レポートは「自分の言葉」で解答してください.
他人と相談したり勉強会を開くことは奨励しますが, 丸写しはいけません.
同様に, ネットで検索したり生成AIに頼ったりすることもあるかもしれませんが,
丸写しはダメです.
必ず自分で内容を理解し, 自分で消化してからレポートにしてください.
生成AIの利用にあたっては、筑波大学発行の
「教育における生成AI活用のガイドライン（学生向け）」を参照し
適切な利用を心がけてください.

GB12601

論理と形式化

水谷哲也

亀山幸義

授業の目的

ソフトウェア科学や人工知能など,

種々の分野で用いられる論理式の正確な読み書きと証明の方法,

論理学の情報科学への応用等を理解する.

命題論理および一階述語論理の形式体系と意味論を理解し,

論理による形式化の手法を習得する.

また, 論理を使ったプログラミングなど,

情報科学への論理の応用について学ぶ.

授業予定

詳しくはシラバス参照のこと.

前半　担当:水谷

04/18, 04/25, 05/02, 05/09, 05/16, 05/23

第1週　論理とは何か: 実例・応用・様々な論理,論理式による表現
情報科学における論理.

第2週　一階述語論理の意味及び体系:
意味論, 形式体系, 一階言語, 自然演繹 (NK).

第3-5週　一階述語論理の証明:様々な論理式の証明, 演習.

第6週　一階述語論理の証明:命題の同値, 双対の原理

後半　担当:亀山幸義先生

05/30, 06/06, 06/13, 06/20

第7週　情報科学における論理:命題論理とSATソルバ

第8週　情報科学における論理:導出原理とProlog

第9週　情報科学における論理:定理証明支援系

第10週　発展的な内容, 授業のまとめ

※ 後半のうち2回程度で, 簡単なプログラミング演習をするので, PCを持参してください.

成績評価

授業への出席を前提として,

授業への出席を前提として,期末レポート(50%),

授業の各回で出題されるレポート・演習の成績(50%)により評価する.

教科書, 参考書

教科書は特に指定しない

前半(第1回-第6回)の参考書

前原昭二	記号論理入門	日本評論社	2200円
林晋	数理論理学	コロナ社	2520円
小野寛晰	情報科学における論理	日本評論社	3300円

授業の前半部は基本的にこれらの本に準拠する.

etc.

講義資料

筑波大学学習管理システムmanabaに置く.

水谷への質問・要望・連絡等は

mizutani@cs.tsukuba.ac.jpまで

メールでの質問は原則として大学から支給されたtsukuba.ac.jpのドメインをもつ

メールを用いること

gmailなど他のドメインのメールには基本的に返事はしません.

様々な理由でそれが不可能な場合は学生証(顔写真が写っているもの)と

本人の質問時の顔(自撮り)をメールに添付してください.

それで本人であることを認証します.

論理とは

論理

思考の形式, 連続した思考の流れの法則.

実際の推論や論証の方法, 流れ.

あるいは, 論理学と同じ.

論理学（logic）

どのような推論が正しい（妥当である）かを体系的に研究.

演繹的推論と帰納的推論とに分けることができる.

通常は単に「論理学」という場合は前者を指す.（後者は「帰納論理学」）

記号論理学(symbolic logic)
推論を構成する文を論理式とよばれる数学の記号に類した記号で表現し,
推論の規則を記号操作の規則として定式化する体系,

命題結合記号(logical connective); 論理記号(logical symbol)
命題どうしをつないで新たな命題をつくるときに用いる記号.

論理式(logical formula)
命題を記号で表したもの.
命題を論理記号でつなぐことによって新たに得られる表現(式).
命題と同一視することもある.

命題(proposition)
真偽が判定できる文(平叙文).

命題論理(propositional logic)
個々の命題の内部には言及せずに, 命題の真偽や推論の方法を論じる.

述語(predicate)
個体xに対する性質Fを,「xはFである」のようにを述べる言語表現.
個体変数xに具体的な値を与えるとF(x)の真偽が定まる.

