3801 Logic Notes

Based on the 2012 autumn lectures by Dr I Strouthos

1/10/2010

logic lecture 1+2

Office hours: Room 712 Mondays 16:15 – 17:00

Wednesday 11:00 – 13:00

Thursday 17:15 – 18:00

Information www.ucl.ac.uk/~ucahist

Chapter 0: Preliminary notions

Countability.

'A set is countable if we are able to count it.'

Definition:

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~~ 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 $\mathbb{Z}$, integers, is countable; 'not by counting' $0,1,2,3,\ldots,-1,-2,-3,\ldots$

'valid pairing' $0,1,-1,2,-2,3,-3,$

4) The set of pairs of natural numbers $\mathbb{N}\times\mathbb{N}$ is countable

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

Possible 'valid counting': $\underbrace{(1,1)}_{\text{add up to 2}}, \underbrace{(1,2)(2,1)}_{\text{add up to 3}}, \underbrace{(1,3)(2,2)(3,1)}_{\text{add up to 4}}$

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

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

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

$3^3 2^2 \neq 3^1 \times 5^4$

Then f is injective

②

5) The set $\mathbb{Q}$, of rational numbers, is countable:

for example, we could imagine $\mathbb{Q}$ as lying inside $\mathbb{N}\times\mathbb{N}$, by sending $\frac{a}{b} \to (a,b)$

$0, \frac{1}{1}, -\frac{1}{1}, \frac{1}{2}, -\frac{1}{2}, \frac{2}{1}, -\frac{2}{1}, \frac{1}{3}, \cancel{\frac{2}{2}}, -\frac{3}{1},$

ignore repeats

logic - lecture 2

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

We will use 'Cantor's diagonal argument'

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

decimal pt

e.g $0.\boxed{1}\ 3\ 5\ 2\ 4 \ldots$

$0.2\ \boxed{5}\ 6\ 0\ 0 \ldots$

$0.1\ 7\ 8\ 9\ 2 \ldots$

$0.1\ 2\ 3\ 4\ 5 \ldots$

$\vdots$

let's produce a real number which cannot be on this list (so the list cannot include all real numbers between 0 and 1)

Define the number $s = 0.s_1 s_2 s_3 s_4$

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

$s_i = 2$ if ~ is equal to 5

eg in this example $s = 0.5255\ldots$

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

Overview of the course

1) Language
2) Proposition
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 $x = \pm 1$

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

$\boxed{x \cdot x = 1 \Rightarrow x = +1 \vee x = -1}$

'$\vee$' signifies 'or'

Our language will contain 'unknowns' or 'variables' like x, y, a, b, c and special 'connecting symbols' like $\Rightarrow$, $\vee$ and also operations like multiplication

③

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

e.g if we 'know' A

and we 'know' ~~B~~ $A \Rightarrow B$

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 symbols / ideas where '0' will define 'false'

~~and~~ '1' will define 'true'

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

if A or first assumption is false but taken to be true, then implication is true - or something false implies anything

something ~~true~~ true is implied by anything

logic - lecture 3

Conclusion of general description:

Consider the 'deduction' : If we have A, and we have $A \Rightarrow B$, then we have B

Is this a valid description? Yes (syntactic)

Is B necessarily true? That depends on whether A is true and whether or not A really implies B

syntactic aspects of logic: deal with the structure / validity of arguments

semantic " —— " : deal with the truth of logical statements (in certain settings)

We will first describe a language that can deal with statements such as:

$x^2 = 1 \Rightarrow x = +1$ or $x = -1$ So we will need to use:
- variables
- relations such as '='
- function such as 'squaring'
- connectives, such as $\Rightarrow, \vee, \wedge$ (or, and)

Actually we don't need to use '$\Rightarrow$' and '$\vee$' and '$\wedge$', because we can get each of these from a single one.

e.g

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

lets also '$\neg$' for negation

A	$\neg A$
0	1
1	0

consider $A \Rightarrow \neg B$

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

If we take $\neg(A \Rightarrow \neg B)$ we get the final column as $A \wedge B$

consider $\neg A \Rightarrow B$

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 $\neg A \Rightarrow B$ means the same as $A \vee B$

Chapter 1 Language

In order to be able to study the structure of mathematical objects and ideas, we will first describe a language / setting in which these objects can be defined and analysed.

The symbols we will use in our language consists of:

1) A countably infinite set of variable ~~symbols~~ symbols, e.g $\{x_1, x_2, x_3, \cdots\}$ or $\{x, y, z, x', y', z', \ldots\}$
2) For each non-negative integer, a countably infinite set of predicate symbols e.g $\{P_1, P_2, P_3, \ldots\}$ or $\{P, Q, R, P', Q', R', \ldots\}$ each of which has arity n. If P is a predicate symbol of arity n, then ~~which~~ we call P an n-ary predicate (symbol)
3) The symbols $\Rightarrow, \neg, \forall$

The set of all strings of symbols in our language will be denoted by $\mathcal{L}_{string}$

e.g for a 1-ary predicate P, a 2-ary predicate Q, and x, y, z, x_1, x_2, x_3 variable symbols: ~~x~~

x, Px, Pxy, Qxy, $Qx_1x_2x_3$, xQ, Qxx, $\neg Px$, $P\neg x$, $P \Rightarrow Q$, $P \Rightarrow PxQxy$, $\forall xPx$, $\forall x$, $\forall \neg \Rightarrow x \neg \forall$ are all strings

Lets now describe the subset of $\mathcal{L}$ string that will be particularly useful to us:

Definition The set of formulae in the first order predicate language (is a subset of $\mathcal{L}_{string}$) denoted by $\mathcal{L}$.

1) If P is a predicate symbol of arity n, and $x_1, \ldots, x_n$ are variable symbols, then $P_{x_1, \ldots x_n}$ is a formula.
2) If α is a formula, then $\neg\alpha$ is also a formula
3) If α, β are formulae then ~~α~~ $\Rightarrow \alpha\beta$ is also a formula.
4) If α is a formula, and x is a variable symbol, then $\forall x\, \alpha$ is a formula.

~~(using variable symbols x, y, z, x_1, x_2, x_3)~~

For example: Px, Qxy, Qxx, $\neg Px$, $\Rightarrow PxQxy$, $\forall xPx$,

while x, Pxy, $Qx_1x_2x_3$, xQ, $P\neg x$, $P \Rightarrow Q$, $\forall x$, $\forall \neg \Rightarrow x \neg \forall$

(using variable symbols x, y, z, x_1, x_2, x_3), a 1-ary predicate P and a 2-ary predicate Q)

logic-lecture 4+5 08/10/2012

Notes:

1) Formulae, particularly 'simple' ones of the form $Px_1 \dots x_n$ (for an n-ary predicate P) are sometimes referred to as propositional functions

2) We often refer to 1-ary and 2-ary predicates as unary and binary predicates respectively

3) Our language is ~~also~~ described of the first order predicate language because it allows us to deal with statements of the form.

"For each real number x, ..."

but not statements of the form.

"For each subset of real numbers ..."

This still ~~[illegible]~~ allows us to 'encode' lots of mathematics, but it is also a deficiency, in some sense, as we will (hopefully) see at the end of chapter 3.

The notion of 'degree' allows us to determine the complexity of formulae.

Definition The degree of a formula α denoted by $\deg(\alpha)$, is the non-negative integer obtained by (starting from 0) and adding:

1) 1, each time a '$\neg$' symbol appears in α

2) 2, " ——— '$\Rightarrow$' " ——— "

3) 1, " ——— "'$\forall$'" ——— "

For example (if P is an unary predicate, Q, is a binary predicate, x, y are variables)

$\deg(Px)=0$, $\deg(\neg Px)=1$, $\deg(\forall x Qxy)=1$, $\deg(\Rightarrow Px Qxy)=2$

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

Note: Formulae of degree 0 are precisely formulae of the form $Rx_1 \dots x_n$ (for an n-ary predicate R, and variables $x_1, \dots x_n$)

- The degree counts the number of 'substructures' in a formula
- The degree provides a kind of order to the set of formulae, $\mathcal{L}$, and helps us prove results about the whole of $\mathcal{L}$.

There are two notable absentees from our language: the left + right brackets '(' and ')'

Without these, mathematical statements are often ambiguous e.g. in 'usual' mathematical language

$\alpha \Rightarrow \beta \Rightarrow \gamma$ would mean $(\alpha \Rightarrow \beta) \Rightarrow \gamma$ or $\alpha \Rightarrow (\beta \Rightarrow \gamma)$

This is not a problem in $\mathcal{L}$

'$(\alpha \Rightarrow \beta) \Rightarrow \gamma$ is written as $\Rightarrow \Rightarrow \alpha \beta \gamma$

$\alpha \Rightarrow (\beta \Rightarrow \gamma)$ is written as $\Rightarrow \alpha \Rightarrow \beta \gamma$

It is, in general, true, that there is no real need for bracket in $\mathcal{L}$, to show this we first introduce the idea of 'weight'

<u>Definition</u> The weight of a string α denoted by weight (α) is the integer obtained by (starting from 0) and adding:

1) -1 each time a variable symbol appears in α
2) $n-1$ each time an n-ary predicate symbol appears in α
3) 0 ——— a '$\neg$' ———
4) $+1$ ——— '$\Rightarrow$' ———
5) $+1$ ——— '$\forall$' ———

For example, if x, y are variables, P is a unary predicate and Q is a binary predicate then:

weight $(Px) = 0-1 = -1$, weight $(\forall x\, Qxy) = -1$, weight $(\neg \Rightarrow \forall x y x) = -1$

weight $(Pxy) = -2$ weight $(\Rightarrow \forall x Px \neg Qxy) = -1$

weight $(\forall x Q) = 1$, weight $(\forall x) = 0$

in general, every formula has weight -1:

<u>Proposition</u>: Let α be a formula (i.e. $\alpha \in \mathcal{L}$). Then weight$(\alpha) = -1$

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

Suppose that $\deg(\alpha) = 0$. Then α has the form $Px_1, \ldots x_n$ (for any n-ary predicate P, and variables $x_1, \ldots, x_n$)

Therefore: weight (α) = weight$(Px_1, \ldots x_n) = (n-1) - n = -1$ as required

So the result holds for all formulae of degree 0

Let us now assume that α is a formula of degree $n+1$, and that the result holds for all formulae of degree smaller than or equal to n.

Consider α s.t. $\deg(\alpha) = n+1$. By construction/definition of $\mathcal{L}$, we have the following possibilities for α:

1) $\alpha = \neg \alpha_1$

2) ~~[illegible]~~ $\alpha = \Rightarrow \alpha_1 \alpha_2$

3) $\alpha' = \forall x \alpha_1$

Proposition. If $\alpha \in \mathcal{L}$, then weight$(\alpha) = -1$

Proof let's prove this by induction on the degree of α

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

in this case: weight(α) = weight$(Px_1, \dots, x_n) = (n-1) - n = -1$ as required

Suppose now, that the result holds for any formula of degree smaller than or equal to n. Consider a formula α of degree $n+1$.

