数理論理学 A メモ

T.Sakuragawa

2022 年 7 月 14 日現在

目次

0　数理論理学とは[*1]

以下の学問分野はある程度似た分野を意味する.

1. 数理論理学 (Mathematical Logic)
2. 数学基礎論 (The Foundations of Mathematics)
3. 記号論理学 (Symbolic Logic)
4. 形式論理学 (Formal Logic)

これらは記号を使って論理を厳密に定義し，それによって論理そのものあるいはそれを使って行う数学について研究する分野である.

ただし研究の方向性はいくつかある.

1. 論理そのものを記号化して研究するもの．必ずしもその中で数学を展開しない.
2. 記号化した論理が最初にあり，それの一部として数学が展開されるという立場のもの.
3. 記号化した論理とそれに纏わる数学を研究対象とし，数学の研究結果と方法を適用するもの.
4. 3 のうち，対象を紙に書ける具体的な記号列のみで表せる論理とし，それらに対する有限的な推論のみを用いて無矛盾性を示すもの.

数理論理学の教科書の多くは 3 の立場で書かれているし，この資料も同じである．ただしそれぞれ中身の一部が 4 の立場を取っている (説明している) 場合がある.

最後の名称: 数学基礎論は数学の基礎に関する研究という意味だが，実際の中身は応用数学とも言える．Hilbert の計画からは正にそうと言える.

0.1　古典命題論理と古典一階述語論理を扱う

数理論理学が対象とする論理は多数ある．それらのうちこの資料では (古典) 命題論理 (propositional logic)，(古典) 一階述語論理 (first order predicate logic) それぞれの証明論，モデル論の土台である意味論 (Semantics)，計算論の入り口を，特にコンピュータ (計算機) との関わりの部分について詳しめに解説する．ここでは用語の一部をざっと説明する．ただし後でもう少し詳しい説明を行う.

古典とは logic の文脈では二値とだいたい同じことを意味する場合が多い．真か偽かどちらかであるような論理を意味する (詳しく言えば Boole 代数であればよい)．命題とは一般に真理 (真偽) 値が定まる言明のことである．命題論理では命題の中身は問わずに命題変数とし，それらを $\wedge$(and) や $\vee$(or)，$\neg$(not) などの論理結合子で結合してできる論理式の性質を考察する.

述語とは引数 (ひきすう) を取って 真理値が定まるもので，例えば $=$ や $\in$ がそうである．述語論理では $\forall$(全て) と $\exists$(存在) という量化記号を用いる．さらに現代の述語論理では定数記号・函数記号も用いる．述語論理では目的によって用いる定数記号と函数記号の集合が異なるため，それらの違いにより論理式達の言語 L が異なってくるところが命題論理と異なる．一階，というのは変数の動く範囲が自然数全体の集合，集合全体などの基本的な数学的対象の集合あるいはクラスであり，函数や集合の上を動く変数はないという意味である.

[*1] 本節の内容と各節の最初の部分は大まかな説明であり，其々に対応する細部の説明があれば参照すること．赤は強調部分或いは重要概念 (の初出) である.

一階述語論理以外の論理には二階述語論理や高階論理がある．一階述語論理を用いるとその中で自然数論など，ある種の数学を展開できる (その中で全ての数学を展開できる集合論を一階述語論理で展開できるという意味では全ての数学を展開できる)．二階述語論理によりある種の解析学を展開できる．

0.2 論理式を数学的に扱う

形式言語理論での扱いとは異なるが，本質的に形式的構文規則 (formal syntax) により論理式を定義する．論理式全体の集合 (the set of logical formulas) を一意に定める．
文法的に正しい論理式と，そうでない論理式を機械的に判定できる (これを指して決定可能と言う．論理式の候補を記号列として入力すると，必ず停止し，yes, no を正しく答えるプログラムを書けること．言い換えると誰が判定しても同じ結果になるということ)．
数学的に厳密に定義する理由は，(1) 数学的に厳密な議論を展開したいためと，(2) 特に不可能性の証明のため．

0.3 数理論理学は主に 4 分野から成る

以下の 4 つの分野から成る．

1. 証明論 (Proof theory)
2. モデル論 (Model theory)
3. 計算論 (Theory of Computation)，再帰理論 (Recurtion Theory)
4. 公理的集合論 (Axiomatic Set Theory)

それぞれある程度後述する．

0.3.1 証明論

正しい論理式を文字列として定義する．意味は (表面上) 考えない (形式的)．
条件なしで正しい論理式を公理 (axiom) と呼ぶ．公理全体の集合は通常は決定可能なものとして与える．大まかに言えば，他の正しい論理式達から他の正しい論理式を導く規則を推論規則 (derivation rule) と呼ぶ．公理から推論規則の適用を有限回繰り返して得られる (正しい) 論理式を定理 (theorem) と呼ぶ．正しい論理式 (定理) 全体の集合は通常，全て機械的に生成できる (再帰的可算あるいは半決定可能．決定可能との違いは，判定するプログラムが止まらない場合も許した上で，論理式がその集合に入っている $\Leftrightarrow$ 論理式をプログラムの入力とすると停止して yes を返すとなること) ものとなる場合が多い．証明論的な正しさが定義される．一階述語論理の閉論理 (自由変数が現れない論理式) の集合 (理論と呼ぶ．通常の数学で言う公理に当たる) から推論規則を繰り返し適用して得られる定理を求めていくのが，数学の現場で行われていることの形式的なモデル化になっている (ただしこの部分は実際に全くそうであることを証明できるわけではない)．

0.3.2 意味論

他の数学的対象を用いて論理式に意味を与える．例えば集合論により各論理式に真偽値を与える (Tarski 意味論，L 構造と呼ぶ)．古典一階述語論理の場合，空でない集合 D，定数・函数記号と述語記号の意味をそれぞれ具体的な D 上の定数・函数，D の (いくつかの直積の) 部分集合として与える．すると各 (閉) 論理式に

対してそれが真か 偽かが数学的に定義される (決定可能とは限らない)．圏論 (Category theory) を使って意味を与える方法もある (図論理 (Categorical Logic)) 論理式の意味論的な正しさが定義される．
証明論的な正しさと意味論的な正しさが一致する場合がある (完全性 (completeness) と健全性 (soundness)．古典命題論理，古典一階述語論理では一致する) ある理論 T に対し，その元の論理式達を全て真にする L 構造を T のモデルと呼ぶ．

0.3.3　計算論

計算 (computation) とは何かと，計算可能 (computable) 函数を厳密に定義する．特性 (charactaristic) 函数が計算可能な (自然数の部分) 集合を決定可能 (decidable) 集合と言う．計算不可能な函数の存在は集合論の濃度 (cardinality) の概念によりすぐ分かる．Turing が対角線論法 (diagoral argument) により具体的に構成した．決定可能性 (decidability)，半決定可能性 (semidecidability)，再帰 (帰納) 的可算 (recursively enumerable) などが基本的な概念．誰が実行しても，計算機が実行しても結果は同じ．曖昧性がない．

0.3.4　公理的集合論

集合論を形式的に (記号論理により) 取り扱う．記号論理を援用しないと厳密に議論しにくい (不可能性等の証明含む)．そもそも数理論理学の誕生の原因の一つが (素朴) 集合論での多数のパラドックスの発見であった．

1　内容

キーワードを並べておく (本節は厳密でない)．

1. (古典) 命題論理
   命題論理式, メタと対象, 命題論理の意味論, 決定問題, 帰納的可算, 非決定性チューリング機械, 多項式時間帰着可能性, NP 完全性, 論理的帰結, 論理的同値, 論理式の同値変形, CNF, DNF, 形式的体系, シーケント計算, (体系の) 健全性, 完全性, Wang のアルゴリズム, Cut 除去定理
2. (古典) 一回述語論理については省略する．

1.1　命題論理式の例

古典命題論理式の例を示す (最初なので時々略さず括弧を書いている)．

$$(\underline{A} \wedge \underline{B}) \to \underline{C}$$

ここで $\underline{A}, \underline{B}, \underline{C}$ を命題変数という．$\wedge, \to$ などは論理記号という (他の呼び方もあり，後述)．

1.2　命題論理の意味論

真理値とは，正しさの程度を表す値であり，古典 (二値) 論理の場合には **T** 即ち真 (正しいこと) と **F** 即ち偽 (誤っていること) である．真理値表とは，論理式中の命題変数の真理値の組み合わせに対し，それぞれの場合の論理式の真理値を書き表したものである．例えば以下が上の論理式の真理値表である．

$\underline{A}$	$\underline{B}$	$\underline{C}$	$\underline{A} \wedge \underline{B}$	$\underline{A} \wedge \underline{B} \to \underline{C}$
F	**F**	**F**	**F**	**T**
F	**F**	**T**	**F**	**T**
F	**T**	**F**	**F**	**T**
F	**T**	**T**	**F**	**T**
T	**F**	**F**	**F**	**T**
T	**F**	**T**	**F**	**T**
T	**T**	**F**	**T**	**F**
T	**T**	**T**	**T**	**T**