述語論理(predicate logic)
文の内部構造に立ち入った推論の形式化を行う.
変数xと量記号∀, ∃を用いることにより,
「すべての個体について性質Fが成り立つ」
「Fを成り立たせるような個体が少なくとも1つ存在する」
といった量表現を扱うことができる.

演繹(deduction)
数学の証明のように, 前提から論理規則に基づいて結論を導きだすこと.
得られた結論は普遍的に正しいものと考えられる.
通常は公理(普遍的に正しいと考えられる命題)から定理
(個別的に正しいことが示された命題)を導く形をとる.

公理(axiom)
1. 証明を必要とせず普遍的に真であると考えられる命題.
　　それ自身は証明不可能であり, 演繹によって正しいと示される他の命題の前提になる.
2. 特定の理論の領域のみで正しいと仮定される基本的な前提となる命題.
　　自明の真ではなく, 理論のとり方によって定まる.
　　逆にいうと, 公理の集合(公理系)を一つ定めれば理論が固定される.
　　ある公理系(理論)で公理である命題も, 他の理論では証明されるべき定理になることも
偽になることもある.

定理(theorem)
公理または定義から出発し, 演繹により証明された命題.

帰納(induction)

推論方法の一つ.

具体的事実の集合から一般的法則ないし公理を導きだすこと.

演繹的推論とは逆に, 特殊な命題の集合から普遍的公理, 推論規則を導き出す.

帰納的論理学(inductive logic)

帰納的推理や帰納法の妥当性を扱う論理学.

数学的帰納法

自然数nに関する命題Fについて,

(1)n=0について成立し, (F(0)),

(2)任意の自然数kについて, 命題が成立(F(k))したと仮定すれば,

k+1についても成立(F(k+1))すること,

が証明されるならば, その命題はどんな自然数についても成立する(∀x(F(x)))

という推論規則.

帰納的定義

ある性質をもつ要素の集合を定義(確定)する場合,

最初に成立する要素をいくつか与え,

次に要素を定める操作・手段を与えることにより集合を確定する方法.

これらは, 演繹的論理学での推論規則・定義であることに注意.

論理学の歴史：
アリストテレスの定言三段論法

アリストテレス（384-322BC）

論理学の祖

定言三段論法

例：

- 全ての論理学者は人間である．ところが，いかなる豚も人間ではない．
  したがって，いかなる豚も論理学者ではない．
- 全ての人間は論理学者である．ある豚は論理学者である．
  したがって，ある人間は豚ではない．

定言

全称肯定　全てのAはBである．

全称否定　いかなるAもBではない．

特殊肯定　あるAはBである．

特殊否定　あるAはBではない．

仮言三段論法・命題論理

仮言 「ならば」でつながれる条件文

テオフラストス（371?-286BC）の仮言三段論法

AならばB
BならばC

∴ AならばC

AならばC
BならばCでない

∴ AならばBでない

AならばB
Aでないならば C

∴ Bでないならば C

クリュシッポス（280-205?BC）の三段論法

A
AならばB

∴ B

AならばB
Bでない

∴ Aでない

（AかつB）でない
A

∴ Bでない

AあるいはB
A

∴ Bでない

AあるいはB
Aでない

∴ B

「かつ」 連言
「あるいは」 ここでは排他的選言

ブールの論理代数

ブール(Boole, 1815-1864)による論理学の発展

x, y, ...:	クラス　ものの集まり　を表す
1:	全てのものからなるクラス
0:	何も含まないクラス

クラスの演算

xy：	xとyのいずれにも含まれる共通部分
x+y：	xとyを合併したクラス
1-x：	全クラスからxを除いたクラス

定言命題は以下のように表される

x(1-y)=0	全てのxはyである.
xy=0	いかなるxもyでない.
xy=v	あるxはyである.（共通部分がvである）
x(1-y)=v	あるxはyではない.

「xならばy, ところがyではない. したがってxではない. 」の証明

前提　x(1-y)=0…(1)　y=0…(2)

(2)を(1)に代入すればx(1-0)=0. したがってx=0.