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: $\deg(\alpha) = \deg(\neg\alpha_1)$

ie $n+1 = 1 + \deg(\alpha_1)$ So $\deg(\alpha_1) = n$

Since $\deg(\alpha_1) = n$, we may use the inductive hypothesis to deduce that weight$(\alpha_1) = -1$

Then weight(α) = weight$(\neg\alpha_1)$ = weight$(\neg)$ + weight(α_1)

$= 0 + (-1)$

$= -1$

So weight$(\alpha) = -1$

2) α is of the form $\Rightarrow\alpha_1\alpha_2$ for formula α_1, α_2

Then $\deg(\alpha) = n+1 = \deg(\Rightarrow\alpha_1\alpha_2) = 2 + \deg(\alpha_1) + \deg(\alpha_2)$

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

So inductively, we may assume that weight$(\alpha_1) = -1$, weight$(\alpha_2) = -1$

Then weight(α) = weight$(\Rightarrow)$ + weight(α_1) + weight(α_2) $= +1 - 1 - 1 = -1$

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

Then: $\deg(\alpha_1) = \deg(\forall x\alpha_1) = 1 + \deg(\alpha_1)$. So $\deg(\alpha_1) = n$

So inductively, weight$(\alpha_1) = -1$

Then weight(α) = weight$(\forall x\alpha_1) = +1 - 1 - 1 = -1$

— Since we have shown that weight$(\alpha) = -1$ in each case, we have concluded the proof □

e.g α : $\Rightarrow P_x Q_{xy}$
then it could be that
$\beta \Rightarrow P$ $\gamma: x Q_{xy} \ldots$

Let's now prove a similar, related result:

Proposition: Let $\alpha \in \mathcal{L}$, such that α is the concatenation ~~by~~ $\beta\gamma$, where β, γ are (nonempty) string (i.e. the string α can be obtained by writing the string β followed on the right by γ)

Then weight$(\beta) \geq 0$

So, no proper initial segment of a formula is a formula.

Proof: By induction on deg(α)

If deg$(\alpha)=0$, then α is of the form $Px_1 \ldots x_n$ (for P n-ary predicate $x_1 \ldots x_n$ variables)

Then, the proper initial segment, β is of the form $Px_1 \ldots x_m$ for $m \lneq n$

So, weight(β) = weight $(Px_1 \ldots x_m) = (n-1)-m = (n-m)-1 \geq 0$ (since $n-m>0$)

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

Suppose that α is a formula of degree $n+1$

Then α must have one of the following forms:

1) $\neg\alpha_1$, for some $\alpha_1 \in \mathcal{L}$

Then as in the previous proof ... deg$(\alpha_1) = n$, so, inductively, we assume that the result holds for α_1: if ε is a proper initial segment of α_1, then weight$(\varepsilon) \geq 0$

Let β be a proper initial segment of α: it must have one of the following forms

Case 1 $\beta: \neg$ Then weight$(\beta) = 0$ (≥ 0)

Case 2 $\beta: \neg\varepsilon$ for ε a proper initial segment of α_1

Then weight $(\beta) = 0 +$ weight(ε) = weight$(\varepsilon) \geq 0$ by inductive assumption

2) $\Rightarrow\alpha_1\alpha_2$ for $\alpha_1, \alpha_2 \in \mathcal{L}$

Then, ... deg$(\alpha_1) < n$, deg $(\alpha_2) < n$, we may assume that the results holds for α_1, α_2

The possible forms of β are:

Case 1 $\beta: \Rightarrow$ weight$(\beta) = +1$

Case 2 $\beta: \Rightarrow\varepsilon$ for ε a proper initial segment of α_1: (so weight $(\varepsilon) \geq 0$)

weight (β) = weight $(\Rightarrow)$ + weight (ε) = $1 +$ weight $(\varepsilon) \geq 0$

Case 3 $\beta: \Rightarrow\alpha_1\varepsilon$ for ε a proper initial segment of α_2 (so weight $(\varepsilon) \geq 0$)

Then weight $(\beta) = +1 - 1 +$ weight (ε) = weight $(\varepsilon) \geq 0$

3) $\forall x\, \alpha_1$, for some $\alpha_1 \in \mathcal{L}$. Then $\deg(\alpha_1) = n$, so we may assume that the result holds for α_1

In this case, β must have one of the following forms:

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

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

Case 3 $\beta: \forall x \varepsilon$, where ε is proper initial segment of α_1

$$\begin{aligned} \text{weight}(\beta) &= \text{weight}(\forall) + \text{weight}(\alpha) + \text{weight}(\varepsilon) \\ &= 1 - 1 + \text{weight}(\varepsilon) \\ &= \text{weight}(\varepsilon) \geq 0 \text{ by assumption} \end{aligned}$$

so weigh $(\varepsilon) \geq$

We have shown that in every possible way / case, weight$(\beta) \geq 0$. This concludes the proof. $\square$

Logic lecture 7 and 8.

The last two propositions indicate that brackets are not needed in order to identify formulae, or formulae 'within' other formulae in $\mathcal{L}$. We may use the notion of weight to help us determine when we have reached a formula, or a formula 'within' a formula in general.

For example suppose we wish to understand $\Rightarrow \neg Px \forall x Qxy$ (P unary predicate, Q binary predicate, x, y variables)

- weight will 'become' -1 only at the end

$$\text{weight}(\Rightarrow) = 1$$
$$(\Rightarrow \neg) = 1$$
$$(\Rightarrow \neg P) = 1$$
$$(\Rightarrow \neg Px) = 0$$
$$(\Rightarrow \neg Px \forall) = 1 \Rightarrow (\Rightarrow \neg Px \forall x) = 0$$
$$(\Rightarrow \neg Px \forall Q) = 1$$
$$(\Rightarrow \neg Px \forall Qx) = 0$$
$$(\Rightarrow \neg Px \forall Qxy) = -1$$

- We start with a '$\Rightarrow$', so this must 'connect' two other formulae, how do we find them? Using the same propositions

$$\Rightarrow \boxed{\neg\ P\ x} \text{ formula } \boxed{\forall\ x\ Q\ x\ y} \quad \leadsto \quad (\neg Px) \Rightarrow (\forall x Qxy)$$

(weights for $\neg P x$: $0 \; 0$, 0, -1; weights for $\forall x Q x y$: 1, 0, 1, 0, -1)

The presence of this internal structure is a nice feature of $\mathcal{L}$, the set of formulae. However, it will be useful to introduce some conventions that allow us to write formulae in the way they commonly appear in mathematics, and use symbols as they are commonly used in mathematics.

For example

- we wish to write '$\alpha \Rightarrow \beta$' instead of $\Rightarrow \alpha\beta$
- we wish to use symbols for 'or' and 'and' directly.
- we wish to write, eg '$x=2$' instead of $=x2$
- we wish to use functions:

  in fact, we may obtain any function using predicates

  e.g suppose we wish to express the function that 'squares', which 'takes' x to x^2

we would define a binary predicate, Q say, such that Qxy will hold whenever $x^2 = y$, not hold if $x^2 \neq y$

Then $Q5\,25$ will be true
but $Q5\,20$ will be false

However, we wish to be able to refer to functions directly
So, let us now define a more 'workable' version of our language

Conditional functional first order ~~lang~~ predicate language

The symbols we will consist of:

- a countable infinite set of variable symbols e.g $\{x, y, z, \dots\}$
- for each non negative integer arity n, a countably infinite set of predicate symbols, e.g $\{P, Q, R, P', Q', R', \dots\}$
- for each non negative integer arity n, a countably infinite set of functional symbols e.g $\{F_1, F_2, F_3, \dots\}$
- the symbols $\neg, \Rightarrow, \forall, (,)$

Using these symbols, we can obtain the following set of formulae:

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

0) Each variable symbol is a variable
1) If $x_1, \dots, x_n$ are variables, and F is an n-ary functional, then $Fx_1 \dots x_n$ is a variable
2) If $x_1, \dots, x_n$ are variables, and P is an n-ary predicate, then $Px_1 \dots x_n$ is a formula.
3) If α is a formula then $\neg\alpha$ is also a formula.
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

$x{+}y \Rightarrow x{+}y{+}1$???
$x{=}y \Rightarrow x{+}1 = y{+}1$ ✓

Furthermore, in $\mathcal{L}_{math}$, we will use the following conventions,

- '$\alpha \Rightarrow \beta$' will denote $\Rightarrow\alpha\beta$
- '$\alpha \vee \beta$' will denote $(\neg\alpha) \Rightarrow \beta$ (or $\Rightarrow\neg\alpha\beta$ in $\mathcal{L}$)
- '$\alpha \wedge \beta$' will denote $\neg(\alpha \Rightarrow (\neg\beta))$ (or $\neg\Rightarrow\alpha\neg\beta$ in $\mathcal{L}$)
- '$\alpha \Leftrightarrow \beta$' will denote $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$ → $(\Rightarrow\alpha\beta) \wedge (\Rightarrow\beta\alpha)$
  $\neg\Rightarrow\ \Rightarrow\alpha\beta\neg\Rightarrow\beta\alpha$
- $\exists x\,\alpha$ will denote $(\neg\ \forall x\, \neg\alpha)$

if '=' denotes 'equals' and '$\leq$' denotes inclusion of subsets, then the set of all subsets of $\mathbb{N}$, is a poset.

let's write these 'rules' as ~~formula~~ formula in $\mathcal{L}$math.

Firstly, we use the predicate '=' and '$\leq$' so the set of predicates is $\Pi = \{=, \leq\}$

$$\underset{\text{arity } 2}{=} \quad \underset{\text{arity } 2}{\leq}$$

we use ~~the~~ no functionals, so the set of functionals is $\Omega = \emptyset$

In this setting, the following are the defining statements of posets translated into $\mathcal{L}$math,

1) $(\forall x)(x \leq x)$

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

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

lets also write these in $\mathcal{L}$:

1) $\forall x \leq x x$

2) $\forall x \forall y \neg \neg \Rightarrow \leq xy \neg \leq yx = xy$

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

$\sim \neg \Rightarrow \alpha \neg \beta$

Example (groups)

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

1) For each $x, y \in G$: $x \cdot y \in G$

2) There exists an element e in G such that $e \cdot x = x$ and $x \cdot e = x$, for all x in G

3) For each x in G, there exists y in G such that $x \cdot y = e$ and $y \cdot x = e$

4) For all x, y, z in G, $x \cdot (y \cdot z) = (x \cdot y) \cdot z$

Lets write these 'rules' as formulae in $\mathcal{L}$math

Firstly, we use the predicate '=' so $\Pi = \{=\}$

arity 2

functionals M, E, so $\Omega = \{M, E\}$

$$\underset{\text{arity } 2}{M} \quad \underset{\text{arity } 0}{E}$$

In this setting, the following are the defining statements of ~~word~~ groups translated into $\mathcal{L}$math.

1) —

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) $(\forall x)(\forall y)(\forall z)(M(x, M(y, z)) = M(M(x, y), z))$

In addition, for a binary predicate or functional symbol, Q say, we will allow ourselves to write down '$x Q y$' instead of Qxy

eg $x+y \xrightarrow{\text{instead of}} +xy$

$x=y \xrightarrow{\text{instead of}} =xy$

Similarly, for an n-ary predicate or functional P, we will allow ourselves to write down

$p(x_1, \ldots, x_n)$ instead of $Px_1 \ldots x_n$

$f(x,y) \overset{\text{instead}}{=} fxy$

~~$\exists x$ [illegible]~~

Notes:

1) In practise, when describing specific mathematical systems/objects, we will use Π to denote the set of predicates that appear
Ω ———— functionals ————

2) An n-ary predicate may be thought of as a complete mathematical statement, which is a relation between n 'things', and which may be true or false in various cases.
On the other hand, an n-ary functional refers to an 'operation' on n things, which gives out an answer, depending on the inputs.