真理値表の値が全て **T** となる論理式をトートロジーと呼ぶ．例えば，$(\underline{B} \vee \underline{A}) \to ((\underline{A} \vee \underline{B}) \vee \underline{C})$ などはそうである．これは各命題変数の値に関わらず，常に真であるということであり，常に成り立つ式，つまり恒真式である．

1.3 証明図の例

シーケント計算 LK での証明図の例を示す．

$$\dfrac{\dfrac{\dfrac{\underline{B} \vdash \underline{B}}{\underline{B} \vdash \underline{A} \vee \underline{B}} \quad \dfrac{\underline{A} \vdash \underline{A}}{\underline{A} \vdash \underline{A} \vee \underline{B}}}{\underline{B} \vee \underline{A} \vdash \underline{A} \vee \underline{B}}}{\dfrac{\underline{B} \vee \underline{A} \vdash (\underline{A} \vee \underline{B}) \vee \underline{C}}{\vdash \underline{B} \vee \underline{A} \to (\underline{A} \vee \underline{B}) \vee \underline{C}}}$$

一方 $\underline{A} \wedge \underline{B} \to \underline{C}$ の証明図は存在しない．論理式 E が恒真式であることと $\vdash E$ の証明図が存在することが同値であることを証明できる．同値であるという事実を指して完全性 (あるいは概念を 2 つに分けて完全性と健全性) と言う．

1.4 標準形の例

一般に論理式の標準形とは，元の論理式と同等で，ある一定の形になっているもののことを言う．多項式の展開が，多項式のある種の標準形を求めていると考え，それと同じようなことを論理式で行うと考えるとよい．例えば $\underline{A} \wedge \underline{B} \to \underline{C}$ の CNF(論理積標準形) と DNF(論理和標準形) は両方とも $\neg\underline{A} \vee \neg\underline{B} \vee \underline{C}$ である ($\neg$ は否定の記号．一般には 2 つの標準形は異なる)．

2 古典命題論理

この資料では，まず古典命題論理 (以下では「古典」の部分や「命題」の部分を省略する場合がある) について紹介する．最初に命題論理式というものを定義する．論理式は帰納的に (必ずしも意味を持たない) 記号列として定義される．数学的に厳密な定義である．これにより，論理式について成り立つ事柄を証明する時に，帰納法により証明することが可能となる．

次にメタと対象の違いについて説明する．対象は，この場合研究対象となっている命題論理式達である．メタレベルでは，それらについて数学的議論を行う．この違いは重要であり，十分に理解しておかないと混乱す

る原因となる．

次に命題論理式の意味を定める．これは個々の論理式を真理値に対する函数として表す方法である．真理値表により，それら函数を表現することについても説明する．また，技術的に便利でメタレベルの証明に利用するため付値を定義する．函数として見た時，あるいは真理値表で表した時に結果の真理値が常に真であるような命題論理式をトートロジーと呼ぶ．

本稿ではその後，決定問題と決定可能性について説明することにする．後者は，必ず停止する計算機プログラムで yes か no で答えられるということである．また授業の本筋から少し外れるが，SAT 問題と NP 完全性についてある程度説明する．

NP 完全性の部分については最初は飛ばしていただいても構わない．

次に論理式の同値変形，標準形について説明し，真理値表からそれに対応する命題論理式を求める方法を説明する．

命題論理の形式的体系についても説明する．これは記号列の操作として正しい論理式を求める方法であり，公理から推論を行なって正しい論理式 (定理と呼ぶ) を導いてゆく．数学でいう証明に対応するものである (ただし実際の数学の証明を行うには，少なくとも一階述語論理が必要である．授業後半で扱う予定である)．形式的体系にはいろいろあるが，本稿ではシーケント計算の体系を扱う．このようにして論理式の正しさには，トートロジーであることと，定理であることという 2 つの正しさが定義される．古典命題論理の場合，結果的にはこれら 2 つの正しさは一致することを示す．

予定している命題論理についての説明はとりあえず以上である．

2.1 論理式

古典命題論理 (classical propositional logic) の論理式は以下のようにして定義される．ここでわざわざ古典と付けたのは古典的でない論理もあるからで，それらの場合には以下に出てくる以外の記号が入ってくる場合もある (以下で導入される演算子を用いて定義できない場合，それらは様相演算子 (modal operator) と呼ばれる場合が多い)．ただし例えば直観論理 (intuitionistic logic) は古典論理と異なる論理で論理式が表す意味も異なるが，利用される記号は以下の定義の場合とほぼ同じである．古典論理は歴史的に最も早く数学的に研究された論理であり，意味のつけ方も最も単純で，通常はこれを最初に学習する．

定義 2.1. 命題論理式

$\mathbf{V}$ を可算無限集合 (自然数全体の集合と要素達を一対一対応させられる集合) とするとき，命題論理の論理式 (formula) の集合 $\mathbf{Pro}$ を以下のように定義する．

1. $\mathbf{V} \subseteq \mathbf{Pro}$ である．
2. $X \in \mathbf{Pro}$ のとき，$\neg X \in \mathbf{Pro}$ である．
3. $X, Y \in \mathbf{Pro}$ のとき，$X \wedge Y, X \vee Y, X \to Y \in \mathbf{Pro}$ である．
4. 以上のようにしてできるもののみが $\mathbf{Pro}$ の元である．

ここで 4 は省略する場合がある．また，常に真，あるいは偽であることを表す論理定数 $\mathbf{T}, \mathbf{F}$ などを付け加える場合もある．$\mathbf{V}$ の元を命題変数 (propositional variable) と呼ぶ．$|\mathbf{V}| = \aleph_0$(アレフゼロと読む．自然数全体の集合の濃度 (cardinal number)．この場合要するに変数の数が自然数と同じだけあるという意味である) としているのは，必要なだけいくらでも変数を用意することができるためである．しかしながら $\mathbf{Pro}$ の各

元，即ち各論理式にはそれぞれ有限個の命題変数しか現れないことに注意してほしい．命題変数以外の記号としては，後述するように補助的に用いる括弧以外には $\neg, \wedge, \vee, \to$ のみである．これら 4 つの記号を論理記号 (logical symbol) あるいは論理演算子 (logical operator)，論理結合子 (logical connective) などと呼ぶ．$\neg$ は単項演算子，$\wedge, \vee, \to$ は二項演算子である．

ここでは $\mathbf{V} = \{\underline{A}, \underline{A}_1, \ldots, \underline{B}.\underline{B}_1, \ldots\}$ としておく．$\mathbf{V}$ を取り換えても数学的理論には変化がないからである．すると論理式の例は $\underline{A} \wedge \underline{B}, (\underline{A} \wedge \underline{B}) \to (A \vee (\neg \underline{D}))$ などである．一方 $\underline{A} \neg \underline{B}, \wedge, \underline{A} \vee$ などは使っている記号は同じだが論理式ではない．

X と $\underline{A}$ は両方とも変数だが，両者の違いは，前者は数学的な議論を行うレベル (メタレベル) の変数であり，後者は数学的な議論の対象となるレベル (対象レベル) の記号のうち変数を表す記号であるということである．この違いは重要であり，詳しくは 2.2 で述べる．

古典論理の場合，論理演算子としてさらに $\leftrightarrow$(同値) や $\veebar$(排他的論理和) などを加えてもよいし，逆に $\wedge$ あるいは $\vee$ を減らすことも可能である．後で出てくるように意味的にはド・モルガンの法則を用いて $\neg$ と，$\vee$ あるいは $\wedge$ により $\wedge$ あるいは $\vee$ を表すことができるからである．同様に $\to$ を減らす，あるいは逆にこれと $\neg$ のみを用いる流儀もある．なお $\leftrightarrow$ や $\veebar$ は 4 つの論理演算子の一部を用いて表すことができる．Shoefer の棒記号 | という二項論理演算子を用いる方法もあり，ただ一つの演算子 | ですべての他の論理演算子を表すことができる．(論理回路についてご存知の方のみへの説明:) これは丁度，論理回路で 2 入力の NAND あるいは NOR のみがあれば全ての論理回路を構成可能であることに対応している．

論理式を以上のようにして帰納的に定義する理由は，こうすれば数学的に厳密に定義できるということに加えて，論理式について何かを証明するときに，論理式の構成に関する帰納法，あるいは構造帰納法 (structual induction) を使うことができるためである．これは非常に重要なポイントなのでよく認識してほしい．

例えば論理式についてのある性質 $P(-)$ がすべての論理式について成り立つことを示したいとする．そのためには，$P(X)$ がすべての命題変数 $X \in \mathbf{V}$ について成り立つことと (base case)，$P(X), P(Y)$ が成り立つと仮定した場合に (帰納法の仮定)，$P(\neg X)$, $P(X \wedge Y)$,$P(X \vee Y)$, $P(X \to Y)$ が成り立つことをすべての論理式 $X, Y \in \mathbf{Pro}$ に対して示せば (induction step) 十分である．自然数を 0 と，後者函数 (successor functions) により帰納的に定義することができ，その場合に自然数についての数学的帰納法が成り立つことと対比してみてほしい．