現代の記号論理学

記号論理学においては「論理記号」をもちいて「命題」を表す.

論理記号

記号	意味	他に使われる記号
⊃	ならば	→ など
≡	同値	⊃⊂, ⇔, ⇄など
∧	かつ	&, ・など
∨	または	+など
¬	否定	～, ⁻など
∀	全ての	∀xのかわりにAx, (x)など
∃	ある	∃xのかわりにEx, (Ex) など

論理記号$\supset$と集合の記号$\subset$, $\supset$について.

F, Gが集合そのものを表しているか命題または述語記号を表しているかで

同じような記号を使っていても意味が異なることに注意(詳しくは後述).

なお, 論理記号としては「ならば」の方向が逆になった「$\subset$」はほとんど用いられない.

命題結合記号(論理記号)と論理式

A,Bを命題(論理式)を表す記号とする.

以下のようにして新たな命題(論理式)がつくられる.

A⊃B	AならばB
A≡B	AとBは同値
A∧B	AかつB, Aでありしかもである
A∨B	AまたはB
¬A	Aでない

A≡Bは (A⊃B)∧(B⊃A)と同じ意味

A∨Bは「AまたはB」であるが, A,Bとも正しい場合を含む.

∀と∃は後述.

入り組んだ表現の場合は括弧を用いるが, 適切に省略する.

記号の結合の強さは以下の通りとする.

（弱い） ≡ ⊃ ∨ ∧ ∃ ∀ ¬ （強い）

例

A∧B∨C∧D　　は　　(A∧B)∨(C∧D)

A⊃B∨C∧D　　は　　A⊃(B∨(C∧D))

同じ記号が連続する場合は, ⊃以外は左側を優先する. ⊃は右を優先する.

例

A∧B∧C∧D　　は　　((A∧B)∧C)∧D

A⊃B⊃C⊃D　　は　　A⊃(B⊃(C⊃D))

命題と命題関数

命題

真偽が定まるような文.

「正しい」か「間違っている」のどちらかに決まる文.

例

　0=0, 0=1, ... ,

　東京は日本の首都である, シカゴはアメリカ合衆国の首都である,

　冥王星は太陽系の惑星の一つである, ...

x, yが変数のとき, x=0, x=yなどはそのままでは真偽が定まらない.

これらは(とりあえず)命題とは考えないことにする.

ではこれは何か.

「命題関数」 ― 関数の値が真理値(真か偽か)であるような関数 ― であると考える.

例

x+3=xy をF(x, y)とする.

F(1, 4)は1+3=1×4という真の命題

F(2, 2)は2+3=2×2という偽の命題

述語と性質

F(x)を命題関数「xはFである」と読むとする.

このFは「…はFである」の部分であるので, これを**述語(predicate)**と呼ぶ.

xに定数aを代入したものF(a)は「aはFである」すなわち

「aはFという性質を持つものである」と考えることができる.

例

G(x)で「xは死ぬ」とする.

sを「ソクラテス」という「定数」とする.

このとき, G(s)で「ソクラテスは死ぬ」という命題を表す.

F(x)で「xは人間である」とする.

すると, $\forall x(F(x) \supset G(x))$で, 「全てのxに対して, xが人間ならばxは死ぬ」

すなわち「人間は死ぬ」ということを表す.

全称命題と存在命題

F(x): 1変数命題関数に対して

∀xF(x)：　全てのxに対してF(x)　　全称命題

∃xF(x)：　F(x)というxが存在する　　存在命題

という命題を表す.

命題関数ではないことに注意.

∀：　　全称記号, 全称作用素　universal quantifier

∃：存在記号, 存在作用素　existential quantifier

これらを**量記号**または**限定作用素** quantifier とよぶ.

多変数の命題関数

n変数の命題関数$F(x_1, \ldots, x_n)$をn項関係とよぶ.

例えば等式$x=y$や不等式$x<y$は2項関係

また, n変数述語ともよぶ.