3) A 0-ary predicate denotes something that doesn't take in inputs, but is true or false in a given ~~[illegible]~~ situation (eg see chapter 2)
A 0-ary functional takes in no input, but gives out something as an output. Its output is independent of its input, it is ... constant.
We will often use 0-ary functionals to refer to distinguished constants
(eg the identity element in a group)

Let's now see some specific examples of mathematical theories, expressed in $\mathcal{L}$ and $\mathcal{L}$ math (this will also help us see how $\mathcal{L}$ and $\mathcal{L}$ math are different)

Example (posets)

A poset, or partially ordered set, consists of a set X, together with the two relations, $=$ and $\leq$, such that

1) For each x in X, $x \leq x$
2) For each x, y in X if $x \leq y$ and $y \geq x$ then $x = y$
3) For each x, y, z in X, if $x \leq y$ and $y \leq z$ then $x \leq z$

e.g if '$=$' denotes equality and '$\leq$' denotes smaller than or equal to, then the sets $\mathbb{N}, \mathbb{Z}, \mathbb{R}$ are posets.

The last few examples we have seen indicate that we may use our 'languages' (in the form of $\mathcal{L}$ or $\mathcal{L}_{math}$) in order to express mathematical statements related to a number of mathematical topics; such as posets, groups, rings, fields etc.

In the next chapter, however, we will introduce a simpler, 'reduced', version of the language, which helps us study the (interplay between the) notions of truth and ~~possibility~~ provability (this is an important part of 'modern' mathematical logic.)

Chapter 2: Propositional Logic

Here, we will work with a simpler version of our language, where 'we' ignore variables.

As a result, we will not use the $\forall, \exists$ symbols here, or predicates of arity greater than zero (or functionals)

The symbols we will use here consist of:

- A countably infinite set of primitive propositions, denoted by $\mathcal{L}_0^P$
- The symbols $\neg, \Rightarrow, (,)$

The primitive propositions are the 'building blocks' of the set of all the propositions

Definition:

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

1) Every primitive proposition is a proposition; 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 $\alpha \Rightarrow \beta$ is a proposition

Furthermore, we will adopt some of the relevant conventions of $\mathcal{L}_{math}$:

$\alpha \vee \beta$ will denote $(\neg\alpha) \Rightarrow \beta$

$\alpha \wedge \beta$ will denote $\neg(\alpha \Rightarrow (\neg\beta))$

$\alpha \Leftrightarrow \beta$ will denote $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

Notes:

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

2) The set of primitive proposition is countable.
~~As a result~~ So the set of propositions of degree 1 are countable.
degree 2 " ——— "
and so on.
Hence, the set of all propositions is also countable.

Semantic aspects of propositional logic

In this part, we will study the notion of 'truth' in the setting of propositions.

In order to determine the truth / falsehood of any proposition, we will start by ascribing truth to the primitive propositions, and then determining the truth / falsehood of other propositions, using 'sensible' rules:

Definition

A valuation is a function $v: \mathcal{L}_0 \to \{0,1\}$ which satisfies the following:

1) For a proposition α, $v(\neg\alpha) = 0$ if $v(\alpha)=1$
and $v(\neg\alpha) = 1$ if $v(\alpha)=0$
i.e. for any $\alpha \in \mathcal{L}_0$: $v(\neg\alpha) = 1 - v(\alpha)$

2) For propositions α, β, $v(\alpha \Rightarrow \beta) = 0$ if $v(\alpha)=1$ and $v(\beta)=0$
$v(\alpha \Rightarrow \beta) = 1$ otherwise

Note: We use '0' to denote the falsehood of a proposition
" —— " '1' " ——— " truth " ——— "

let's show that a valuation is determined by its values on the primitive propositions:

Proposition:

If v, v' are two valuations such that ~~....~~ $v(\alpha) = v'(\alpha)$ for all $\alpha \in \mathcal{L}_0^0$, then
primitive
$v(\alpha) = v'(\alpha)$ for all $\alpha \in \mathcal{L}_0$ → propositions.

17/10/2012

Proof:

lets use induction on the degree of a proposition.

let $\alpha \in L_0$. If $deg(\alpha)=0$, then α is a primitive proposition, so the result holds by assumption, $v(\alpha)=v'(\alpha)$

Now suppose that the result holds for all propositions of degree smaller than or equal to n.

Suppose that $deg(\alpha)=n+1$ $(\alpha \in L_0)$

Then α must have one of the following forms:

1) $\alpha = \neg\alpha$, for some $\alpha \in L_0$. Then $deg(\alpha) = 1 + deg(\alpha_1) = n+1$

So $deg(\alpha_1)=n$

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

Then by definition: $v(\alpha) = v(\neg\alpha_1) = 1 - v(\alpha_1)$, $v'(\alpha) = 1 - v'(\alpha_1)$; since $v(\alpha_1)=v'(\alpha_1)$, $v(\alpha)=v'(\alpha)$ as required

2) $\alpha = (\alpha_1 \Rightarrow \alpha_2)$ for $\alpha_1, \alpha_2 \in L_0$. Then $deg(\alpha_1) \leq n$, $deg(\alpha_2) < n$

So by inductive hypothesis: $v(\alpha_1) = v'(\alpha_1)$

$v(\alpha_2) = v'(\alpha_2)$

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

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

since v, v' agree on α_1, α_2, $v(\alpha)=v'(\alpha)$

So in either case, we have shown that $v(\alpha)=v'(\alpha)$

This concludes the proof.

logic - lecture 10+11 22/10/2012

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 $\alpha \in \mathcal{L}_0$)

i.e "if two valuations agree on that primitive ~~proposition~~ propositions, then they agree on all propositions"

Let's now give a related result

Proposition: 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

$$v(\alpha) = f(\alpha) \text{ for all } \alpha \in \mathcal{L}_0^P$$

Proof

By induction on the degree of a proposition

Lets 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 $\alpha_1, \alpha_2 \in \mathcal{L}_0$

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

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

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 ~~may~~ 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 table.

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 $v(\alpha \wedge \beta)=0$ 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 $v: \mathcal{L}_0 \to \{0,1\}$, we have $v(\alpha)=v(\beta)$ then we say that α and β are semantically equivalent

6

22/10/2012

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

$\alpha \Rightarrow \alpha$ is a tautology
truth table column
$\alpha \Rightarrow \alpha$ contains only ones

- $\neg\neg\alpha$

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

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 '$\alpha \vee \beta$', '$\beta \vee \alpha$' <u>mean</u> 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 tautology

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

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 tautology

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

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

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

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

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

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

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 $v(\alpha) = 0, v(\beta) = 1$ then $v((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)) = 0$

6

22/10/2021

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

lets 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 lets try to show this by contradiction

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

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 isn't 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 argument used in the previous ~~the~~ 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 <u>semantic</u> <u>tableaux</u> <u>method</u>

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

$$\alpha \vee \beta$$

$\alpha \qquad \beta$

'$\alpha \vee \beta$ 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

$$\alpha \wedge \beta$$

$$\alpha , \beta$$

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

$$\neg\neg\alpha$$

$$\alpha$$

$\alpha \vee \beta$: $\alpha \qquad \beta$

$\neg(\alpha \vee \beta)$: $\neg\alpha , \neg\beta$

$\alpha \wedge \beta$: α , β

$\neg(\alpha \wedge \beta)$: $\neg\alpha \qquad \neg\beta$

$\alpha \Rightarrow \beta$: $\neg\alpha \qquad \beta$

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

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

$\alpha \Leftrightarrow \beta$: $\alpha , \beta \qquad \neg\alpha , \neg\beta$

iff

$\neg(\alpha \Leftrightarrow \beta)$: $\alpha , \neg\beta \qquad \neg\alpha , \beta$

- 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 isn't a tautology, we wish to determine for which kinds of valuations α fails to be true)

propositional are $\neg, \Rightarrow$

Then, as in the proofs, from last time, we use the idea that α is a tautology precisely if $\underline{\neg\alpha}$ 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 $\underline{\neg\alpha}$
2) Breakdown $\neg\alpha$ into simpler and simpler proposition, using the basic rules of semantic tableaux (that we saw last time) until we are down to the 'indecomposable' 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 branch doesn't 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 'indecomposable' propositions of an open branch, we can describe a valuation v such that $v(\neg\alpha)=1$

i.e. such that $v(\alpha)=0$

Basic rules of propositional semantic tableaux

$$
\begin{array}{c} \neg\neg\alpha \\ | \\ \alpha \end{array}
\qquad
\begin{array}{c} \alpha\vee\beta \\ / \quad \backslash \\ \alpha \quad\; \beta \end{array}
\qquad
\begin{array}{c} \alpha\wedge\beta \\ | \\ \alpha,\beta \end{array}
\qquad
\begin{array}{c} \alpha\Rightarrow\beta \\ / \quad \backslash \\ \neg\alpha \quad\; \beta \end{array}
\qquad
\begin{array}{c} \alpha\Leftrightarrow\beta \\ / \quad \backslash \\ \alpha,\beta \quad\; \neg\alpha,\neg\beta \end{array}
$$

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

Examples of semantic tableaux

1) Is $\alpha \Rightarrow \alpha$ a tautology?

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

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

both α and $\neg\alpha$ are closed

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

2) $\alpha \Rightarrow \beta$

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 $v(\alpha) = \overset{\alpha}{1}$ and $v(\beta) = \overset{\neg\beta}{0}$

Since the open branch contains α and $\neg\beta$.

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

consider the tableaux for $\neg\neg(\alpha \Rightarrow \beta)$

$$\neg\neg(\alpha \Rightarrow \beta)$$
$$|$$
$$\alpha \Rightarrow \beta$$
$$/ \quad \backslash$$
$$\neg\alpha \qquad \beta$$

open branch open branch

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

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

4) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

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

24/10/2012

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

|

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

|

$\beta, \neg\alpha$

α don't breakdown
other breakdown

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

5) $(\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \alpha)$

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

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

|

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

/ \

$\neg\alpha$ $\quad$ β

| $\quad$ |

$\beta, \neg\alpha$ $\quad$ $\beta, \neg\alpha$

breakdown $\alpha \Rightarrow \beta$
then
$\neg(\beta \Rightarrow \alpha)$ on new line

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 s.t. $v(\alpha) = 0$ and $v(\beta) = 1$

Alternative tableaux: decompose '$\neg(\beta \Rightarrow \alpha)$' before '$\alpha \Rightarrow \beta$' (final conclusion is the same)

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

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

|

$\beta, \neg\alpha$

/ \

$\neg\alpha$ $\quad$ β

open $\quad$ open

same conclusion as earlier

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

consider a tableaux for

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

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

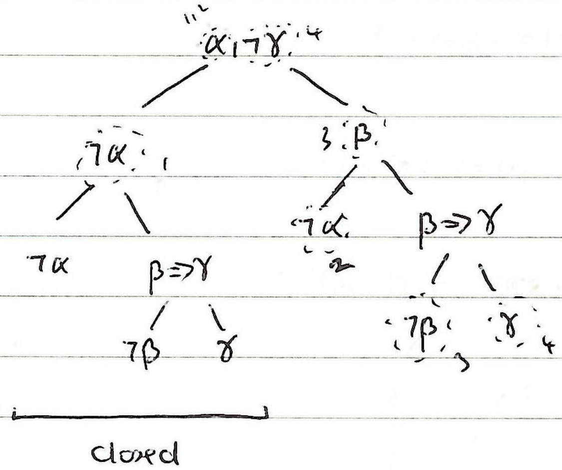
|

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

|

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

|



Every branch is closed:

$((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ is a tautology

Logic - lecture 13+14 29/10/2012

Some more examples of semantic ~~tableaux~~ tableaux

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

Consider a tableaux for the negation of the proposition:

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

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

$$(3)\ \alpha\vee\beta,\ \neg\gamma$$

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

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

closed (under the first γ) closed (under the second γ)

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

closed open open closed

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

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

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

or any valuation v, such that

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

---

is $\neg\alpha\Rightarrow\neg\beta\Rightarrow\neg\alpha\Rightarrow\beta\Rightarrow\alpha$ a tautology?

