_0$, $f_1(n_1, \dots, n_k) = n$ is recursive.

e.g f_1 may be obtained as the composition of a zero function and n successor functions.

2) The addition function $f_2 : \mathbb{N}_0^{\,2} \to \mathbb{N}_0$, $f(m, n) = m+n$, is recursive.

e.g We may apply primitive recursion, and use the projection and successor functions

$$f_2(m, 0) = m \quad \text{and} \quad f_2(m, k+1) = f_2(m, k) + 1$$

projection

successor function

3) The <u>multiplication</u> function $f_3 : \mathbb{N}_0^2 \to \mathbb{N}_0$, $f_3(m,n) = mn$, is recursive

e.g. may apply primitive recursion, and use the zero function, and the addition function (f_2):

$$f_3(m,0) = 0, \quad f_3(m,k+1) = f_3(m,k) + m$$
$$= f_2(f_3(m,k), m).$$

4) The <u>exponention</u> function $f_4 : \mathbb{N}_0^2 \to \mathbb{N}_0$, $f_4(m,n) = m^n$, is recursive.

e.g. may apply primitive recursion, and we the constant and multiplication functions (f_2, f_3)

$$f_4(m,0) = 1, \quad f_4(m,k+1) = m f_4(m,k)$$
$(m^{k+1} = m m^k)$
$$= f_3(m, f_4(m,k))$$

We may similarly construct versions of "subtraction" and "division" as recursive (partial) functions

—/—

Using the way of encoding computable (partial) functions, described earlier, it is possible to show that:

Any <u>computable</u> (partial) function is <u>recursive</u>.

So, overall, the notions of recursive and computable (partial) functions coincide.

— (—

Using this, and the encoding of computable functions described earlier, we may deduce that there is a of all recursive (partial) functions (the set of such objects is countable).

$f_0, f_1, f_2, f_3, \dots, f_n$.

where $f_n = \begin{cases} \text{the (partial) function encoded by } n, & \text{if } n \text{ is the code of a (partial) function} \\ \text{undefined} & \text{otherwise.} \end{cases}$

e.g. $f: \mathbb{N}_0 \to \mathbb{N}_0$, $n \mapsto n+1$ express using $S_1 \xrightarrow{R_1 + 1} S_0$

code of instruction : $2^1 3^0 = 2$

code of function : $2^2 = 4$

So, f_4 is the successor function.

Let's now use the list, to construct a non-recursive (partial) function:

Proposition: Consider $g: \mathbb{N}_0 \to \mathbb{N}_0$ defined as follows

$$g(n) = \begin{cases} f_n(n)+1 & \text{if } f_n \text{ is defined.} \\ 0 & \text{if } f_n \text{ is undefined.} \end{cases}$$

Then g, is not recursive.

Proof (by contradiction): Suppose that g is recursive. Then, since g is also defined everywhere, it must correspond to some recursive function on the list, f_m say, which is defined everywhere.

But then $g(m) = f_m(m)+1$

$$\neq f_m(m).$$

So, g and f_m take different values at m, i.e they cannot be the same function.

So, g is not recursive $\square$

If g was recursive, it would give a recursive way of deciding which recursive functions are defined where.

This is relevant to the halting problem, the problem of determining whether or not a given resisgter machine, given certain input, will terminate.

There is no recursive way of deciding the halt problem.

If we "place" this problem in a first order predicate setting, we can show that there is no <u>decidability theorem</u> for first order predicate logic.

—/—