関係, 命題関数, 述語はみな同義語と考え, 主に述語とよぶことにする.

n+1変数述語のひとつの変数に定数を代入するとn変数述語が得られる.

例

2変数述語$x=y$のxに1を代入すると1変数述語$1=y$が得られる.

また, n+1変数述語のひとつの変数に対し量記号を適用すると

n変数述語が得られる.

例

2変数述語$x=y$のxに対し存在記号を適用すると$\exists x(x=y)$を得る.

これは1変数述語である.

例

変数として自然数を考える.

F(x, y)が $x=2y \vee x=2y-1$のとき

$\forall x\, F(x, y)$ すなわち $\forall x(x=2y \vee x=2y-1)$

$\exists y\, F(x, y)$ すなわち $\exists y(x=2y \vee x=2y-1)$

$\forall x \exists y\, F(x, y)$ すなわち $\forall x \exists y(x=2y \vee x=2y-1)$

「全てのxに対して, (xごとに) $x=2y$ または $x=2y-1$となるyが存在する」

$\exists y \forall x\, F(x, y)$ すなわち $\exists y \forall x(x=2y \vee x=2y-1)$

「あるyに対して, 全てのxで$x=2y$ または $x=2y-1$となる」

例

変数として自然数を考える.

$\forall x \forall y\ (x<y)$　全てのxに対してどんなyでもx<y となる.　偽

どんなxをもってきても$\forall y(x<y)$が真になるということ.

これはあり得ない.

$\forall x \exists y\ (x<y)$　全てのxに対してあるyでx<y となる.　真

たとえばxに対してyをx+1ととればよい.

$\exists x \forall y\ (x<y)$　あるxに対してどんなyでも x<y となる.　偽

$\forall y(x<y)$を真にするxがひとつ以上あるかということ.

ひとつもないので偽.

$\exists x \exists y\ (x<y)$　あるxに対してあるyなら x<y となる.　真

xとしてnをとったときyとして例えばn+1をとればよい.

自由変数, 束縛変数

量記号∀x, ∃xの作用範囲における変数xを**束縛変数**bound variableという.
このとき, 量記号はその変数を束縛する, 変数は量記号で束縛されるという.
それ以外の変数を**自由変数**free variableという.
例

$x=2y \vee x=2y-1$	x, yとも自由変数
$\forall x(x=2y \vee x=2y-1)$	xが束縛変数, yが自由変数.
$\exists y(x=2y \vee x=2y-1)$	yが束縛変数, xが自由変数.
$\forall x\exists y(x=2y \vee x=2y-1)$	x, yとも束縛変数

$\forall x(x=2y) \vee x=2y-1$	左側のxが束縛変数, yと右側のxが自由変数.

同じ式の中の同じ変数(やその他の式や項など)を区別するため,
出現(occurrence)という表現を用いる.
xの右側の出現, 左から2番目の出現, etc.

$\exists z(x=2y \vee x=2y-1)$

このような, 量記号の直後に出現しない変数を置く式も許す.

束縛変数は「仮の変数」であるため，変数名を置き換えても意味が変わらない．

ただし，置き換え方には制限がある．

例

$\forall x \exists y(x=2y \vee x=2y-1)$　と　$\forall z \exists y(z=2y \vee z=2y-1)$　は同じ．

$\forall x(x=2y) \vee x=2y-1$　と　$\forall z(z=2y) \vee x=2y-1$　は同じ．

$\forall x(x=2y) \vee x=2y-1$　と　$\forall y(y=2y) \vee x=2y-1$　は異なる．

自由変数には他の項（変数や，同じタイプの記号表現）を代入してもよい．

ただし，束縛変数と同じ変数名を含んではいけない．

例

変数として自然数を考える．

$\exists y\ (x<y)$　（xより大きいyが存在する）に対して，

xに他のものを代入することを考えたとき．

$\exists y\ (3<y)$とか $\exists y\ (n+1<y)$とか $\exists y\ (x+1<y)$などの代入は許す．

$\exists y\ (y<y)$や $\exists y\ (y+1<y)$は許されない．

⇒なぜか.