Consider the following tableaux

take negation of statement for a tableaux

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

$$\cdot\ (\neg\alpha)\Rightarrow(\neg\beta),\ \neg((\neg\alpha\Rightarrow\beta)\Rightarrow\alpha) \quad (2)$$

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

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

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

closed closed

$$\alpha$$

closed

Since every branch is closed, we deduce that

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

is a tautology.

Let's extend our notion of truth to include ideas such as: a proposition is true whenever some (other) propositions are true!

Definition:

Let S be a set of propositions ($S \subset \mathcal{L}_0$), and α be a propositions ~~are true~~! ($\alpha \in \mathcal{L}_0$)

Then, we say that S semantically implies or entails, 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 $S \models \alpha$

Note: if $\emptyset \models \alpha$, then α is always true.

i.e α is a tautology

We will often ~~#~~ simplify this to α is a tautology : $\models \alpha$

Examples

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

$\models (\alpha \Rightarrow \alpha)$

$\{\beta\} \models (\alpha \Rightarrow \beta)$ 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 $S \models \alpha$ holds by using the semantic tableaux method.

General idea: $S \models \alpha$ holds

$\Updownarrow$

for any valuation v s.t $v(s)=1$ $\forall s \in S$, $v(\alpha)=1$

$\Updownarrow$

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

$\Updownarrow$

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

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

Examples

- $\{\alpha, \alpha \Rightarrow \beta\} \models \beta$?

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

  $$\alpha, \alpha \Rightarrow \beta, \neg\beta$$
  $$/ \quad \backslash$$
  $$\neg\alpha \qquad \beta$$
  closed closed

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

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

  Consider the following tableaux

  $$\alpha \Rightarrow \beta, \beta \Rightarrow \gamma, \neg(\alpha \Rightarrow \gamma)$$
  $$|$$
  $$\alpha, \neg\gamma$$
  $$/ \quad \backslash$$
  $$\neg\alpha \qquad \beta$$
  $$/ \ \backslash \qquad / \ \backslash$$
  $$\neg\beta \quad \gamma \quad \neg\beta \quad \gamma$$
  closed closed closed closed

  Every branch is closed, so the semantic entailment holds.

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

  Consider the following tableaux:

  $$(\alpha \Rightarrow \beta), (\alpha \Rightarrow \gamma), \neg(\beta \Rightarrow \gamma)$$
  $$|$$
  $$\beta, \neg\gamma$$
  $$/ \quad \backslash$$
  $$\neg\alpha \qquad \beta$$
  $$/ \ \backslash \qquad / \ \backslash$$
  $$\neg\alpha \quad \gamma \quad \neg\alpha \quad \gamma$$
  open closed open closed

  Since there exist open branches we deduce that the 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 s.t. $v(\alpha) = 0, v(\beta) = 1, v(\gamma) = 0$

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

Syntactic aspects of propositional logic

We begin by seeing how we can define the notion of 'proof'

To construct proofs, we will use the following axiom:

Axiom 1 : $\alpha \Rightarrow (\beta \Rightarrow \alpha)$ for any $\alpha, \beta \in \mathcal{L}_0$

Axiom 2 : $(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ 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 $\alpha, \beta \in \mathcal{L}_0$

We will also use the following rule of deduction, known as modus ponens

If we have α and 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 $1 \le i \le n$, t_i is either:

- an (occurrence of an) axiom or
- an element of S (also known as a hypothesis) or
- is deduced by modus ponens from two preceding propositions.

  i.e. t_i is some proposition δ, and for some $j, k < i$

  t_j is some proposition γ and t_k is the proposition $\gamma \Rightarrow \delta$.

$$\begin{array}{ll} t_1 & \\ \vdots & \\ t_j & \gamma \\ t_k & \gamma \Rightarrow \delta \\ \vdots & \\ t_i & \delta \\ \vdots & \\ t_n & \alpha \end{array}$$

Notes:

1) All our axioms are instances of tautologies (have seen it already)
2) Each axiom given above corresponds to an infinite collection of propositions (they all 'work' for all propositions 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 can say that S syntactically implies or proves α, and write $S \vdash \alpha$

if there exists a proof of α from S

Note: If $\emptyset \vdash \alpha$, ie if we can prove α without using any hypotheses, 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 above 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. $((\neg\alpha) \Rightarrow (\neg\beta)) \Rightarrow ((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$ Axiom 3
4. $((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha$ modus ponens on lines 1, 3
5. α modus ponens on lines 2, 4

- lets 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 $\alpha \Rightarrow \alpha$ is a theorem, i.e. let's show that

$\vdash (\alpha \Rightarrow \alpha)$

We use the following proof :

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

The examples of proofs we have seen may indicate that even proofs of relatively simple results may appear to be complicated.
Lets now see a result that will help us in the setting of proofs.

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.
Lets first assume that $S \vdash (\alpha \Rightarrow \beta)$
i.e there is a sequence of propositions $t_1, \ldots, t_n$ 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:

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

$t_{n+1} : \alpha$ hypothesis (in $S \cup \{\alpha\}$)

$t_{n+2} : \beta$ modus ponens on lines t_n, t_{n+1}

Then, we obtain a proof of β from $S \cup \{\alpha\}$, as required, i.e $S \cup \{\alpha\} \vdash \beta$
Lets now assume that $S \cup \{\alpha\} \vdash \beta$ and lets consider a proof of β from $S \cup \{\alpha\}$, i.e a sequence of propositions $t_1, \ldots, t_m$, 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 of $\alpha \Rightarrow \beta$ 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, $1 \leq i \leq m$:
i) t_i is an axiom, we may still assume t_i, 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 validly replace t_i by $\alpha \Rightarrow t_i$

2) t_i is a hypothesis, in $S \cup \{\alpha\}$ → then it remains a hypothesis

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 be a hypothesis (it may not be 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 ~~have~~ need to use α as a hypothesis in this case.

idea:

$S \cup \{\alpha\} \vdash \beta$ $S \vdash (\alpha \Rightarrow \beta)$

$t_j: \gamma \rightsquigarrow \alpha \Rightarrow \gamma$

$t_k: \gamma \Rightarrow \delta \sim (\alpha \Rightarrow (\gamma \Rightarrow \delta))$

$t_i: \delta$ $\alpha \Rightarrow \delta$

3) t_i is some proposition, δ say, which is deduced by modus ponens on two ~~propositions~~ previous lines e.g say that for $j, k < i$, t_j is some $\gamma \in \mathcal{L}_0$ and t_k is $\gamma \Rightarrow \delta$

We may inductively assume that t_j and t_k have already been successfully replaced by $\alpha \Rightarrow t_j$ and $\alpha \Rightarrow t_k$

i.e by $\alpha \Rightarrow \gamma$ and $\alpha \Rightarrow (\gamma \Rightarrow \delta)$

Then the following sequence of lines will lead to $\alpha \Rightarrow \delta$

$$\begin{bmatrix} \alpha \Rightarrow \gamma \\ \alpha \Rightarrow (\gamma \Rightarrow \delta) \end{bmatrix}$$

should be γ

$(\alpha \Rightarrow (\gamma \Rightarrow \delta)) \Rightarrow ((\alpha \Rightarrow \delta) \Rightarrow (\alpha \Rightarrow \delta))$ 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 t_i by $\alpha \Rightarrow t_i$

So, in the proof of β from $S \cup \{\alpha\}$, we may replace each line t_i by $\alpha \Rightarrow t_i$ without using α as a hypothesis.

In particular, the last line $t_m: \beta$ may be successfully replaced by $\alpha \Rightarrow t_m$ i.e $\alpha \Rightarrow \beta$. So the above argument 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 □

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 the deduction theorem, it suffices to show that

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

31/10/2012

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 $\vdash (\neg\neg\alpha) \Rightarrow \alpha$

<u>Note</u> : in the <u>proofs</u> that follow, we will sometimes assume the validity of <u>theorems</u> 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

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

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

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

Note:

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

Similarly, 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 hypotheses and may write 'assumed' to justify the use of such an α.

Such a convention works well, 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 $\beta \in \mathcal{L}_0$

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

Proof:

Let's first suppose that $S \vdash \beta$. Then, any proof of β from S is also a proof of β from $S \cup \{\alpha\}$

i.e. $S \cup \{\alpha\} \vdash \beta$

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 occurence 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 □

Lets now give proofs of some theorems, using the Deduction Theorem

$\vdash (\neg\neg\alpha) \Rightarrow \alpha$

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. $(\neg\neg\neg\alpha \Rightarrow \neg\alpha) \Rightarrow ((\cancel{\neg\neg\alpha}\ \neg\neg\neg\alpha \Rightarrow \alpha) \Rightarrow \neg\neg\alpha)$ Axiom 3
3. $(\neg\neg\neg\alpha) \Rightarrow (\neg\alpha)$ 'theorem' 'assumed'
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

from before $\vdash (\neg\neg\alpha) \Rightarrow \alpha$
$\downarrow$
$(\neg\neg(\neg\alpha) \Rightarrow \neg\alpha)$

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

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. β 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):

Proposition

Let $S \subset \mathcal{L}_0$ and $\alpha, \beta \in \mathcal{L}_0$

Then

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

Proof:

We prove the two directions separately

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

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

Consider a valuation v s.t $v(s)=1$ $\forall s\in S$ and $v(\alpha)=1$ 12/11/2012

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 definition of a valuation: if $v(\beta)=0$ then $v(\alpha\Rightarrow\beta)=0$)

So, for such a valuation v, satisfying $v(s)=1$ $\forall s\in S$ 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 s.t $v(s)=1$ for all $s\in S$

We consider the two cases for $v(\alpha)$

1) $v(\alpha)=1$:

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) $v(\alpha)=0$:

Then by the definition of a valuation $v(\alpha\Rightarrow\beta)=1$ (irrespective of the value of $v(\beta)$)

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

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

Completeness theorem for propositional logic:

Here, we will try to show that semantic and syntactic implications are equivalent, in the sense that, for $S\subset\mathcal{L}_0$ and $\alpha\in\mathcal{L}_0$:

$$S\models\alpha \text{ if and only if } S\vdash\alpha \quad \leftarrow \text{Completeness theorem.}$$

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

Soundness theorem for propositional logic:

Let $S\subset\mathcal{L}_0$, and $\alpha\in\mathcal{L}_0$

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

Proof:

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 $1\le i\le n$, 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 this 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 require

So, in either case, $v(t_i)=1$. So $v(t_i)=1$ for each $1\leq i\leq n$

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' $S\models\alpha$ to a proof, a finite sequence of lines.

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

ALTERNATE PROOF

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

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

Then, since $S\models\alpha$: $\{s_1,\ldots s_n\}\models\alpha$

$$\left.\begin{array}{l}\{s_2\cdots s_n\}\models(s_1\Rightarrow\alpha)\\ \{s_3\ldots s_n\}\models s_2\Rightarrow(s_1\Rightarrow\alpha)\\ \quad\vdots\\ \models s_n\Rightarrow((s_{n-1}\Rightarrow(\ldots(s_2\Rightarrow(s_1\Rightarrow\alpha))\end{array}\right\}\text{ using our earlier proposition}$$

$\vdash s_n\Rightarrow(s_{n-1}\Rightarrow(\ldots(s_2\Rightarrow(s_1\Rightarrow\alpha))))$ since we assume all tautologies are axioms.

12/10/2012

$$\left.\begin{array}{l} \{s_n\} \vdash s_{n-1} \Rightarrow (\dots \Rightarrow (s_2 \Rightarrow (s_1 \Rightarrow \alpha))) \\ \{s_n, s_{n-1}\} \vdash s_{n-2} \Rightarrow \dots \\ \{s_1, \dots, s_n\} \vdash \alpha \\ \vdots \\ S \vdash \alpha \end{array}\right\}$$ by the Deduction theorem.