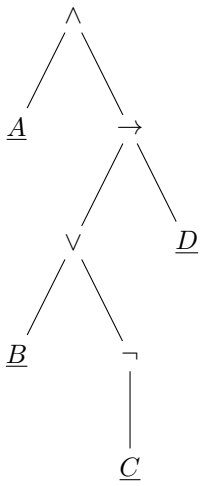
記号として $\wedge$ の代わりに $\&, \to$ の代わりに $\Rightarrow$ や $\supset$，$\neg$ の代わりに $\sim$ を使う場合がある[*2]．$\wedge$ は「かつ」(and)，$\vee$ は「または」(or)，$\neg$ は「〜でない」(not)，$\to$ は「ならば」(implies) という風に読んでよいし，いずれそのような意味を与えることを想定してはいるが，ここでは意味を未だ考えず，論理式を単なる記号の列，あるいはそれらから作られる木 (tree) として扱っていることに注意してほしい．

つまり数学的考察の対象とする論理式は記号列あるいは木である．両者の間には，正確に言えば前者の記号列から (コンパイラなどの言語処理系のように) 構文解析を行って木として表現したのが後者であるという違いと関係がある．しかしそれらの間の変換を比較的容易に行うことができるし，論理式についての証明中で，どちらかの形で表されていることを暗黙のうちに仮定して証明しても問題がない場合が多いため，どちらの形で表すことを仮定しているかを特に明記しない場合が多い．変換は，記号列や木の大きさの線形〜二乗程度の作業量である．

$$\underline{A} \wedge (\underline{B} \vee \neg \underline{C} \to \underline{D}), \tag{1}$$

$$\underline{A} \wedge ((\underline{B} \vee (\neg \underline{C})) \to \underline{D}), \tag{2}$$

[*2] 本稿では $\Rightarrow$ や $\Leftrightarrow$ はそれぞれメタレベル (2.2 参照) の「ならば」,「同値」(必要十分条件) として使用している．



ただし本稿に限らず紙面上に論理式を表記する場合には大部分の場合，記号列として表している．

■部分論理式　ある論理式 $X \in \mathbf{Pro}$ について，X と Y が等しい場合も含めて $Y \in \mathbf{Pro}$ が X の一部となっている場合がある．つまり X を定義 2.1 に従って構成する途中で Y が現れている場合である．このような場合，Y を X の部分論理式 (subformula) と呼ぶ．論理式 X の大きさは有限なので，X の部分論理式全体も有限個である．命題変数 A が論理式 X の部分論理式である場合，A は X に出現する (occur) と言う．ある論理式に出現するすべての命題変数の集合も有限である．例えば上 4 つは一番下の式の部分論理式である．

$$\underline{B}, \tag{3}$$
$$\underline{B} \vee \neg \underline{C} \to \underline{D}, \tag{4}$$
$$\underline{B} \vee \neg \underline{C}, \tag{5}$$
$$\underline{A} \wedge (\underline{B} \vee \neg \underline{C} \to \underline{D}), \tag{6}$$
$$\underline{A} \wedge (\underline{B} \vee \neg \underline{C} \to \underline{D}), \tag{7}$$

しかし以下はその式の部分論理式ではない．

$$\underline{E}, \tag{8}$$
$$\underline{C} \vee \neg \underline{B}, \tag{9}$$
$$\underline{A} \wedge (\underline{C} \vee \neg \underline{B} \to \underline{D}). \tag{10}$$

■補助記号　論理式を記号列として表す場合には，括弧を付けて曖昧性をなくす必要がある．例えば $\underline{A} \wedge \underline{B} \vee \neg \underline{C} \to \underline{D}$ は，$\underline{A} \wedge ((\underline{B} \vee (\neg \underline{C})) \to \underline{D})$，$(\underline{A} \wedge \underline{B}) \vee (\neg (\underline{C} \to \underline{D}))$ などの複数の可能性がある．

本稿では，$\neg$ が最も結合力が高く，$\wedge$ と $\vee$ がその次で，$\to$ が一番低いとすることで，括弧の数を減らして表す．ただし今の例の場合には，このようにしても $(\underline{A} \wedge (\underline{B} \vee (\neg \underline{C}))) \to \underline{D}$ と $((\underline{A} \wedge \underline{B}) \vee (\neg \underline{C})) \to \underline{D}$ の二通りの可能性があり，それぞれ括弧をできる限り省略した場合には $\underline{A} \wedge (\underline{B} \vee \neg \underline{C}) \to \underline{D}$ と $(\underline{A} \wedge \underline{B}) \vee \neg \underline{C} \to \underline{D}$ となる．論理演算子の結合力は文献によって異なる場合があるので注意すること．

括弧なしで $\wedge, \vee, \to$ がそれぞれ 3 つ以上連続して使われた場合にどう結合するか決めるのに，左右どちらに優先して結合するかを指定するのが一つの方法である．しかし本稿ではそうせず，括弧を付けて曖昧性をなくすこととする．ただし，後で見るように $\wedge$ と $\vee$ については意味上結合法則と交換法則が成り立つため，可換環の演算の $+$ や $\cdot$ のように括弧を付けずに表すこともある．

演習問題

1. 命題変数が 5 回以上出現する命題論理式を 3 つ以上示せ.
2. 使われている記号は命題論理式と同じだが，命題論理式ではない記号列を 3 つ以上示せ.

2.2 メタと対象

ここでメタ言語と対象言語について説明する．数理論理学のうち，古典命題論理の直接の数学的な研究対象は，**Pro** という集合とその元である．このような記号列，あるいは木から成る研究対象の集合を対象言語 (object language) と呼ぶ．一方，それらについて数学的な議論を行う際には自然言語である英語や日本語を用いる．これらの言語のことをメタ言語 (meta language) と呼ぶ．メタ言語まで記号化・(1 レベル上の) 対象化することも可能だし，実際行う場合もあり，その場合にはそれについて議論を行うのはメタ・メタ言語ということになる．ここではそれについてはこれ以上触れない.

数理論理学では対象言語とメタ言語をはっきり区別することが必要である．論理記号 $\neg, \wedge, \vee, \to$ や論理定数 $\mathbf{T}, \mathbf{F}$，補助記号 (,) は対象言語の記号である.

特に同じ用語を双方に対して用いる場合に，慣れるまで混乱を招きやすいため，区別が重要となる．例えば証明 (proof) という言葉は，メタ言語で書かれた数学的な証明を指すのに使われるし，対象言語である論理式の正しさを導き出すある種のデータのことを指すのにも使われる (後出)．後者は対象レベルの証明である (本稿では対象レベルについてなるべく証明図あるいは証明木という言葉を使うことにする)．また，定理 (theorem) という言葉は，メタ言語で書かれた数学的な定理を指すのに使われるし，対象言語である論理式のうち，それの証明図が存在して正しいと確認できる論理式を指すのにも使われる (これも後出，文脈から問題ない時には本稿ではなるべく定理スキーマなどの言葉を使うことにする)．メタ言語で書かれた数学的な定理の証明はメタ言語で書かれるし，対象言語の定理 (スキーマ) の証明図は対象レベルの言語 (あるいは木) で記述される.

また，記述中に現れる変数が，メタ変数なのか対象言語の変数なのかを区別することも重要である．例えば本稿では，命題論理で扱う命題変数は 対象言語の変数であり，$\underline{A}, \underline{B}_1$ などというように 下線_を付けて表記している．これは対象言語の変数であることを明示するためである．ただし論理演算子 $\neg, \wedge, \vee, \to$ は対象記号だが，_を付けていない．それに対し，本文中の X, Y, Z などの変数記号はメタ変数であり，これらに入るのは対象言語の論理式である．他にも本稿では Σ を記号の集合を表すメタ変数として用いている．また A という記号を Σ^* の部分集合を表すメタ変数として用いている．これは $\underline{A}$ という対象言語の変数とは別物である.

ただし，メタ変数と対象言語の変数の表記上の区別を常に厳密につけると煩わしい面もある．読者 (ともちろん著者) の意識の中ではっきりと区別が付いていれば十分なので，読者の文脈を読み取る力を信用して同じ記号を用いている教科書もよくある.

演習問題

3. ここまでに出現した，本文中の対象言語の変数とメタ変数を全て指摘せよ.