∃y (x<y)のxの部分はyによって束縛されていない,

そこにyを含むもの(例えばy+1)を代入すると束縛されてしまい意味が変わる.

代入されるy(例えばy+1)はあくまで自由変数である.

⇒代入したくなったらどうするか.

演習問題.

ヒントは,

『束縛変数は「仮の変数」であるため, 変数名を置き換えても意味が変わらない. 』

∃y (x<y)と∃z (x<z)は同じ意味である.

※　自由変数は「式の外部からアクセス可能な変数」　＝　大域変数

束縛変数は「式の内部だけで用いられる変数」　＝　局所変数

のようにイメージすると理解しやすいかも.

論理記号の用例

F,Gを述語とする.

$\forall x(F(x) \supset G(x))$

GはFのための必要条件, FはGのための十分条件.

Fという性質はGという性質を含む.

Fという性質をもつもの(x)は必然的にGという性質をもつ.

集合として考えたとき, FはGの部分集合である.

すなわち $\forall x(x \in F \supset x \in G)$, すなわち$F \subset G$. FはGに含まれる.

$\forall x(F(x) \equiv G(x))$ すなわち$\forall x((F(x) \supset G(x)) \wedge (G(x) \supset F(x)))$ (*)

FはGの必要十分条件である.

FとGは同値(equivalent), 等価, 等しい, 同じ.

(*) これは

$$\forall x(F(x) \supset G(x)) \wedge \forall x(G(x) \supset F(x))$$

と同値であるが自明ではない. 証明が必要.

$\forall x(F(x) \vee G(x))$

FまたはGである.

Fという性質を持つかGという性質を持つかのいずれかである.

両方の性質を持ってもよい. (排他的ではない)

集合として考えたとき，xはFとGの和集合の要素である. すなわち $\forall x(x \in F \cup G)$.

$\forall x(F(x) \wedge G(x))$

FかつGである.

Fという性質を持ち, さらにGという性質を持つ.

集合として考えたとき，FとGの共通部分の要素である. すなわち $\forall x(x \in F \cap G)$.

$\forall x(\neg F(x))$

Fではない.

Fという性質を持たない.

集合として考えたとき，Fの補集合の要素である. すなわち $\forall x(x \notin F)$.

「FはGではない」という式は以下の4通りが考えられる.

$\forall x(F(x) \supset \neg G(x))$　　Fは必ず「Gではない」.

$\neg \forall x(F(x) \supset G(x))$　　Fは必ずしもGではない.

$\forall x(F(x) \equiv \neg G(x))$　　Fとは「Gでない」ということである.

$\neg \forall x(F(x) \equiv G(x))$　　FとGとは異なる.

F : 白馬, G :馬　とすると,「白馬は馬に非ず」は上記1-4のうちいずれか.

あるいは他の解釈があるか.

論理式の用例

変数として自然数を考える.

a<bという命題は, =と+より, ∃x(¬x=0∧a+x=b) と表すことができる.
a≤bは, a=b∨a<b である.

「bはaで割り切れる」は, a|b と表記するが, これは∃x(ax=b)で定義できる.

「aは素数である」すなわち「aは2以上で, 1と自分自身以外に約数を持たない」を
Prim(a)と表記するとする. これは

∀x(x|a⊃(x=1∨x=a))∧(1<a)

展開すると
∀x(∃y(xy=a)⊃(x=1∨x=a))∧∃x(¬x=0∧1+x=a)

で表すことができる.