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

A key idea is that of consistency:

- A set of proposition S ($S \subset \mathcal{L}_0$) is consistent if there is no proposition α ($\in \mathcal{L}_0$) such that $S \vdash \alpha$ and $S \vdash (\neg\alpha)$

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

Proposition:

Let S be a consistent set of propositions. Then, for any $\alpha \in \mathcal{L}_0$, at least one of $S \cup \{\alpha\}$, $S \cup \{\neg\alpha\}$ is consistent.

logic - lecture 18 14/11/2012

__From last time:__ A set of proposition S is __consistent__ if there is __no__ proposition α s.t $S \vdash \alpha$ __and__ $S \vdash \neg\alpha$

Let's now show that we may 'extend' consistent sets indefinitely:

__Proposition__

Suppose that a set of propositions S ($S \subset \mathcal{L}_0$) is consistent. Then, for any proposition α, at least one of $S \cup \{\neg\alpha\}$, $S \cup \{\alpha\}$ is consistent.

__Proof__

Consider $\alpha \in \mathcal{L}_0$, 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 s.t

$$S \cup \{\neg\alpha\} \vdash \beta \text{ and } S \cup \{\neg\alpha\} \vdash (\neg\beta)$$

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 argument/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:

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

Therefore, $S \cup \{\alpha\}$ inherits consistency from S. If $S \cup \{\alpha\}$ were inconsistent, then for some $\gamma \in \mathcal{L}_0$:

~~Proposition~~ $S \cup \{\alpha\} \vdash \gamma$ and $S \cup \{\alpha\} \vdash \neg\gamma$

~~Suppose that~~ 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 a 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 $s \in S$

Let's try to construct such a function,

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

check online notes to make sure is right

Note that, as mentioned earlier, $\mathcal{L}_0$ is a countable set, 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) If $S \cup \{\neg\alpha_1\}$ is consistent, then set $S_1 = S \cup \{\neg\alpha_1\}$ and set $v(\alpha_1)=0$, $v(\neg\alpha_1)=1$

  In either case, S_1 is consistent

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

  If $S_1 \cup \{\alpha_2\}$ is consistent, then $S_2 = S_1 \cup \{\alpha_2\}$ 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 $S \subset S_1 \subset S_2 \subset S_3 \subset \dots \subset S_n \subset \dots$

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

$$S_m \vdash \beta \text{ and } S_n \vdash (\neg\beta)$$

Then, for some large enough number K (e.g. K = maximum of m and n) $S_K \vdash \beta$ 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.

~~Alternatively~~

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, $\bar{S} \vdash \beta$, i.e. $\bar{S} \vdash \beta$ and $\bar{S} \vdash (\neg\beta)$

i.e. $\bar{S}$ is inconsistent. This contradicts the consistency of $\bar{S}$ $\therefore$ $\bar{S}$ is deductively closed.

From last time

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

Proof so far (in brief): Try to construct 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.

- $v(s)=1$ for all $s \in S$ $\qquad \mathcal{L}_0 = \{\alpha_1, \alpha_2, \dots, \alpha_n, \dots\}$
- $v(\alpha_1)=1, v(\neg\alpha_1)=0$ if $\underbrace{S \cup \{\alpha_1\}}_{=S_1}$ consistent

  $v(\alpha_1)=0, v(\neg\alpha_1)=1$ otherwise (if $\underbrace{S \cup \{\neg\alpha_1\}}_{S_1}$ consistent)

  $\vdots$

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

  $v(\alpha_{n+1})=0, v(\neg\alpha_{n+1})=1$ otherwise

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

$\bar{S}$ is consistent

$\bar{S}$ is deductively closed (if $\bar{S} \vdash \alpha$ then $\alpha \in \bar{S}$)

Lets now complete the above proof.

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

- lets check that, for any $\alpha \in \mathcal{L}_0$:

  $v(\neg\alpha)=1$ if $v(\alpha)=0$ and ~~...~~ $v(\neg\alpha)=1$ if $v(\alpha)=1$

This is ensured by the 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 $v(\beta)=0$

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

Let's verify this:

$\Rightarrow$ 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) $\neg\alpha \Rightarrow (\alpha \Rightarrow \beta)$ is a theorem

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

→ Finally suppose that $v(\alpha) = 1$, $v(\beta) = 0$. We need to show that $v(\alpha \Rightarrow \beta) = 0$

and we argue by contradiction.

0	0	
1	0	
0	1	
1	1	

$\Rightarrow = 0$ matter pt

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 our assumption that $v(\beta) = 0$

So, it must be the case that $v(\alpha \Rightarrow \beta) = 0$

So our function $v: L_0 \to \{0,1\}$ is a valuation

Also by ~~contradiction~~ construction, $v(s) = 1$ for all $s \in S$

So, as required, v is a model of S. □

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

Adequacy Theorem:

Let $S \subset L_0$ and $\alpha \in L_0$

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

Proof

Proof: 19/11/2012

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 $v: \mathcal{L}_0 \to \{0,1\}$ s.t

$v(s)=1$ 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 □

By combining the Soundness and Adequacy theorems (for propositional logic), we obtain the Completeness theorem.

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.

$S \models \alpha$ if and only if $S \vdash \alpha$

We complete this chapter with two consequences of the Completedness Theorem

Compactness Theorem for propositional logic

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

Then there exists a finite subset of S, S' say, such that

$S' \models \alpha$

Proof

Suppose $S \models \alpha$

Then, by Completedness Theorem $S \vdash \alpha$

But a proof is a finite sequence of ~~proportions~~ 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 ~~th~~, such that $S' \vdash \alpha$
So, by completedness theorem it must be the case that $S' \models \alpha$ (and S' is finite) □

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

Proof
By the Completedness 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 □

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 include:
- 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 intepreted as true or false, e.g if F is a unary 'squaring' functional then ~~...~~ Fx ($=\ldots x^2$) is simply an expression

Furthermore, some formulae may be true and false in different cases.
every element has an inverse
$$(\forall x)(\exists y)((Fxy = E) \wedge (Fyx = E))$$

Then:

If we work in $\mathbb{N} \cup \{0\}$ and F represents addition

E —— zero ← then the statement is false

If we work in $\mathbb{Z}$, and F —— addition

E —— zero

then the statement is true

If we work in $\mathbb{Z}$, and F —— multiplication

E —— one

then the statement is false

So we need some 'structure', like $\mathbb{N}$ or $\mathbb{Z}$ above, to interpret 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: We often write
'U is an $\mathcal{L}(\Pi, \Omega)$-structure'
if U is the non-empty set 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 Ch. 1)

Let's now move towards 'interpreting' formulae in given structures.
We first try to use variables unambiguously:

Definition (s):

An occurrence of a variable x in a formula is free if the ~~...~~ 'x' is within the scope, the occurrences is bound (the 'x' in '$\forall x$' is also said to be bound)

e.g. if x, y, z are variables, P unary predicate
Q binary predicate

$\forall x Px$ — bound

$\forall x Qxy$ — bound, free

$\forall x Qxx$ — bound

$(\forall x Px) \vee Qxy$ — bound, free, free

A hopefully more familiar example is:

free $\int_0^x f(x)\,dx$ bound $\Bigg\}$ $\xrightarrow[\text{written as}]{\text{more 'properly'}}$ $\int_0^x f(t)\,dt$

From now on, we may assume that all are formulae are clean

ie that in a given formula, we cannot have both bound and free occurences of the same variable

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

So $(\forall x Px) \vee Qxy$
$\downarrow$
A hopefully more familiar example is $(\forall z Pz) \vee Qxy$

Using this convention, we may define a sentence as a formula with no free (occurences of) variables.

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

lets now consider some 'things' that will take values when we interpret them:

- The set of terms in a given first order predicate language is defined inductively as follows:

1) Each variable symbol is a term

2) Each 0-ary functional is a term

3) For each n-ary functional F, and terms $\alpha_1, \alpha_2, \dots \alpha_n$,

$F\alpha_1, \dots \alpha_n$ is a term

Logic - lecture 21

21/11/2012

From last time:

<u>sentence</u>: a formula containing no <u>free</u> variable

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

for a binary predicate '$\geq$' and variable x, y 0-ary functional '1'

Some other definitions

- The set of <u>terms</u> 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 $t_1, \ldots, t_n$ are terms then $Ft_1, \ldots, t_n$ is a term.

e.g if $\bar{0}, \bar{1}$ are 0-ary functionals, and x, y are variables (+ is a binary functional)

then $x, y, 0, 1, x+1, y+0, (x+y)+1$, ~~...~~ $(x+x)+y$, $x+y$, $1+1$ are terms

<u>Closed terms</u> are terms containing <u>no</u> variables

e.g in the examples 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 U be an $\mathcal{L}(\Pi, \Omega)$ is defined as follows:

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 $t_1, \ldots, t_n$ be closed.
   Then, the interpretation of $F(t_1, \ldots, t_n)$ is $F_U((t_1)_U, \ldots, (t_n)_U) \in U$.
2) Let P be an n-ary predicate symbol and $t_1, \ldots, t_n$ be closed terms.
   Then, the interpretation of $P(t_1, \ldots, t_n)$ is $[P(t_1, \ldots, t_n)]_U$ and $[P(t_1, \ldots, t_n)]_U = \begin{cases} 1 \\ 0 \end{cases} \ldots$

$$\begin{cases} 1 & \text{if } P_U((t_1)_U \ldots (t_n)_U) \text{ holds in } U \\ 0 & \text{if } P_U((t_1)_U, \ldots (t_n)_U) \text{ does not hold in } U \end{cases}$$

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

4) If α, β are sentences with intepretations α_U, β_U, then

$$(\alpha \Rightarrow \beta)_U = \begin{cases} 0 & \text{if } \alpha_U = 1 \text{ and } \beta_U = 0 \\ 1 & \text{otherwise} \end{cases}$$

5) if $(\forall x)\alpha$ is a sentence, then the interpretation $((\forall x)\alpha)_U$ is defined as follows:

Define a 0-ary predicate $\bar{u}$ for each u in U ~~[illegible]~~ $(\bar{u}_U = u)$. Then if $\alpha[\bar{u}/x]$ holds for each u in U, then we say that $(\forall x)\alpha$ is true $[(\forall x)\alpha]_U = 1$. Otherwise $(\forall x)\alpha$ is false then $[(\forall x)\alpha]_U = 0$

Example

let $\pi = \{=\}$, $\Omega = \{+, 0, 1\}$ x, y variables

$=$: binary; $+$: binary; $0, 1$: 0-ary

consider $(1+1)_U$ if $U = \mathbb{N}$: $(1+1)_{\mathbb{N}} = 1_{\mathbb{N}} +_{\mathbb{N}} 1_{\mathbb{N}} \dots \boxed{2_{\mathbb{N}}}$

~~if $U = \mathbb{Z}_2$: $(1+1)_{\mathbb{Z}_2} = 1_{\mathbb{Z}_2} +_{\mathbb{Z}_2} 1_{\mathbb{Z}_2}$~~

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

$\mathbb{Z}_2$: integers modulo 2

consider $(1+1=0)_U$, if $U = \mathbb{N}$: $(1+1=0)_{\mathbb{N}} = 0$ 'i.e $(1+1=0)_{\mathbb{N}}$ is false'

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

$((\forall x)(x \geq 1))_U$ Substitute every $u \in U$ for x into '$x \geq 1$' and check if '$x \geq 1$' is true

Notation if α is a formula $\alpha[t/x]$ is the formula obtained by replacing every (free) occurence of x in α by t (t may be any term)

$\mathbb{N} = \mathbb{A}$

let α be $x \geq 1$. Then $\alpha[y/x]$ is $y \geq 1$

$\alpha[0/x]$ $\quad 0 \geq 1$

$\alpha[1/x]$ $\quad 1 \geq 1$

e.g $[(\forall x)(x \geq 1)]_{\mathbb{N}} = 1$ $\quad (\bar{1} \geq 1)$ holds $(\bar{2} \geq 1)_U$ holds $(\bar{3} \geq 1)$ holds ...