2.3 意味論

ここまでは論理式は単なる記号列に過ぎず，意味を持たなかった．今から論理式に意味を与えることを考える．一般に論理式などの記号列の意味についての数学的な理論を意味論 (semantics) と呼ぶ．本稿ではまず，意味論で論理式の正しさを定義することになる．一方これまでの記号列としての理論を構文論 (syntax) と呼ぶ．構文論的な正しさの定義をいずれ行い (こちらを先に行ってもよい)，両者が一致することを示すこととなる．

古典命題論理の意味論の場合，各論理式や命題変数に以下の集合の元を与えることを考える．[*3]

$$\mathbf{TV} \stackrel{\text{def}}{=} \{\mathbf{T}, \mathbf{F}\}$$

TV の元を真理値 (truth value)，あるいは真偽値と呼ぶ．**T** は真，**F** は偽を表している．**T**, **F** の代わりに $\top, \bot$ あるいは **true**,**false**，**1**,**0** などと表記する場合もある．このように二値の真理値を使うため，その場合には古典論理は二値論理とも言われる．

T, **F** と論理定数の T,F は別物なので注意してほしい．前者は対象言語の外にある数学的対象だし，後者はそれぞれ対象言語中の記号である．古典論理の意味は **TV** を用いて与えるのが普通なので，それを強調する場合には二値論理という言い方をする．これは **TV** が 2 つの元から成るためである．**TV** を他のものに変えて意味を付けることも可能であり，例えば **TV** の代わりに一般のブール代数 (boolean algebra) を用いることも可能である．**TV**(に演算を入れたもの) は自明 (一点集合) でない最も簡単なブール代数なのである．**TV** を用いて論理記号や論理式の意味を表すことを始めたのはイギリスの数学者 Boole であり，この名前が付いている．ただし古典命題論理の場合には，**TV** を一般化しても目新しい結果を得られるわけではなく，**TV** のみを考えていても同じである．

さらに，**TV** やブール代数ではなく，論理演算子 $\neg, \wedge, \vee, \rightarrow$ や，他の増やしたあるいは減らした論理演算子の意味を与えられるような別の代数を用いて論理式の意味を与えることも可能であり，そうすると一般には古典論理とは異なる別の論理となる．例えば Heyting 代数を用いると直観論理と呼ばれる論理となる．直観論理で用いる論理演算子は定義 2.1 と同じである．ただしこの場合には古典論理の場合と異なり，4 つの論理演算子の一部を他のものを使って表すことはできない[*4]．そういった論理では，一般に意味論で用いられる真偽値が 3 つ以上の値から成るために多値論理とも言う．

2.4 真理函数

まず，各論理演算子 $\neg, \wedge, \vee, \rightarrow$ の意味を函数として表すことを考える．これは以下のような論理式の意味を表すことと等価である:$\neg\underline{A}, \underline{A} \wedge \underline{B}, \underline{A} \vee \underline{B}, \underline{A} \rightarrow \underline{B}$．右側に参考のため $\veebar$ と $\leftrightarrow$ を付け加えておいた．例えば表の下から 2 行目は，$\underline{A}$ の値が **T**，$\underline{B}$ の値が **F** のとき，各論理式の値がどのようなものになるかを表している．ただし $\neg$ は 1 引数だが表の数を減らすために手抜きをして他の演算子と同じ表に入れている．これが $\underline{B}$ の値に影響されないことを読み取ってほしい．このような表を真理値表 (truth value table) と呼ぶ．

基本的な 4 つの論理演算子の真理値表のうち，$\neg, \wedge, \vee$ については比較的納得できるのではないだろうか[*5]．しかし $\rightarrow$ についてはいかがだろう．$\underline{A}$ が成り立っていない場合に常に **T** ということでよいのだろうか．たと

[*3] **TV** という表記法を本稿では用いているが，これは一般的なものではない．
[*4] $\neg$ を $\rightarrow$ と $\bot$ で表すことは可能である．
[*5] 実は $\vee$ については $\rightarrow$ に次いで問題がある．

$\underline{A}$	$\underline{B}$	$f_{\neg}(\underline{A})$ $\neg\underline{A}$	$f_{\wedge}(\underline{A},\underline{B})$ $\underline{A}\wedge\underline{B}$	$f_{\vee}(\underline{A},\underline{B})$ $\underline{A}\vee\underline{B}$	$f_{\rightarrow}(\underline{A},\underline{B})$ $\underline{A}\rightarrow\underline{B}$	$\underline{A}\veebar\underline{B}$	$\underline{A}\leftrightarrow\underline{B}$
F	**F**	**T**	**F**	**F**	**T**	**F**	**T**
F	**T**	**T**	**F**	**T**	**T**	**T**	**F**
T	**F**	**F**	**F**	**T**	**F**	**T**	**F**
T	**T**	**F**	**T**	**T**	**T**	**F**	**T**

表 1　論理演算子の真理値表

えばこれは，$\underline{A}$ と $\underline{B}$ にまったく関係がない場合，(人によっては) 違和感を生じる場合がある．例えば「1=2 ならば，今日の天候は晴れである」の真偽値は，実際の天候に関わらず T となる．そのようなことを言明して意味があるのだろうか，そのように「ならば」を使ってよいのだろうか，という感想を持つ人もいるだろう．つまり，上の真理値表の $\rightarrow$ は，通常人間が使っている「ならば」とは異なるものかもしれない．しかし，条件から結論を導き出すときに正しい推論を行うという意味では実はこれで十分である．このことは後で示すことにする．また，論理式の真偽値として **TV** を用いる場合，これ以外に「ならば」として使えそうな真理函数があるかというと，調べてみるとわかるようにないのである．二値論理の $\rightarrow$ はこのような特性を持った「ならば」であり，実質含意 (material implication) と呼ばれる．実はもっと「ちゃんとした」ならばを実現しようという研究には様々なものがあるが，ここでは触れない．

真理値表は，他の一般の論理式についても書くことができる．そのためには，上記の 4 つの論理演算子についての真理値表があれば十分である．なぜならそれらのみを用いて，定義 2.1 のようにしてできるもののみが論理式だからである．

例えば $E \equiv \underline{A}\wedge\underline{B}\rightarrow\underline{C}$ という論理式の真理値表は以下のようなものとなる．ここで，論理式の構造に沿っ

$\underline{A}$	$\underline{B}$	$\underline{C}$	$\underline{A}\wedge\underline{B}$	$\underline{A}\wedge\underline{B}\rightarrow\underline{C}$
F	**F**	**F**	**F**	**T**
F	**F**	**T**	**F**	**T**
F	**T**	**F**	**F**	**T**
F	**T**	**T**	**F**	**T**
T	**F**	**F**	**F**	**T**
T	**F**	**T**	**F**	**T**
T	**T**	**F**	**T**	**F**
T	**T**	**T**	**T**	**T**

て帰納的に部分論理式の真理値が定まり，それによって段階的に一つ上のレベルの部分論理式の真理値が論理演算子の真理値表により定まることに注意してほしい．つまり論理式の真理値は帰納的に定まる．真理値表は，函数を定めていると考えることもできる．つまり今の例では函数 $f_E : \mathbf{TV}\times\mathbf{TV}\times\mathbf{TV}\rightarrow\mathbf{TV}$ を定めている．このような函数を真理函数 (truth-value function) という．多値論理の場合には真理値表や真理函数もそれに応じたものとなる．真理値の数が有限個であれば有限の大きさの真理値表として真理函数を表せる．

演算子に対する真理値表の各列の一番上の項目に，$\underline{A}\wedge\underline{B}$ に重ねて $f_{\wedge}(\underline{A},\underline{B})$ などと記している．これは各論理演算子の表す真理値上での函数に名前をつけて書き込んだものであり，厳密には各論理演算子そのものが

函数なのではなく，それに対応する真理函数が定義されるということに対応している．

上のようにして個々の論理式 $X \in \mathbf{Pro}$ には真理函数が対応する．論理式 X に出現する命題変数全体の集合を $\mathrm{FV}(X)$ で表すと，$f_X : \mathbf{TV}^{|\mathrm{FY}(X)|} \to \mathbf{TV}$ という函数となる．少し考えると分かるように，二値命題論理の場合，$|\mathbf{TV}| = 2$ なので n 引数の真理函数は一般に 2^{2^n} 種類存在する．ここで一つ重要なことは，どのような真理函数であっても，それを 4 つの基本的な論理演算子を組み合わせて表現できるということである (論理式の標準形のところで説明する)．二値論理以外の場合にはこれは一般には成り立たない．

例えば $\underline{A} \leftrightarrow \underline{B}$ や $\underline{A} \veebar \underline{B}$ は $(\underline{A} \to \underline{B}) \wedge (\underline{B} \to \underline{A})$，$(\underline{A} \wedge \underline{B}) \vee (\neg\underline{A} \wedge \neg\underline{B})$ などとして表すことができる．

また，$\neg\neg\underline{A}$ と $\underline{A}$，$\neg\underline{A} \wedge \neg\underline{B}$ と $\neg(\underline{A} \vee \underline{B})$，$\underline{A} \wedge \underline{B}$ と $\underline{B} \wedge \underline{A}$，$\underline{A} \wedge (\underline{B} \wedge \underline{C})$ と $(\underline{A} \wedge \underline{B}) \wedge \underline{C}$ 等の真理値表が等しいこと，$\underline{A} \to \underline{A}$ と $\underline{A} \wedge \neg\underline{A}$ の真理値が 常に $\mathbf{T}$ であること等を確認してほしい．なお，論理定数 T と F の値はそれぞれ常に $\mathbf{T}$ と $\mathbf{F}$ である．

演習問題

以下の論理式の真理値表を書け．

4. $\neg\neg\underline{A}$
5. $\underline{A}$
6. $\neg\underline{A} \wedge \neg\underline{B}$
7. $\neg(\underline{A} \vee \underline{B})$
8. $\underline{A} \wedge \underline{B}$
9. $\underline{B} \wedge \underline{A}$
10. $\underline{A} \wedge (\underline{B} \wedge \underline{C})$
11. $(\underline{A} \wedge \underline{B}) \wedge \underline{C}$
12. $\underline{A} \to \underline{A}$
13. $\underline{A} \vee \neg\underline{A}$
14. $\underline{A} \leftrightarrow \underline{B}$
15. $\underline{A} \veebar \underline{B}$