「素数は無限にある」は　$\forall x \exists y(x<y \wedge \mathrm{Prim}(y))$　と表される. なぜか？

一般に, 自然数のある性質Fを満たす自然数が無限個あることは,

$$\forall x \exists y(x<y \wedge F(y))$$

で表される.
もし, 有限個の自然数しかFを満たさないとすれば, そのうちの最大の要素がある.
これをnとする.
このとき, $n<y \wedge F(y)$なるyは存在しないので, $\exists y(n<y \wedge F(y))$は成り立たない.
すなわち $\forall x \exists y(x<y \wedge F(y))$は正しくない.
逆に,　$\forall x \exists y(x<y \wedge F(y))$が正しくないとする.
「全ての自然数xについてG(x)」の否定は「ある自然数nについて¬G(n)」なので,
あるnについて $\neg \exists y(n<y \wedge F(y))$が成り立つことになる.
「nより大きいあるyについてF(y)が成り立つ」の否定は
「nより大きいどんなyについてもF(y)は成り立たない」, すなわち
「F(y)が成り立つどんなyもn以下である」すなわち$\forall y(F(y) \supset y \leq n)$である.
これはF(y)が成り立つ自然数が有限個であることを示している.

情報科学における論理

- プログラミング自身が論理的思考の具現化
- プログラムの仕様表現(Formal Specification)を示すこと
- 論理式の真偽を計算機で解くこと

例題

GCDプログラム（Python3）

```
def gcd(x, y):
    while y != 0:
        x, y = y, x % y
    return abs(x)
```

仕様

整数x, yの最大公約数GCD(x, y)を

求めるプログラム

但しx, yのどちらかは0でないとする

仕様の論理的表現

入力(実行開始直前)の変数x, yの値をx_I, y_I,
出力(実行終了直後)の変数xの値をx_Oとおく.
このとき, 仕様は

$$(x_I \neq 0 \vee y_I \neq 0) \wedge Z(x_I) \wedge Z(y_I) \supset (x_O = GCD(x_I, y_I))$$

(Z(x)はxが整数であることを示す述語)

と表すことができる.

例題

自然数値であるxの階乗を求めるプログラム

```
while z != x:
    z = z + 1
    y = y * z
```

の仕様を表現する.

ここで, 入力時の変数の値をx_I, y_I, z_I, 出力時の変数の値をx_O, y_O, z_Oとする.

また, N(x)はxが自然数であることを示す述語とする.

仕様(例)

$$N(x_I) \wedge y_I=1 \wedge z_I=0 \supset (y_O=x_I!)$$

言語の(割と)厳密な定義

まとめとして, 一階述語論理で用いる言語の定義を詳しく述べる.

言語(language):
ある事柄を論じようとしたとき, それを述語論理で記述するために
必要となる記号の集まり
それらがどのように並べてよいかという文法(syntax)に従う.

一階述語とは, 個体(自然数とか, 実数とか,「人」とか)に関する性質を述べるものである.
述語の性質を述べる述語は「第2階の述語」といい, 本授業では扱わない.

項(term), 式(formula)

項

個体定数, 個体変数は項である.

f がn引数の関数記号, t_1, ... , t_n が項であるとき, f(t_1, ... , t_n) は項である.

式(論理式)

pがn引数の述語記号, t_1, ... , t_n が項であるとき, p(t_1, ... , t_n) は

式である. (原子論理式)

A, A_1, A_2が式であるとき, (¬A), (A_1∧A_2), (A_1∨A_2), (A_1⊃A_2) は式である.

Aが式, xが個体変数であるとき, (∀xA), (∃xA)は式である.

論理記号の結合の優先順位

（弱い） ≡ ⊃ ∨ ∧ ∃ ∀ ¬ （強い）

同じ記号が並ぶときは左にあるものが強い

⊃は右が強い

それに従って適宜カッコを省略する

eg.　A ∧ B ∧ C は ((A ∧ B)∧C), A ∨ B ∨ C は ((A ∨B) ∨ C)),

　A ⊃ B ⊃ C は (A ⊃ (B ⊃ C))

cf.　四則演算だと

（弱い） - ＋ ÷ × -(単項)（強い）

同じ記号が並ぶときは左にあるものが強い

eg.　x + y + z は ((x + y) + z), x × y × z は ((x × y) × z)

x ÷ y ÷ z は ((x ÷ y) ÷ z), x - y - z は ((x - y) - z)

メタ変数

「xが変数であるとき」というような表現において, “x”は「本当の」変数ではなく,
変数であることを表す表現である.
このような変数をメタ変数(metavariable)という.
「Aが式のとき」という場合, もちろん式はAという一文字の記号であるわけではなく,
具体的には例えば (x=y ∧ y>z) (すなわち(=(x, y) ∧ >(y, z))) というようなものである.
そういったありとあらゆる式を代表してメタ変数Aで表す.