$[(\forall x)(x \geq 1)]_{\mathbb{Z}} = 0$ e.g consider $-\bar{2}$ s.t $-\bar{2}_{\mathbb{Z}} = -2$ - the actual integer -2

$(-\bar{2} \geq 1)_{\mathbb{Z}}$ doesn't hold

21/11/2012

Notes:

1) Note all strings have interpretations.

eg for a variable x, 0-ary functional $\bar{1}$, binary functional $(+)$, $x, x+1$ have no interpretation.

2) In part (5) of the above definition, we needed to define an 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 ~~[illegible]~~ $\alpha \in \mathcal{L}(\Pi, \Omega)$ and an $\mathcal{L}(\Pi, \Omega)$-structure U

check in notes

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

or that U is a model of α

U models 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 model of S.

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

Definition

A theory T ~~[illegible]~~ in a specified $\mathcal{L}(\Pi, \Omega)$ first order predicate language is a set of sentences in $\mathcal{L}(\Pi, \Omega)$

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

A <u>sentence</u> is a formula that doesn't contain <u>free</u> variables

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

Consider a sentence α in some $\mathcal{L}(\Pi,\Omega)$

If, for ~~a sentence~~ some $\mathcal{L}(\Pi,\Omega)$-structure $\mathcal{U}$, $\alpha_{\mathcal{U}}=1$ (i.e. if the interpretation of α is 1 in $\mathcal{U}$)

we say that α is <u>true</u> in $\mathcal{U}$, or, equivalently, that α <u>holds</u> in $\mathcal{U}$.

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

Equivalently, if $\alpha_{\mathcal{U}}=1$ we say that $\mathcal{U}$ is a <u>model</u> of α ($\mathcal{U}$ <u>models</u> α)

if $\alpha_{\mathcal{U}}=1$ for all $\alpha\in T$, we say that $\mathcal{U}$ is a <u>model</u> of T ($\mathcal{U}$ <u>models</u> T)

e.g. $\mathbb{N}$ is a model for '$(\forall x)(x\geq 1)$'

$\mathbb{Z}$ is not a model for/of '$(\forall x)(x\geq 1)$'

<u>Example of theories</u>

We first note that for theories including the equality predicate '=', we will always add some sentences 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) $(\forall x)(\forall y)(\forall z)\ (((x=y)\wedge(y=z))\Rightarrow(x=z))$ 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 predicate '$\geq$'
and the binary functional '+'
together with equality

Then we will use the following sentences

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

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

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

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

1) Theory of groups

let $\Pi = \{=\}$ and $\Omega = \{\cdot, E\}$ identity

where '=' has arity 2

'·' has arity 2

'E' has arity 0

Then, the following theory has all groups as models:

- $(\forall x)(((x \cdot E) = x) \wedge ((E \cdot x) = x))$
- $(\forall x)(\exists y)(((x \cdot y) = E) \wedge ((y \cdot x) = E))$
- $(\forall x)(\forall y)(\forall z)(((x \cdot y) \cdot z) = (x \cdot (y \cdot z)))$

$$\left.\begin{array}{l}(\forall x)(x = x) \\ (\forall x)(\forall y)((x = y) \Rightarrow (y = x)) \\ (\forall x)(\forall y)(\forall z)(((x = y) \wedge (y = z)) \Rightarrow (x = z))\end{array}\right\} \text{basic equality relations.}$$

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

Possible problem:

we usually would like equality to only identify non-distinct elements of a structure.
Given a structure U, with distinct elements a and b, then we don't 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 normal models

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

examinable 2) Theory of posets

$\Pi = \{=, \leq\}$, $\Omega = \phi$

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

26/11/2012

∘ $(\forall x)(x=x)$

∘ $(\forall x)(\forall y)((x=y)\Rightarrow(y=x))$

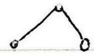
∘ $(\forall x)(\forall y)(\forall z)(((x=y)\wedge(y=z))\Rightarrow(x=z))$

∘ $(\forall x)(\forall y)(\forall z)((x=y)\Rightarrow((x\leq z)\Rightarrow(y\leq z)))$

∘ $(\forall x)(\forall y)(\forall z)((x=y)\Rightarrow((z\leq x)\Rightarrow(z\leq y)))$

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

3) A graph consists of a nonempty set of vertices and a (possibly empty) set of edges.

∘ An edge may connect two distinct vertices 

∘ An edge may not ~~man~~ connect a vertex with itself

∘ It is not necessary for two vertices to be connected by an edge

Let's define a theory of graphs: $\Pi=\{\sim\}$ binary predicate saying '$x\sim y$' if there's an edge between x and y

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

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

Let's now try to make theories more specific:

Eg: 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

$\Pi=\{=\}$, $\Omega=\{\circ, E, A_1, A_2\}$

arity 0

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

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

$\left.\begin{array}{l} \circ\ \neg(E=A_1) \\ \circ\ \neg(E=A_2) \\ \circ\ \neg(A_1=A_2)\end{array}\right\}$ so 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 model 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.

Syntactic aspects of first order predicate logic
To define ~~proofs~~ proofs in this setting, we use the following axioms
Axiom 1

$\alpha \Rightarrow (\beta \Rightarrow \alpha)$ for all formula α, β

Axiom 2

$(\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$ for all formulae α, β, γ

Axiom 3

$(\neg\alpha \Rightarrow \neg\beta) \Rightarrow (((\neg\alpha) \Rightarrow \beta) \Rightarrow \alpha)$ for all formulae α, β

Axiom 4

$$(\forall x)\alpha \Rightarrow \alpha\left[{}^{t}/_{x}\right]$$

for a variable x, formula α, and term t s.t no free variables of t appears bound in $(\forall x)\alpha$

eg $(\forall x)\overbrace{(x \geq y)}^{\alpha}$

$t = z \hookrightarrow z > y$ ✓

$t = 2 \hookrightarrow 2 > y$

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

Axiom 5

$$((\forall x)(\alpha \Rightarrow \beta)) \Rightarrow (\alpha \Rightarrow (\forall x)\beta)$$

assuming no free occurence of x in α

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

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

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 axiom 4 and 5

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

Set $t=z$ and use Axiom 4

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

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

constant (in o) Set $x=\bar{z}$

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

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

not allowed, since here $t=y$ and y is bound in α

$\alpha = (\exists y)(x \cdot y = E)$

As for Axiom 5, consider

$(\forall x)(y=1 \Rightarrow x+y=x+1)$

$\Downarrow$

$(y=1) \Rightarrow ((\forall x)(x+y=x+1))$ allowed

However, the following is not an instance of Axiom 5

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

$\Downarrow$

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

free variable | bound x's

$(x=0) \Rightarrow (\forall z)(y+z=y)$ ✗

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

i) <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 occurences of x appear in the premises/hypotheses used to obtain α

<u>Then</u>; given a set of formulae S ($S \subset \mathcal{L}$) and a formula α

A proof of α from S consists 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:

i) an axiom

2) an element of S (a hypothesis)

3) deduced by modus ponens on two earlier lines

i.e for $j, k < i$ $\quad t_j$ 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 $j < i$ t_j is a formula δ and t_i is the formula $(\forall x)\delta$

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

<u>Example of proofs</u>:

arity 2   arity 2

- Let T be a <u>theory</u> of groups, with $\Pi = \{=\}$ and $\Omega = \{\cdot, E\}$

arity 0

Lets 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

- Lets show that $\{T, z=E\} \vdash E=z$

1. ~~[illegible]~~ hypothesis $(\forall x)(\forall y)((x=y) \Rightarrow (y=x))$
2. ~~[illegible]~~ $(\forall x)(\forall y)((x=y) \Rightarrow (y=x)) \Rightarrow (\forall y)((z=y) \Rightarrow (y=z))$ 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 28/11/2012

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 occurence of z appeared in our hypotheses, 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 on the syntax of first order predicate logic:

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

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

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 a 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 $S \models \alpha$

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

Furthermore 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:

Theorem

Let S be a set of sentences in a first order predicate language

If S is consistent, then S has a model.

don't need to know proof for this version

This theorem then leads to:

Adequacy Theorem for first order predicate logic

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

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

Combining the soundness and adequacy theorems, we obtain completeness theorem for first order predicate logic

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

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

we may restate the completeness theorem as follows:

S has a model if and only if S is consistent

↓

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

let's consider some consequences of the completeness theorem

Compactness Theorem

Let S be a sentences in some first order predicate language, and α be a sentence in that language

Then, $S \vdash \alpha$ if and only if $S' \vdash \alpha$

where S' is a finite subset of S

Proof:

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 finite subset of S has a model, then so does S

Proof

Consider the set S, and assume that S doesn't have a model (i.e we will prove this result by contradiction) By the completeness theorem, we deduce that S is inconsistent, i.e there ~~are~~ is a sentence α in $\mathcal{L}(\Pi, \Omega)$ such that $S \vdash \alpha$ and $S \vdash (\neg\alpha)$

Since proofs are finite (sequences of formulae), proofs of $\neg\alpha, \alpha$ from S will use 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 03/12/2012

$$S' \cup S'' \vdash \alpha \text{ and } S' \cup S'' \vdash (\neg\alpha)$$

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

So, by the completeness Theorem, $S' \cup S''$ doesn't have a model as required

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 the 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 ~~Potpogy~~ T has arbitrarily large finite models (i.e such that, for any natural number $n \in \mathbb{N}$, there exists an $\mathcal{L}(\Pi,\Omega)$ structure U with at least n elements, which is a model of T)

Then, T also has an infinite model (countably infinite)

Proof

We construct an infinite model for T '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 $\Omega' = \Omega \cup \{c_1, c_2, \ldots, c_n, \ldots\}$

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

where $c_1, c_2, \ldots$ are functionals of arity 0

We now extend our theory to one where the constant are all distinct

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

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

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

Consider S a finite subset of T'

Since S is finite, it includes only finitely many of the constants $c_1, c_2, \ldots$ 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 model of T'

So by the Completeness 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 uncountably many symbols, we may similarly prove an 'uncountable' version of the above theorems:

<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>countably</u> <u>infinite</u> model.

Then T has an <u>uncountably</u> <u>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.

We set $\Pi = \{=\}$, $\Omega = \{1, s\}$

binary, arity 0, arity 1

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) $(\forall x)(\forall y)((s(x) = s(y)) \Rightarrow (x = y))$ injective

PA 6) $(\forall x)(\neg(s(x) = 1))$

PA 7) $(\forall y_1)(\forall y_2) \ldots (\forall y_n)\left(p[1/x] \wedge (p \Rightarrow p[s(x)/x]) \Rightarrow (\forall x) p\right)$

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

Notes: ∘ This 'weak' version of Peano arithmetic does not include 03/12/2012

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's arithmetic

∘ Note: The sentence PA7 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 'inductions')

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 PA7 twice as follows:

Firstly in PA7 set $y_1 = y$ 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 PA7 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 exists that has the natural numbers as the unique normal model.

logic - lecture 27 05/12/2012

Chapter 4: Computability

In this chapter we will study (ideas related to) computable and recursive (partial) function, 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 $(1 \le i \le n)$ 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 $R_j > 0$, subtracts 1 from R_j, and moves to state S_k
   - if $R_j = 0$, moves to state S_l

$$\begin{tikzcd} S_i \arrow[r, bend left, "R_j - 1"] \arrow[dr, dashed] & S_k \\ & S_l \end{tikzcd}$$

Examples of register machines:

Register:

R_1	R_2	R_3	R_4	
1	0	3	2	...