2.5 付値

真理値表により論理式の真理函数を表現でき，それが元の論理式の性質を反映していると考えられることは前の小節で見た．ただし，真理函数は論理式に出現する命題変数によって引数の数が異なることになる．これを，一般にどのような命題論理式の真偽値も求められる形式に変更することを考え，$\phi : \mathbf{V} \to \mathbf{TV}$ を導入する．このような数を付値 (valuation) と呼ぶ．ここでは付置全体の集合を $\mathcal{F}$ で表すことにする．函数空間を表す通常の記法を使うと $\mathcal{F} = \mathbf{TV}^{\mathbf{V}}$ である．付値 $\phi \in \mathcal{F}$ は元々は命題変数の真偽値を定めるものであるが，$\mathbf{Pro}$ に対し拡張することが可能である．それは次のようにして行う．$X \in \mathbf{Pro}$ とする．

1. $X \in \mathbf{V}$ のとき，$\phi'(X) \stackrel{\mathrm{def}}{=} \phi(X)$.
2. $X \equiv \neg Y$ のとき，$\phi'(X) \stackrel{\mathrm{def}}{=} f_{\neg}\phi'(Y)$.
3. $X \equiv Y \wedge Z$ のとき，$\phi'(X) \stackrel{\mathrm{def}}{=} f_{\wedge}(\phi'(Y), \phi'(Z))$.
4. $X \equiv Y \vee Z$ のとき，$\phi'(X) \stackrel{\mathrm{def}}{=} f_{\vee}(\phi'(Y), \phi'(Z))$.

5. $X \equiv Y \to Z$ のとき，$\phi'(X) \stackrel{\text{def}}{=} f_{\to}(\phi'(Y), \phi'(Z))$.
6. $X \equiv \mathrm{T}$ のとき，$\phi'(X) \stackrel{\text{def}}{=} \mathbf{T}$.
7. $X \equiv \mathrm{F}$ のとき，$\phi'(X) \stackrel{\text{def}}{=} \mathbf{F}$.

ここで $\equiv$ は，両辺が記号列あるいは木として等しいということを表している．また下の 2 行は論理定数 T と F を導入した場合のみである．この場合，T と F を命題変数と同じ扱いとし，ただし付値 ϕ としての $\phi(\mathrm{T}) = \mathbf{T}, \phi(\mathrm{F}) = \mathbf{F}$ となるもののみを考えてもよい．

重要なのは，上の場合分けが論理式の形のすべての場合を一意的に尽くしていて，しかも各定義の右側の論理式は，X よりも簡単なものになっている (正確には，論理式の記号列の長さが短いもの，あるいは木の大きさが小さいものになっている) ため，これにより **Pro** の任意の元 X に対し $\phi'(X)$ が定義されるということである．つまり付値が一つ与えられれば，すべての論理式の真偽値が定まる．このことを論理式の構成による帰納法できちんと示すことも可能だが，単純だし直感的にはほぼ明らかなので帰納法の例題として用いる場合以外にはわざわざ示さない場合が多い．

$X \in \mathbf{V}$ であれば $\phi'(X) = \phi(X)$ が成り立つので，拡張した ϕ' の代わりに通常元の ϕ を用いて表す．

演習問題

16. 任意の $A \in \mathbf{V}$ に対し $\phi_{\mathbf{T}}(A) \stackrel{\text{def}}{=} \mathbf{T}, \phi_{\mathbf{F}}(A) \stackrel{\text{def}}{=} \mathbf{F}$ として $\phi_{\mathbf{T}}, \phi_{\mathbf{F}} \in \mathcal{F}$ を定義する．2.4 の演習問題の各論理式 X について，$\phi_{\mathbf{T}}(X), \phi_{\mathbf{F}}(X)$ を計算せよ．
17. 同じ各論理式 X について，それぞれの $\phi(X) = \mathbf{T}$ となる $\phi \in \mathcal{F}$ は存在するか．具体的に示すか存在しないことを示せ．
18. $\phi(X) = \mathbf{F}$ の場合はどうか．

2.6 モデル

ある論理式 $X \in \mathbf{Pro}$ と，付値 ϕ について $\phi(X) = \mathbf{T}$ となるとき，ϕ は X を充足する (satisfy) と言う．またこのとき ϕ を X のモデル (model) と呼び，$\phi \models X$ と書く．そうでない場合には $\phi \not\models X$ と書く．二値論理で $\mathbf{TV} = \{\mathbf{T}, \mathbf{F}\}$ のときには後者の場合，$\phi(X) = \mathbf{F}$ である．

次にモデルの定義を論理式の集合に拡張する．$G \subseteq \mathbf{Pro}$ とする．G は無限集合かもしれない．付値 ϕ が，G に属するすべての論理式のモデルになっているとき，ϕ は G のモデルであると言い，$\phi \models G$ と表記する．これは G が有限集合のときには，G のすべての論理式を $\wedge$ で結んでできる論理式のモデルになっていることと同値である．

演習問題

19. 2.4 の演習問題の各論理式について，そのモデルを具体的に示すか，存在しないことを示せ

2.7 トートロジー，充足可能性

すべての付値がモデルであるような論理式 $X \in \mathbf{Pro}$ を恒真 (valid) であると言う．特に命題論理の恒真な論理式をトートロジー (tautology) と呼ぶ．トートロジーとは元々同義語反復という意味である．

逆にすべての付値 $\phi \in \mathcal{F}$ により $\phi(X) = \mathbf{F}$ となる論理式 $X \in \mathbf{Pro}$ のことを恒偽式という．恒偽式でない

論理式，即ちある付値 ϕ が論理式 X を充足するとき，X を充足可能 (satisfiable) であるという．そうでないとき，充足不能 (unsatisfiable) という．X がトートロジーであることと，$\neg X$ が充足不能であることは同値である (なぜか?).

ある論理式 $X \in \mathbf{Pro}$ がトートロジーであるかどうかは機械的に (アルゴリズムによって) 求めることができる．これを，決定可能であるという言い方をする (2.8 参照)．付値の情報を完全に与えるにはすべての命題変数の真偽値を与える必要があるが，具体的に X が与えられれば，X の真偽値に影響を与えるのは $\mathrm{FV}(X)$ の値のみであり，有限の大きさの X の真理値表を作成してすべての結果が $\mathbf{T}$ であることを確認すればよいからである．

また同様に，X が充足可能であるかどうかも決定可能であることがわかる．一般に二値命題論理の論理式の充足可能性を調べる問題を SAT と呼んでいる．しかし，以上のような方法で充足可能性を調べるとすると，$2^{|\mathrm{FV}(X)|}$ 種類の入力に対して X の真偽値を求める必要があり，原理的には結論を求められるとしても，$|\mathrm{FV}(X)|$が少し大きくなるととても実際的な時間で計算を行うことが不可能になってしまう．実は，SAT は効率的な解法がないと一般に信じられている決定問題である．

演習問題

20. 2.4 の演習問題の論理式について，トートロジーであるものと充足可能なものがそれぞれどれか答えよ．
21. $X \in \mathbf{Pro}$ がトートロジーであることと，$\neg X$ が充足不能であることが同値であることを示せ．
22*. プログラミング言語 LISP を用い，命題論理式をリストで表す方法を定め，命題論理式を具体的に与えた時にそれの真理値表を求めるプログラムを記述せよ．(「情報処理の方法と演習 A」の問題より．*付きの問題は難しかったり時間がかかるため，興味のある人のみが解いてみるのが良さそう)

2.8 決定問題と決定可能性

論理式を記号列として表す場合には，先に見たように記号列の中に，論理式を表すものとそうでないものがある．ここで重要なことの一つは，記号列が渡された場合に，それが論理式を表すかどうかを機械的に，つまりアルゴリズムにより判定可能であるということがある．正式には決定可能 (decidable) であると言い，これは以下のように定義される概念である．

一般に，空でない有限集合 Σ があるとき，Σ の元から成る長さ 0 以上の有限列全体の集合を Σ^* と書く (これはオートマトンや言語理論の文脈でよく使われる記法である)．またこの場合，Σ の元を記号 (symbol) または文字と呼ぶ．長さ 0 の記号列を ϵ と書く場合が多い．一方長さ 1 以上とした場合には Σ^+ と記す．即ち $\Sigma^+ = \Sigma^* \backslash \{\epsilon\}$ である．

$A \subseteq \Sigma^*$ を決定問題 (decision problem) と言う．即ち Σ^* の元に対し，それが A に属するかどうかを求めるという問題である．決定問題に対し Turing 機械 (TM)M により答えを与えることを考える．A の元をテープに書いてその両側を M のブランク記号の列とし，その元の左端 (元=ϵ のときはヘッドでブランク記号を指しておく) にヘッドを置いて動作を開始した場合には停止して yes がテープに書かれる結果となり，同様に A の補集合の元をテープに書いた場合にはやはり停止して no が結果となるような，ある一つの TM M が存在する場合に A は (M により) 決定可能と言う．いずれの場合にも M は必ず停止しなければならない．ただしここでは簡単のためにブランク記号 $\notin \Sigma$ で，M の記号は Σ を含むとしている．