部分式

ある式に対して, それを原子論理式から組み立てる過程に現れる式

例:

$A(x) \vee \forall x(B(x) \wedge C(x))$の部分式は

$A(x)$, $B(x)$, $C(x)$, $B(x) \wedge C(x)$, $\forall x(B(x) \wedge C(x))$, $A(x) \vee \forall x(B(x) \wedge C(x))$

出現

式中のそれぞれの位置での記号

例:

$A(x) \vee \forall x(B(x) \wedge C(x))$においてxの出現は3回（$\forall x$のxは通常カウントしない）

スコープ

$\forall xF$, $\exists xF$の形の部分式の出現のFを, $\forall x$または$\exists x$のスコープという

例:

$A(x) \vee \forall x(B(x) \wedge C(x))$において$\forall x$のスコープは $B(x) \wedge C(x)$

自由変数, 束縛変数

定義

（1） 原子論理式の変数の出現は自由変数である.

（2） $F_1 \wedge F_2$, $F_1 \vee F_2$, $F_1 \supset F_2$, $\neg F_1$の変数の出現は F_1, F_2の対応する出現が
自由変数ならば自由変数, 束縛変数ならば束縛変数である.

（3） $\forall x F$, $\exists x F$の変数xの出現はすべて束縛変数である. その他の変数の出現は,
Fにおいて自由変数ならば自由変数, 束縛変数ならば束縛変数である.

例:

$A(x) \vee \forall x(B(x) \wedge C(x))$

閉じた論理式

閉じた式(論理式)

式Aが自由変数を一つも含まないとき, Aを閉じた式という.

Aに出現する自由変数がx_1, ... , x_n のとき, $\forall x_1 \ldots \forall x_n$ A　をAの閉包(closure)という.

真理値

命題の真偽を表すために, **真理値(truth value)**を用いる.

正しい命題	⊤	真 true	(⊤と表すこともある)
間違った命題	⊥	偽 false	(⊥と表すこともある)

真理値の計算

命題論理においては, **真理表**によって式の真偽を計算することができる.

（命題論理: 命題の真偽性や推論形式を, 命題の内部構造に立ち入らずに,
結合形式のみに即して論ずる）

¬の真理表

A	¬A
⊥	⊤
⊤	⊥

⊤　正しい命題　　真　true

⊥　間違った命題　　偽　false

2項の論理記号に対しては, 以下のような$2^2=4$通りの場合を考えればよい.

A	B	A□B
⊥	⊥	
⊥	⊤	
⊤	⊥	
⊤	⊤	

⊤　正しい命題　　真 true

⊥　間違った命題　　偽 false

∧の真理表

A	B	A∧B
⊥	⊥	⊥
⊥	⊤	⊥
⊤	⊥	⊥
⊤	⊤	⊤

∨の真理表

A	B	A∨B
⊥	⊥	⊥
⊥	⊤	⊤
⊤	⊥	⊤
⊤	⊤	⊤

⊃の真理表

A	B	A⊃B
⊥	⊥	⊤
⊥	⊤	⊤
⊤	⊥	⊥
⊤	⊤	⊤

述語論理の式に対してはこのような簡単な計算はできない.

$\forall x F(x)$ は xの変域(**対象領域**)が有限個の $e_1, e_2, \ldots, e_n$ であれば
$F(e_1) \wedge F(e_2) \wedge \ldots \wedge F(e_n)$ と一致するが,

対象領域が無限集合であればそのような有限の方法では処理できない.