1) a register machine that adds 1 to R_1:

$$S_1 \xrightarrow{R_1+1} S_0$$

This changes our register to

R_1	R_2	R_3	R_4	
2	0	3	2	...

2) a register machine that adds 1 to R_j:

$$S_1 \xrightarrow{R_j+1} S_0$$

3) a register machine that adds 3 to R_2

$$S_1 \xrightarrow{R_2+1} S_2 \xrightarrow{R_2+1} S_3 \xrightarrow{R_2+1} S_0$$

4) a register machine that clears R_{20}

(ie that makes R_{20} the value at the end, whatever the original value is)

$$\begin{tikzcd} S_1 \arrow[loop above, "R_{20}-1"] \arrow[r, dashed] & S_0 \end{tikzcd}$$

So the register would look like

	R_{20}	
	3	

example

$R_{20}=3$

2

1

$\underline{\underline{0}}$ ← Stop

We will try to investigate the types of functions that can be expressed using register machines:

Definition

A function $f : \mathbb{N}_0^k \rightarrow \mathbb{N}_0$ is computable if there exists a register machine such that when started with n_1 in R_1, n_2 in R_2, n_k in R_k, and zero values in the remaining registers, the associated program ends (i.e reaches state S_0) with $f(n_1, n_2, \dots, n_k)$ in R_1

R_1	R_2	R_3	$\dots$	R_k	R_{k+1}	R_{k+2}
n_1	n_2	n_3		n_k	0	0

$\downarrow$ Register machine

R_1	
$f(n_1, n_2, \dots, n_k)$	$\dots$

Examples

1) The function $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ is computable
$$n \mapsto n+1$$

Below is a diagrammatic description of a register machine that computes f:

$$S_1 \xrightarrow{R_1 + 1} S_0$$

(By definition the register machine should work for any register so choose R_1)

2) The function $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$, $f(n) = n+3$ is computable

For example, via

$$S_1 \xrightarrow{R_1 + 1} S_2 \xrightarrow{R_1 + 1} S_3 \xrightarrow{R_1 + 1} S_0$$

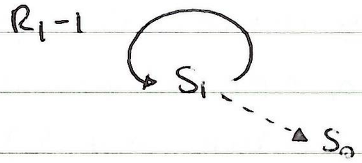
3) The function $f : \mathbb{N}_0^k \rightarrow \mathbb{N}_0$ (the zero function)
$$(n_1, n_2, \dots, n_k) \mapsto 0$$

n_1	n_2	n_3	$\dots$	n_k

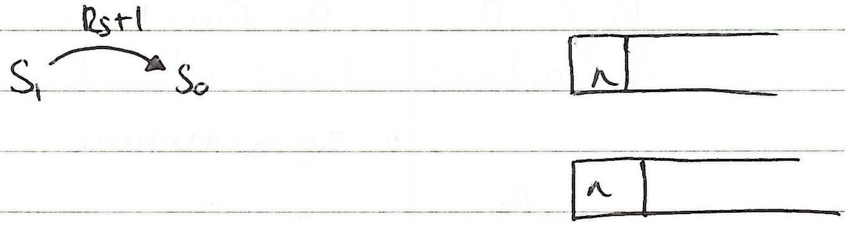
$\downarrow$

0	$\dots$

is computable, eg via



4) The identity function $f: \mathbb{N}_0 \to \mathbb{N}_0$, where $f(n) = n$, is computable, e.g via



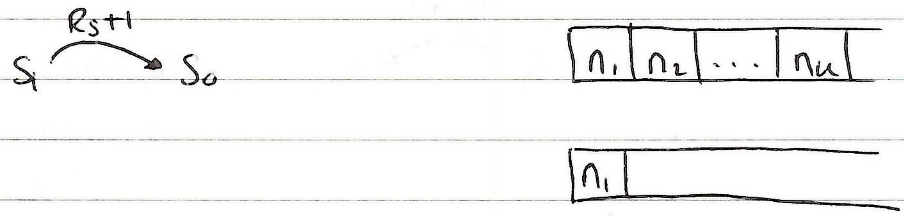
we don't want to change R_1, so change a different register

'use any program that leaves R_1 unaltered'

5) The projection function $f: \mathbb{N}_0^k \to \mathbb{N}_0$, defined via $f(n_1, n_2, \dots, n_k) = n_i$ for some $1 \le i \le k$ is computable

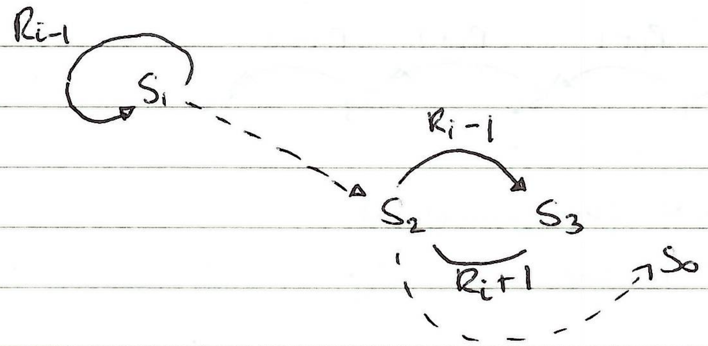
If $i = 1$, we need a program that 'doesn't change R_1'

e.g

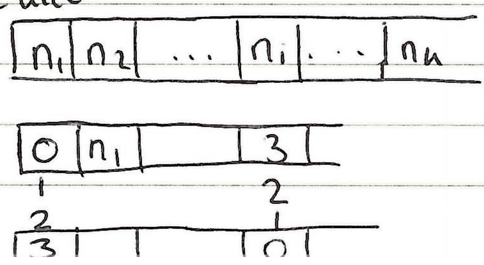


$\hookrightarrow f(n_1, n_2, \dots, n_k) = n_1$

If $i > 1$ we first clear R_1 and then add value of R_i to R_1



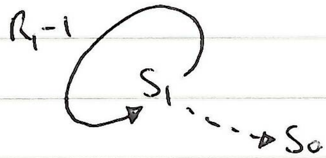
It would act like



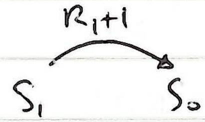
Logic - lectures 28+29 10/12/2012

Examples of computable functions

- zero function



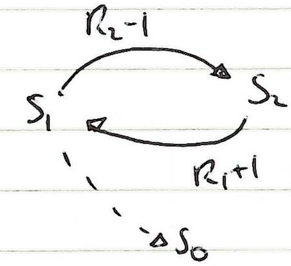
- Successor function



The addition function $f: \mathbb{N}_0^2 \to \mathbb{N}_0$

$(m,n) \longmapsto m+n$

is also computable, eg via the register machine



| m | n |

$m+1$ $n-1$

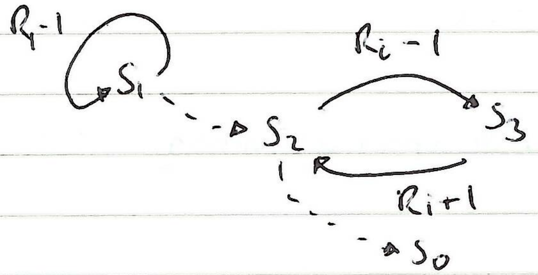
$m+2$ $n-2$

$m+3$ $n-3$

⋮

| m+n | 0 |

Note that such a register machine is "present" inside the projection function register machine from last time



$f(n_1, n_2, \ldots, n_k) \mapsto n_i$ $i \neq 1$

R_1 R_i

n_1 n_i

↓

0 n_i

1 $n_i - 1$

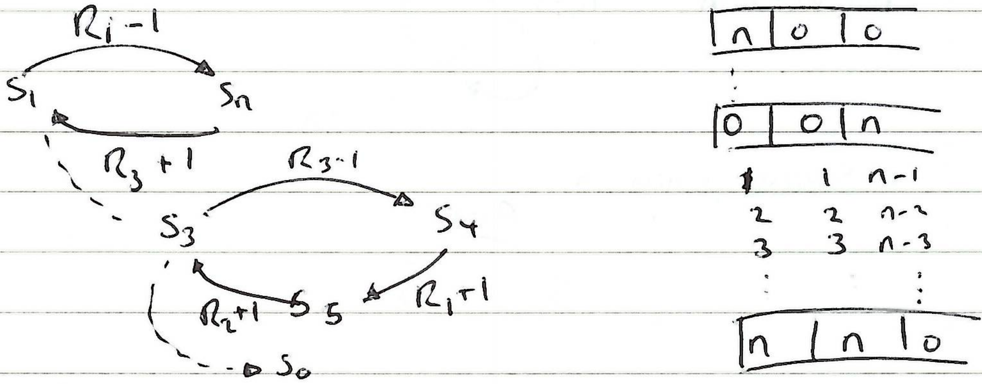
2 $n_i - 2$

⋮ ⋮

n_i 0

lets now show how we can copy a register entry.

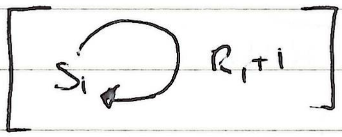
Suppose we start with $R_1=n$, $R_2=0$ and we want to end with $R_1=n, R_2=n$



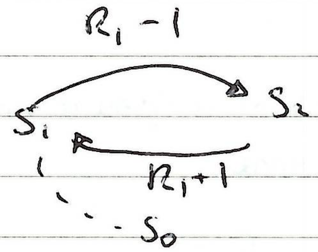
So we can express the 'copy operation' in terms of a register machine.

Some register machines (at least given certain inputs) may lead to processes that do not terminate:

For example:



or



If this ~~expression~~ 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 well 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 $f\big|_A : A\to\mathbb{N}_0$

is a function (f restricted to A only)

For example 10/12/2012

$f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ s.t $f(0)=0$

$f(n)$ is undefined for $n>0$

$f: \mathbb{N}_0 \Rightarrow \mathbb{N}_0$ f is only defined for square numbers

$n \mapsto \sqrt{n}$ $f(1)=1$, $f(2)$ undefined

$f(3)$ undefined , $f(4)=2$

$f: \mathbb{N}_0 \rightarrow \mathbb{N}_0$ f only defined on the subset of even numbers (in $\mathbb{N}_0$)

$n \mapsto \frac{n}{2}$

A partial ~~function~~ function $f: \mathbb{N}_0^k \Rightarrow \mathbb{N}_0$ is computable if there exists a register machine such that, when started with n_1 in R_1, $n_1 \ldots n_k$ 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
- doesn't terminate if $f(n_1, \ldots, n_k)$ is undefined

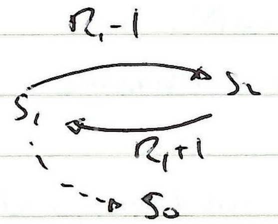
For example

The partial function $f: \mathbb{N}_0 \Rightarrow \mathbb{N}_0$ satisfying

$f(0)=0$

$f(n)$ is undefined for $n \geq 1$

is a 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 <u>composition</u>

not examinable

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.

$$h: \mathbb{N}_0^l \to \mathbb{N}_0 \quad (n_1, n_2, \dots, n_l) = f(g_1(n_1, \dots n_l), \dots, g_k(n_1, n_2, \dots n_l))$$

function does not always compose both ways (need to check)

e.g if $f: \mathbb{N}_0^2 \to \mathbb{N}_0$
$(m,n) \mapsto m+n$.