この条件についてやや厳密さを欠いた説明をすれば，記号列が A の元かどうかを判定するプログラムが存

在する，と言い換えることができる．Σ を一つ固定した場合，決定可能な集合全体の濃度は $\aleph_0$ である (なぜか？)．TM M に対し，$S_M \stackrel{\text{def}}{=} \{\sigma \in \Sigma^* \mid$ 入力σに対しMが停止して yes がテープに書かれる$\}$ とする．この記法を使うと，$A \subseteq \Sigma^*$ 決定可能 $\stackrel{\text{def}}{\Leftrightarrow}$ 全ての $\sigma \in \Sigma^*$ に対し必ず停止する TM M が存在して $A = S_M$ となる．

決定可能性を $A \subseteq \mathbf{N}$ に対して定義することもできる．これは，$s \in \Sigma^*$ と $n \in \mathbf{N}$ を次のようにして一対一対応付けられるからである．$|\Sigma| = m$ とする．まず全単射 $\phi' : \Sigma \to \{0, \ldots, m-1\}$ を一つ選ぶ．$\phi(\epsilon) = 0, \phi(sx) = m\sigma(s) + \phi'(x)$ として $\phi : \Sigma^* \to \mathbf{N}$ を定義する．なお $x \in \Sigma$ であり，sx は列 s の末尾に記号 x を付加してできる列である．つまり ϕ は Σ による m 進表現を与える函数である．s から n への対応は，$n = \phi(s) + \frac{m^{|s|}-1}{m-1}$ により与えられる．ここで $|s|$ は s の長さである．

ただし $m = 1$ の場合には $n = |s|$ という対応とする．この場合，s を n の 1 進表現と言う．

例えば偶数全体の集合，奇数全体の集合，すべての素数からなる集合等はすべて決定可能である．同様に，記号列でコーディング出来る他のデータに対しても決定可能性の定義を行うことができる．

あるいは再帰函数 (recursive function) により，$\mathbf{N}$ の部分集合が決定可能であることを TM を介さずに直接定義し，2 つの定義が同等であることを証明することもできる．

決定可能な集合の補集合，2 つの決定可能な集合の和集合と交集合はやはり決定可能である (なぜか?)．つまり Σ^* や $\mathbf{N}$ の部分集合のうち，決定可能な集合全体はブール代数となる．

Hilbert の数学の算術化のプログラムでは，当初からアルゴリズム (に相当する) 概念が入っており，数学の定理の集合が論理式の集合の中で決定可能であることを示すことも目標としていた．もちろん当時は決定可能という概念そのものがまだあやふやな段階であり，今日見られるような定義を与えていた訳ではないし，自然数論を展開できるような数学を行う場合，定理の集合は決定可能とはならないことが結局は示された訳で，Hilbert の当初の目論みはそのままの形では残念ながら外れてしまった訳である．しかし，決定可能性と次の半決定可能性は数理論理学において重要な役割を果たす概念である．

■注意: 決定可能という言葉は数理論理学において他の文脈でも使われる場合がある．この資料の後半に出てくる一階述語論理の閉論理式 E について，$\vdash E$ か $\vdash \neg E$(E の否定) のどちらかが理論 Th から推論規則により証明できる時，Th において E は決定可能 (命題) であると言う．これは決定問題 $A \subseteq \Sigma^*$ の決定可能性とは異なる概念であり，混同しないこと．

演習問題

23. Σ^* の濃度を答えよ．
24. 2^{Σ^*} の濃度を答えよ．
25. 空集合 $\emptyset$ や全体集合 Σ^* や $\mathbf{N}$ の中で決定可能かどうか根拠付きで答えよ．
26. Σ^* や $\mathbf{N}$ のただ一つの元から成る集合がそれらの中で決定可能かどうか根拠付きで答えよ．
27. 決定可能な集合が少なくとも可算無限個存在することを示せ．
28. 決定可能な集合の濃度が高々可算無限であることを示せ．
29. 偶数全体の集合 Even $\subseteq \mathbf{N}$ が決定可能であることを示せ．
30. 素数全体の集合 Prime $\subseteq \mathbf{N}$ が決定可能であることを示せ．
31. 2 つの決定可能集合 $X, Y \subseteq \mathbf{N}$ の和集合 $X \cup Y$ が決定可能であることを示せ．
32. 31 で $X \cap Y$ が決定可能であることを示せ．
33. 決定可能集合 $X \subseteq \mathbf{N}$ の補集合 $\mathbf{N} \backslash X$ が決定可能であることを示せ．

34*. 再帰函数の定義を調べよ.

35*. ブール代数の定義を調べよ.(これらの言葉が本文中に現れる.調べない場合,現れている文を無視して構わない.)

2.9 帰納的可算性と半決定可能性

やや厳密さを欠く書き方をすると,$A \subseteq \Sigma^*$ が帰納的可算 (recursively enumerable)(R.E.) であるとは,有限長の記号列を 0 回以上出力し続けるプログラム P が存在し,P が出力する記号列全体と A が一致するときである.P は 0 個以上有限個の記号列を出力後,無限ループ等に入り,その後何も出力しなくなってもよいし,あるいは無限に記号列を出力し続けてもよい.また,出力する記号列に重複があっても構わない[*6].帰納的可算な集合のことを半決定可能 (semidecidable) な集合とも言う.Σ を一つ固定した場合,帰納的可算な集合全体の濃度はやはり $\aleph_0$ である.一方 Σ^* の部分集合全体の濃度は $\aleph$ なので,帰納的可算でない集合の方がずっと多いことがわかる.

なお,$A \subseteq \mathbf{N}$ に対しても帰納的可算性を同様に定義することができる.

A が決定可能であることと,A と A の補集合が両方とも半決定可能であることは同値である (なぜか?).半決定可能な集合の補集合は半決定可能であるとは限らないことが,半決定可能だが決定可能ではない集合の存在から導かれる.後者の例をある条件を満たす要素の集合として与える方法はまったく自明ではなく,Turing が対角線論法と呼ばれる方法を用いて示したものである (ここでは述べない).ただし 2 つの半決定可能な集合の和集合と交集合はやはり半決定可能である (なぜか?).

帰納的可算性を自然数の部分集合や,記号列でコーディング出来る他のデータに対して定義することもやはり可能である.

演習問題

36. A が決定可能であれば A が半決定可能であることを示せ.
37. A と A の補集合が両方とも半決定可能であるとき,A は決定可能であることを示せ.
38. 半決定可能な集合全体の集合の濃度が高々可算無限であることを示せ.
39. 半決定可能でない Σ^* の部分集合が存在することを示せ.(ヒント:Σ^* の部分集合全体の集合の濃度と,半決定可能な集合全体の集合の濃度を比較せよ.)
40. 2 つの半決定可能集合の和集合が半決定可能であることを示せ.
41. 2 つの半決定可能集合の交集合が半決定可能であることを示せ.

2.10 非決定性チューリング機械

計算機科学と関わりがある SAT に言及することとなったので,それの周辺について説明しておくことにする.ただし 2.10〜2.12 は本稿での論理 (logic) の理解のためには必ずしも必要ではないので,飛ばしてもよい.

非決定性 (nondeterministic) チューリング機械 (NDTM) というのは通常の TM の拡張の一つである.計算の各ステップで有限個の可能性の内一つを実行していく.最悪の場合,ステップ数と共に可能性は指数函数的に増えてゆくことになる.それら全ての計算の可能性のうちの一つが停止して yes という答えを出せば,全

[*6] 重複を許しても許さなくても同等の定義となる.

体の答えが yes となるような仮想的な計算機が NDTM である．通常の TM，あるいは別の言い方をすれば各ステップで可能性が常に一つしかない NDTM を決定性 (deterministic) チューリング機械 (DTM) という．NDTM は現状あくまで理論的な存在であり，現実の計算機でいえば，並列計算機の各 CPU に各可能性を実行させた場合に近い動作を行うものである．現実に実装できたとしても可能性の数が (入力データのサイズに対し) 指数函数的に増大するために CPU の数がすぐに不足してしまう[*7]．

このことは，DTM が von Neumann 型計算機[*8]という形で現実に効率よく実装されているのとは対照的である．

「効率的に」という言葉は決定問題を分類する際にテクニカルタームとして用いられる．決定問題 A を正しく判定する機械 M があるとする．M の入力データのサイズを n としたときに，ある $c, d \geq 0$ により計算時間 (M のステップ数) が $n^c + d$ 以下となるとする．これを指して，A は M により多項式時間 (polynomial time) 決定可能であると言う．A が M により (理論的な意味で) 効率的に解けるというのは，M が DTM で，M により A が多項式時間で決定可能な場合のことである．

DTM M が存在して M により多項式時間決定可能となるような決定問題全体の集合をクラス P と呼ぶ．同様に NDTM により多項式時間決定可能な決定問題全体の集合をクラス NP と呼ぶ．

クラス P$\subseteq$ クラス NP である．等号が成り立つかどうかはまだわかっていないが，成り立たないことが強く予想されている．これを P$\neq$NP 予想といい，クレイ研究所のミレニアム問題の一つである．