⇒ **モデル**と**意味論**を考える必要がある.(後述)

「AならばB （A⊃B）」に注意.
特に, 前提が間違っている論理（推論）に注意
また. 前提と結論の間に因果関係があるとは限らないことにも注意.

1. 「晴れならば遠足に行きます. 」⇒ 雨が降っても遠足が決行された.
2. 「2+2=4ならば東京は日本の首都である」
「2+2=5ならばコペンハーゲンは日本の首都である」

「AまたはB （A∨B）」にも注意.

1. 「Trick or Treat?（お菓子かいたずらか）」⇒
お菓子をあげたのにいたずらされた.
2. 「2+2=4または東京は日本の首都である」
「2+2=5またはコペンハーゲンは日本の首都である」
3. 「テストで良い点数を取るか怒られるか」⇒ 良い点数を取ったのに怒られた.

“Trick or treat?”（“Treat me or I’ll trick you.”）

直訳	「いたずらするか(お菓子を)おごるか. 」	A∨B
意訳	「おごってくれなきゃいたずらするぞ」	¬B⊃A
	「いたずらされたくなかったらお菓子をおごれ」	¬A⊃B

“Trick or treat?”（“Treat me or I’ll trick you.”）

直訳	「いたずらするか(お菓子を)おごるか.」	A∨B
意訳	「おごってくれなきゃいたずらするぞ」	¬B⊃A
	「いたずらされたくなかったらお菓子をおごれ」	¬A⊃B

子供　「お菓子をくれなきゃいたずらするぞ」

母親　「あんたはお菓子をあげてもあげなくてもいたずらばっかりしてるじゃないの!」

A:いたずらする　B:おごる

以下の文で同値のものはどれか.

「いたずらするかおごるか」	$A \vee B$
「おごってくれなきゃいたずらするぞ」	$\neg B \supset A$
「いたずらしないならおごれ」	$\neg A \supset B$
「おごるならいたずらしない」	$B \supset \neg A$
「いたずらするならおごらない」	$A \supset \neg B$

次の推論は正しいか.

大学生は勉強が嫌いである.

私は大学生である.

だから私は勉強が嫌いである.

次の文は正しいか.

もし大学生は勉強が嫌いであり, そしてもし私は大学生ならば,

私は勉強が嫌いである.

次の推論は正しいか.

大学生は勉強が嫌いである.

私は大学生である.

だから私は勉強が嫌いである.

ところが本当は私は勉強が嫌いでない.

ということは大学生は勉強が嫌いであるとは限らない.

レポート問題

1. n+1引数の命題関数$F(x_1, x_2, \ldots, x_{n+1})$に対して，ある変数$x_i$ $(1 \leq i \leq n+1)$を1つだけ量記号∃で束縛した場合，どのような命題関数が得られるか．
形と引数の数を答えよ．
2. $\forall y(x<y)$に対して，xにy+1を代入したいとする．
xをy+1で置き換えて $\forall y(y+1<y)$にしてはいけない．
また，$\forall y(x<y) \wedge x=y+1$も代入した結果とはいえない．
何がいけないのか．どのようにすれば正しい式になるのか．考察せよ．
3. 以下の文を，述語記号，関数記号などを適切に与えることにより，
述語論理の式として表現せよ．
「もし大学生は勉強が嫌いであり，そしてもし私は大学生ならば，
私は勉強が嫌いである．」

締切　4月25日　8:40
提出先　manaba

レポート解答の注意事項

少なくとも水谷担当分の第1回から第6回までは以下の方針で行います．
亀山先生の担当の第7回から第10回については亀山先生の指示に従ってください．

- 解答はpdfで提出してください．
    - Word等で作成した場合，手書きのノートを写真で撮影した場合も，pdfに変換してください．
- 解答用紙の形式は特に指定しませんが，必ず学籍番号と名前を書いてください．
    - 学籍番号と名前が書いていないものは．解答として認めません。．
- 手書きのノートなどを撮った写真を提出する場合，写真は解答内容を読めるように撮影してください（斜めから撮ったりしない）．