and $g_1: \mathbb{N}_0 \to \mathbb{N}_0$, $g_2: \mathbb{N}_0 \to \mathbb{N}_0$
$n \mapsto n$ $\quad$ $n \mapsto n^2$

Then the h^* is the theorem is defined as follows

$$h: \mathbb{N}_0 \to \mathbb{N}_0$$
$$h(n) = f(g_1(n), g_2(n))$$
$$= f(n, n^2)$$
$$= n + n^2$$

Explicit example (using a register machine)

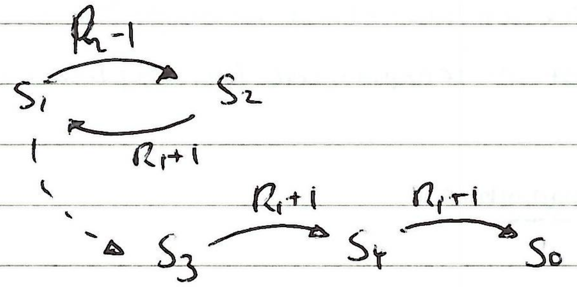
$f: \mathbb{N}_0^2 \to \mathbb{N}_0$ $\quad$ $g: \mathbb{N}_0 \to \mathbb{N}_0$
$(m,n) \mapsto m+n$ $\quad$ $n \mapsto n+2$

These are both computable

Then, consider $g \circ f = \mathbb{N}_0^2 \to \mathbb{N}_0$
$(m,n) \mapsto (m+n)+2$

The theorem says $g \circ f$ is computable, and we may verify this using the following register machine

m	n	



m+n	0
m+n+1	0
m+n+2	0

10/12/2012

We now consider the process of <u>recursion</u>

Lets use addition to derive multiplication

Let $f : \mathbb{N}_0^2 \longrightarrow \mathbb{N}_0$
$(m,n) \longmapsto mn$

To find $f(3,2)$

$f(3,0) = 0$
$f(3,1) = 0+3 = 3$
$f(3,2) = 3+3 = 6$

$f(m,0) = 0$ computable (zero function)

$f(m, k+1) = f(m,k) + m$ computable (addition)

In this recursive step, we are using the original input 'm'

10/12/2012

lets now define $f: \mathbb{N}_0 \to \mathbb{N}_0$ recursively
$n \mapsto n!$

Recursive step: $f(0) = 1$ (convention: $0! = 1$)
$f(k+1) = (k+1) f(k)$ for $k \in \mathbb{N}_0$

$f(0) = 1$
$f(1) = 1 \cdot 1$
$f(2) = 2 \cdot 1$
$f(3) = 3 \cdot 2$

In this recursive step, what we do is we are using the step itself.
i.e. there is a dependency on k.

In the type of recursion we will encounter, primitive recursion, we will allow both the original input value(s) and the step (counter) itself to play a role in the recursion.

Crucially, recursion respects computability

Theorem:
let $f: \mathbb{N}_0^k \to \mathbb{N}_0$ and $g: \mathbb{N}_0^{k+2} \to \mathbb{N}_0$ be computable (partial) functions. Then, applying primitive recursion to f and g gives a computable (partial) function
i.e. the following is computable.

$h: \mathbb{N}_0^{k+1} \to \mathbb{N}_0$

$h(n_1, \dots, n_k, 0) = f(n_1, \dots, n_k)$ 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 $f: \mathbb{N}_0^k \to \mathbb{N}$ be a computable (partial) function
Then, the following is a computable, possibly partial function.

$g: \mathbb{N}_0^k \to \mathbb{N}_0$
$g(n_1, \dots, n_k) = n$ if $f(n_1, \dots, n_k, n) = 0$
and $f(n_1, \dots, n_k, m) > 0$ for any $m < n$
$g(n_1, \dots, n_k)$ is undefined if there is no $n \in \mathbb{N}$ st $f(n_1, \dots, n_k, n) = 0$

eg consider $f: \mathbb{N}_0^2 \to \mathbb{N}_0$
$(m, n) \mapsto m+n$

let apply minimalisation to the 'second' input:

$g(0)=0$ since $f(0,0)=0$

$g(3)$ is <u>undefined</u> since $f(3,n)\neq 0$ for any $n\in\mathbb{N}_0$

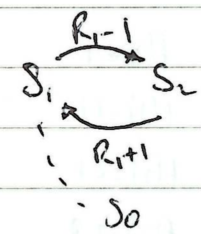
So overall, g is the following computable function:

$g:\mathbb{N}_0\to\mathbb{N}_0$

$g(0)=0$

$g(m)$ undefined for any $m>0$

$\Big\}$ this is <u>computable</u> as shown earlier

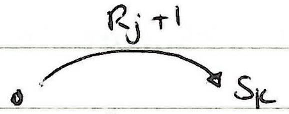


Let's now see how to <u>encode</u> programs

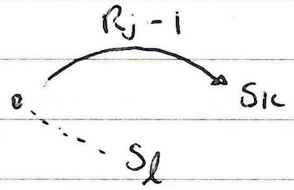
ie to find, for each programme we will try to find a <u>unique</u> identifier, in the natural numbers

Let's start by encoding single instructions.

Suppose we are in state S_i, there are two possible instructions:

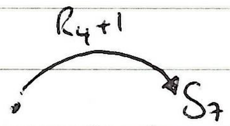


we encode this using 2^j3^k

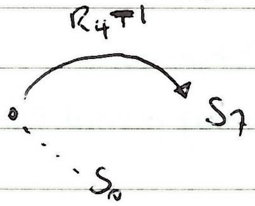


we encode this using $2^j3^k5^{l+1}$

Then eg



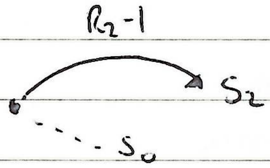
encoded by $2^4 3^7$



encoded by $2^4 3^7 5^1$

Similarly: $180 = 2^2\times 3^2\times 5^1$

so 180 encodes



defines $f:\mathbb{N}_0\to\mathbb{N}_0$

$f(0)=0$, $f(n)=n+1$ for $n>0$.

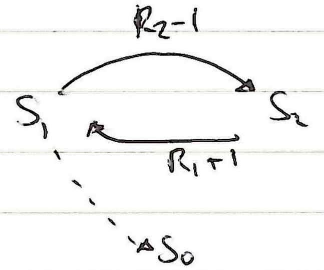
- How do we now encode the whole program?
- A program has a number of states, each associated to an instruction
Use the following code / natural number

$$2^{\text{code of state } S_1}\; 3^{\text{code of state } S_2} \cdots p_n^{\text{code of state } S_n}$$

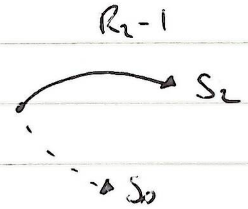
$\uparrow$ n^{th} prime number

<u>Note</u>:
A program has a finite number of <u>states</u>, so the above code will always be a well defined natural number

e.g. lets encode



Code of instruction in S_1:



is $2^2 3^2 5^1$

Code of ~~instruction~~ instruction S_2: $S_1 \xleftarrow{R_1+1} S_2$ $\quad 2^1 3^1 = 6$

So, code ~~the~~ of whole program is $2^{180} 3^6$

Recursive (partial) function

Definition

A partial function on $f: \mathbb{N}_0^k \to \mathbb{N}_0$ is recursive if it can be defined (inductively) as follows:

1) Any zero function $f: \mathbb{N}_0^k \to \mathbb{N}_0$ is recursive

$f(n_1, \dots, n_k) = 0$

2) The successor function $f: \mathbb{N}_0 \to \mathbb{N}_0$ is recursive

$f(n) = n+1$

3) Any projection function $f: \mathbb{N}_0^k \to \mathbb{N}_0$ is recursive

$f(n_1, \dots, n_k) = n_i$ for $1 \le i \le k$

4) Applying composition to recursive (partial) functions leads to a recursive (partial) function

5) Applying primitive recursion to recursive (partial) functions leads to a (recursive) partial function

6) Applying minimalisation to a recursive (partial) function leads to a recursive, possibly partial, function.

Note: The functions involved in parts (1) to (3) are computable functions as shown earlier, similarly, the processes involved in parts (4) to (6) take computable, possibly partial functions.

So If f is a recursive (partial) function, then f is a computable (partial) function

Examples of recursive functions:

1) The constant function $f_1: \mathbb{N}_0^k \to \mathbb{N}_0$ is recursive

equivalent to $f(n_1, \dots, n_k) = n$

$(n_1, \dots, n_k) \mapsto n$

e.g it can be obtained by composing a zero function with 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 use primitive recursion and the projection and successor functions

$f_2(m,0) = m$ $f_2(m,k+1) = f(m,k)+1$

projection successor

3) The multiplication $f_3: \mathbb{N}_0^2 \to \mathbb{N}_0$ is recursive

e.g we may apply primitive recursion, and use the zero function and the addition function (from earlier)

$f_3(m,0)=0$ $\quad$ $f_3(m,k+1)=f_3(m,k)+m$ ← another way of writing this

zero function $\quad$ addition $\quad$ $f_3(m,k+1)=f_2(f_3(m,k),m)$

4) The exponentiation function $f_4: \mathbb{N}_0 \to \mathbb{N}_0$ is recursive

$$f_4(m,n)=m^n$$

e.g. we may use primitive recursion, and the constant and multiplication functions from earlier

$$f_4(m,0)=1 \qquad f_4(m,k+1)=m f_4(m,k) = f_3(m, f_4(m,k))$$

From these, we may obtain forms of 'subtraction' and 'division' a recursive functions

Using a way of encoding computable (partial) functions by ~~a~~ natural numbers, it is possible to show that:

If f is a computable (partial) function, then f is a recursive (partial) function

Overall, we obtain an equivalence of recursive and computable (partial) functions

We may then use computable partial functions / register machines to show that recursive partial functions are countable.

For any $n \in \mathbb{N}$, let

$$f_n: \mathbb{N}_0 \to \mathbb{N}_0 = \begin{cases} \text{if } n \text{ \underline{is} a code for a computable (partial) function, then } f_n \text{ is the corresponding function} \\ \text{if } n \text{ is \underline{not} the code for any register machine, then } f_n \text{ is undefined} \end{cases}$$

eg/ $S_1 \xrightarrow{n+1} S_0$ $\quad$ Instruction in S_1: $\xrightarrow{R_1+1} S_0 \rightsquigarrow 2^1 3^0$

So the whole program corresponds to $2^2=4$

Hence f_4 denotes the successor function.

So, we may assume that there is a list containing all recursive partial functions

$$f_1, f_2, \ldots$$

We use this list to define a non-recursive function:

<u>Proposition</u>: Consider $g: \mathbb{N}_0 \to \mathbb{N}_0$

$$g(n) = \begin{cases} f_n(n)+1 & \text{if } f_n(n) \text{ is defined} \\ 0 & \text{if } f_n(n) \text{ is undefined} \end{cases}$$

This is not recursive.

Proof (by contradiction):

Suppose that g is recursive. Then, since it is defined everywhere, it corresponds to some recursive function f_m that is defined everywhere

But then $g(m) = f_m(m) + 1 \neq f_m(m)$

Since $g(m) \neq f_m(m)$, it cannot be the case that g is f_m

So, g is not recursive □

The function g above relates to what is known as the 'halting program'

It shows that it is not possible to have a register machine that tells us (that decides) whether or not any register machine will terminate, given a certain input.

Placing this is first order logic shows that this type of logic has no decidability theorem.