大まかに言えば，クラス P に属さない問題は実際的には入力が少し大きくなると (現在使われている動作原理による) 計算機で解けなくなる問題である．クラス P に属したとしても，多項式の次数が大きければ計算の手間は膨大になり得るが，属さなければ入力が大きい場合，まったく解けないと言える．

一方直観的に言えば，クラス NP に属する問題は，問題の解き方 (あるいは解ける証拠) を与えられると，それが正しいことを多項式時間程度で確認できるような問題である．例えば SAT はクラス NP に属す問題であり，論理式 X が充足可能であることの証拠とはこの場合，具体的な X のモデル ϕ のことである．ただし FV(X) についてのみ値を定めればよい．実際に具体的にこのような ϕ が与えられれば，$\phi(X)$ の真偽値を求めてそれが **T** であることを確認するのは簡単であり，X の大きさの 2〜3 乗程度の手間で求められる．

SAT は決定問題であり，答えは yes か no かのどちらかなので，それと論理式 X の具体的な充足方法 ϕ を求める問題 (これはそのままでは決定問題ではない) とではギャップがあるように思うかもしれない．しかし後者の問題は，SAT を多項式時間で解くアルゴリズムが存在すれば，やはり多項式時間で解けるのである．具体的には，X に対し SAT を解いて答えが yes であれば，命題変数の一つ A を **T** と置いて X を簡単化し，それに対して SAT を適用し，答えが yes であれば A の値を **T** とし，no であれば A の値を **F** とする (no の場合，A を **F** として X が充足可能となるのはなぜか?)．これを繰り返してすべての命題変数の真偽値を求める．このようにしても多項式時間で具体的なモデルを求められる．

このように，P$\neq$NP 予想とは，問題の答え (あるいはその証拠) をヒントなしで求めるのは，答えがわかっていてそれが正しいことを確かめるよりも真に難しいという常識的にはほとんど当然と思ってしまうことを数学的に表したものであるとも言える．しかしながらまだそれを証明できていないのである．

演習問題

42. P$\subseteq$NP を示せ．

[*7] いわゆる分子計算機だとある程度の範囲で実現可能性があるかもしれない．

[*8] この名称で呼ぶのは問題があるとも言われている．

43. SAT∈NP を示せ.

2.11 多項式時間帰着可能性

ある決定問題 $A \subseteq \Sigma^*$ を別の決定問題 $B \subseteq \Sigma^*$ に帰着できる場合がある．$x \in \Sigma^*$ のときに，$x \in A$ かどうか調べるとする．テープ上に x を書き込み，これを機械 M により処理すると必ず停止してテープ上に記号列 $f_M(x)$ が得られるとする．ここで f_M は機械 M が表す函数である．そして $X \in A$ と $f_M(x) \in B$ が同値となるとする．このとき，A は M により B に帰着 (reduce) される，という．さらに M に x を与えた場合の計算時間が前の小節のように x の大きさの多項式で抑えられるとする．このとき，A は M により B に多項式時間で帰着される，という．また，そのような DTM M が存在するとき，A は B に多項式時間帰着可能 (polynomial-time reducible) であるという．

一般に，A を B に M で帰着できる場合には，ある意味 A の方が B よりも簡単であることを表している．ただしこれは M がどのようなものかが問題であり，どのような M を使ってもよければ，任意の決定可能な問題 A を，yes と no の両方の結果があり得るような他の決定問題に帰着することが可能である (なぜか?)．したがって M を制限することが必要であり，多項式時間帰着可能とい うのは意味のある制限になっている．

演習問題

44. 決定問題 A を恒等 $\mathrm{TM}_{\mathrm{id}}$[*9](すぐに停止して入力列をそのまま出力列とする $\mathrm{TM}_{\mathrm{ia}}$) により A 自身に帰着できることを示せ.
45. 決定可能な決定問題 A を一つ固定する．$\{0\} \subseteq \Sigma^*$(列 0 のみから成る集合) に帰着できる M_A が存在することを示せ．簡単のために 0 や yes,no の文字がどの元であると仮定してよい.

46*. SAT は 3SAT[*10]に多項式時間帰着可能であることを示せ.

2.12 NP 困難性，NP 完全性

任意の決定問題 $A \in$ class NP を $B \subseteq \Sigma^*$ に多項式時間帰着可能な場合，B は NP 困難 (NP hard) であると言う[*11]．つまり B はクラス NP のどの問題よりも難しい問題である．ここで帰着のための計算時間を多項式時間に限定しているところがポイントである．また B が同時にクラス NP にも属している場合，B は NP 完全 (NP complete) であると言う．これは，B が NP 問題のうちで一番難しい類の問題であることを表している.

歴史的には NP 完全であることを示された最初の具体的決定問題が SAT である．これは Cook により示された．一つの決定問題 B が NP 完全であることが示されたとする．他の NP に属する決定問題 C に B が多項式時間帰着可能であれば，C も NP 完全であると言える (なぜか?)．そのようにして何千という問題が NP 完全であることが示された．部分和問題，ハミルトン閉路問題などという名称を聞かれた人もいるだろう．そういった問題が SAT 以外の NP 完全問題として有名である (ただしこれらの問題の一部はそのままでは決定問題ではなく，同じ土俵の上で論じるには決定問題への変形が必要である)．それらのうち一つでも多項式時

[*9] 一般的な用語ではない

[*10] $(\underline{A_1} \vee \neg \underline{A_2} \vee \underline{A_3}) \wedge \cdots$ のように，1 個以上 3 個以下のリテラル (命題変数かそれに ¬ が 1 つ付いたもの) を ∨ で結んだものを，0 個以上 ∧ で結んだ形の命題論理式について充足可能性を考える決定問題．つまり論理式の形を制限した SAT 問題.

[*11] ここで帰着の仕方は実行時間が多項式時間で抑えられればよく，A に依存してよい.

間で計算できないということを証明できれば P≠NP の証明となる．

演習問題

47. NP 困難な決定問題 $A \subseteq \Sigma^*$ を決定問題 $B \subseteq \Sigma^*$ に多項式時間帰着できるとする．この時，B も NP 困難であることを示せ．

2.13 論理的帰結

この節では意味論的に論理的帰結を定義し，それにより二値論理の $\rightarrow$ が正しい推論を行うための条件を満たしていることを示す．ただしこれは結局，数学での「ならば」が実質含意であるということを示しているとも言えるかもしれない．

定義 論理的帰結

$X \in \mathbf{Pro}, G \subseteq \mathbf{Pro}$ とする．すべての付値 ϕ に対し，ϕ が G のモデルであれば ϕ が X のモデルであるとき，X を G の論理的帰結 (logical consequence) と言う．

上記の論理的帰結は意味論的帰結とも言う．X が G の論理的帰結のときに $G \models X$ と表記する．そうでない場合には $G \not\models X$ と表記する．また，特に G が有限集合のときには G を論理式の列として記し，それを表すメタ変数としては Γ を用いることにする．特に X が $G = \emptyset$(空集合)，Γ で言えば空列の論理的帰結であることと，X がトートロジーであることは同値となる．

例:$\underline{A}, \underline{B} \models \underline{A} \wedge (\underline{B} \vee \underline{C})$ が成り立つことと，$\underline{A} \not\models \underline{B} \vee \underline{C}$, $\models \underline{A} \vee \neg \underline{A}$ を示せ．

論理的帰結 $G \models X$ は，G を仮定した場合にそれらから X が導かれることを表している．つまり G から X を結論してよいということである．

次の補題が成り立つ．

補題 1. $G \subseteq \mathbf{Pro}$, $X, Y \in \mathbf{Pro}$ とする．$G \cup \{X\} \models Y$ と $G \models X \rightarrow Y$ は同値である．

証明: $(\Rightarrow)$ ϕ を任意の付値として，$\phi \models G$ のときに $\phi \models X \rightarrow Y$ を示す．$\phi \not\models X$ の場合と $\phi \models X$ の場合に場合分けする．$\phi \not\models X$ であれば，$\rightarrow$ の真理値表から $\phi \models X \rightarrow Y$ である．次に $\phi \models X$ であるとする．$G \cup \{X\} \models Y$ の定義から $\phi \models G \cup \{X\}$ ならば $\phi \models Y$ なので，$\phi \models G$ と $\phi \models X$ を併せると必ず $\phi \models Y$ である．従って $\phi \models Y$ であり，やはり $\rightarrow$ の真理値表から $\phi \models X \rightarrow Y$ である．

$(\Leftarrow)$ ϕ を $\phi \models G \cup \{X\}$ となるような付値として，$\phi \models Y$ を示す．このとき $\phi \models G$ かつ $\phi \models X$ である．$G \models X \rightarrow Y$ の定義から $\phi \models X \rightarrow Y$ である．$\rightarrow$ の真理値表から $\phi \models Y$ がいえる． □

系 2. $X, Y \in \mathbf{Pro}$ とする．$X \models Y$ と $\models X \rightarrow Y$ は同値である．

証明: 上の補題で $G = \emptyset$ とした場合である． □

演習問題

48. $X \models Y$ と $\models X \rightarrow Y$ がそれぞれ成り立つかどうかを，X と Y に具体的な論理式を当てはめて確認せよ．論理式の組み合わせは 3 種類以上について確認すること．

2.14 論理的同値

$X, Y \in \mathbf{Pro}$ の真理函数が等しいとき，言い換えるとすべての付値 ϕ に対し $\phi \models X$ と $\phi \models Y$ が同値であるとき，X と Y は論理的同値 (logically equivalent)，あるいは意味論的同値 (semantically equivalent) であると言う．論理的同値を表す記号は統一されていないが本稿では $X \models\!\dashv Y$ と表記することにする．

補題 3. $X \models\!\dashv Y$ の必要十分条件は，$X \models Y$ かつ $Y \models X$ であることである．

補題 4. $X \models\!\dashv Y$ と $\models X \leftrightarrow Y$ は同値である．

証明略.
例:

1. $X \to Y \models\!\dashv \neg X \vee Y$
2. $\neg (X \vee Y) \models\!\dashv \neg X \wedge \neg Y$
3. $\neg (X \wedge Y) \models\!\dashv \neg X \vee \neg Y$
4. $\neg\neg X \models\!\dashv X$
5. $X \wedge (Y \vee Z) \models\!\dashv (X \wedge Y) \vee (X \wedge Z)$
6. $X \vee (Y \wedge Z) \models\!\dashv (X \vee Y) \wedge (X \vee Z)$
7. $X \wedge X \models\!\dashv X$
8. $X \wedge Y \models\!\dashv Y \wedge X$
9. $X \wedge (Y \wedge Z) \models\!\dashv (X \wedge Y) \wedge Z$
10. $X \vee X \models\!\dashv X$
11. $X \vee Y \models\!\dashv Y \vee X$
12. $X \vee (Y \vee Z) \models\!\dashv (X \vee Y) \vee Z$
13. $(X \vee Y) \wedge X \models\!\dashv X$
14. $(X \wedge Y) \vee X \models\!\dashv X$
15. $(X \wedge \neg X) \vee Y \models\!\dashv Y$
16. $(X \wedge \neg X) \wedge Y \models\!\dashv X \wedge \neg X$
17. $(X \vee \neg X) \vee Y \models\!\dashv X \vee \neg X$
18. $(X \vee \neg X) \wedge Y \models\!\dashv Y$

さらに，論理定数 $\mathbf{T}$ や $\mathbf{F}$ を導入している場合には以下が成り立つ．

1. $X \wedge \neg X \models\!\dashv \mathbf{F}$
2. $X \vee \neg X \models\!\dashv \mathbf{T}$
3. $X \to X \models\!\dashv \mathbf{T}$
4. $X \wedge \mathbf{T} \models\!\dashv X$
5. $X \vee \mathbf{T} \models\!\dashv \mathbf{T}$
6. $X \wedge \mathbf{F} \models\!\dashv \mathbf{F}$
7. $X \vee \mathbf{F} \models\!\dashv X$
8. $\neg\mathbf{T} \models\!\dashv \mathbf{F}$

9. $\neg\mathbf{F} \models\!\!\models \mathbf{T}$
10. $\mathbf{T} \to X \models\!\!\models X$
11. $\mathbf{F} \to X \models\!\!\models \mathbf{T}$
12. $X \to \mathbf{T} \models\!\!\models \mathbf{T}$
13. $X \to \mathbf{F} \models\!\!\models \neg X$

演習問題

49. $X \models Y$ かつ $Y \models X$ と，$X \models\!\!\models Y$ が同値であることを示せ.
50. $X \models\!\!\models Y$ と $\models X \leftrightarrow Y$ が同値であることを示せ.
51. 上記の例についてそれらが正しいことを個々に確認せよ.
52. 左側に一つ論理式がある場合に限定すると，$\models$ は論理式の二項関係となる．$\models$ が前半 (まえはん) 順序関係であることを確認せよ．即ち前半順序関係の以下の 2 つの公理を満たすことを示せ.
    i. $X \models X$
    ii. $X \models Y$ かつ $Y \models Z \Rightarrow X \models Z$
53. 上の問題において，次のような，半順序関係の場合に追加する公理の反例を挙げよ．ここで $\equiv$ は字面上等しい，即ち全く同じ論理式であるという二項関係であった.
    iii. $X \models Y$ かつ $Y \models X \Rightarrow X \equiv Y$
54. $\models\!\!\models$ は論理式の二項関係である．$\models\!\!\models$ が同値関係であることを示せ．即ち同値関係の以下の 3 つの公理を満たすことを示せ．(一般に，前半順序関係があるとき，その関係が両向きに成り立っているという新しい関係を定義すると同値関係になることを示してもよい．)
    i. $X \models\!\!\models X$
    ii. $X \models\!\!\models Y \Rightarrow Y \models\!\!\models X$
    iii. $X \models\!\!\models Y$ かつ $Y \models\!\!\models Z \Rightarrow X \models\!\!\models Z$

2.15 論理式の同値変形

命題変数として [] という記号を付け加えて論理式の集合を改めて定義し，それらのうち [] がちょうど一箇所出現する論理式の集合を $\mathbf{Pro}_{[]}$ と表記する．また $\mathbf{Pro}_{[]}$ の元を $X'[]$ などと表記し，$X'[]$ の中の記号 [] を論理式 $Y \in \mathbf{Pro}$ で置き換えたものを $X'[Y]$ と表記することにする．$X'[Y] \in \mathbf{Pro}$ となる．すると，$X \in \mathbf{Pro}$ の部分論理式 Y があるとき，$X \equiv X'[Y]$ と表すことができる．また，X の部分論理式 Y を $Z \in \mathbf{Pro}$ で置き換えたものは $X'[Z]$ と表記される.

■例: $X'[] \stackrel{\text{def}}{\equiv} \underline{A} \to [] \in \mathbf{Pro}_{[]}, Y \stackrel{\text{def}}{\equiv} \underline{B} \in \mathbf{Pro}$ のとき，$X'[Y] \equiv \underline{A} \to \underline{B} \in \mathbf{Pro}$ である．これの Y を $\underline{C}$ で置き換えた論理式は $X'[C] \equiv \underline{A} \to \underline{C}$ である.

補題 5. 付値 ϕ に対し，$\phi(Y) = \phi(Z)$ であるとする．このとき $\phi(X'[Y]) = \phi(X'[Z])$ である.

証明: (概略) 論理式 $X'[]$ の構成に関する帰納法で証明できる. □

補題 6. $Y \models\!\!\models Z$ であれば $X'[Y] \models\!\!\models X'[Z]$ が成り立つ.

証明: $Y \models\!\models Z$ を仮定すると，上の補題からすべての ϕ について $\phi(X'[Y]) = \phi(X'[Z])$ だから． □

この補題を利用して，論理式の部分論理式をそれと論理的同値な論理式に置き換えてゆくことで，論理式をどんどん変形していくことが可能になる．なぜなら論理的同値性は同値関係だからであり，推移性が成り立つからである．

演習問題

55. 補題 5 を証明せよ．

2.16 論理式の標準形

ここでは古典命題論理の 2 つの標準形について説明する．

2.16.1 論理積標準形

論理積標準形 (conjunctive normal form, CNF) というのは，1 レベル目がすべて論理積 $\wedge$ で結ばれていて，2 レベル目がすべて論理和 $\vee$ で結ばれ，3 レベル目が命題変数か，あるいはそれに否定 $\neg$ が一つだけ付いた形になっているものである．

$\wedge, \vee$ は意味論的にはそれぞれ結合法則と交換法則を満たしているので，意味論的な取り扱いの際には $\wedge$ 同士，$\vee$ 同士の括弧を外しても問題ない[*12]．従ってこのような扱い方をする場合がある[*13]．

■例:

$(\underline{A} \vee \neg \underline{B}) \wedge (\neg \underline{C} \vee \neg \underline{B} \vee \underline{D}) \wedge \underline{B}, \neg \underline{A} \wedge (\underline{C} \vee \neg \underline{B} \vee \underline{E}) \wedge (\underline{B} \vee \underline{E})$

論理式として論理定数 T や F が許されている場合には，ここでは T,F を 1 レベル目の式として付け加えておくことにする．つまり T, F そのものも標準形ということである．論理定数を導入していない場合には，トートロジーである標準形の論理式は $\underline{A} \vee \neg \underline{A}$ などである．一応，一番上のレベルも 1 つ以上必要だからである．またここでは $\wedge$ と $\vee$ が列を引数に取るように書いている．これは先に出てきたように意味的には結合法則が成り立つからで，多項式の場合の $\cdot$ や $+$ と同様である．ただし正式にはどちらかに結合するとしておく必要がある．ここでは例えば $P \vee Q \vee R \vee S$ は $P \wedge (Q \wedge (R \wedge S))$ を表すとしておく．

2 レベル目に出てくる形の論理式を，節 (clause) と呼ぶ．3 レベル目に出てくる，命題変数かそれに一つだけ $\neg$ が付いた形の論理式を，リテラル (literal) と呼ぶ．2 つの呼び名は標準形以外でも使う場合がある．節をリテラルの集合であると捉える方法もあり，そうした場合には空節 (empty clause) も節の一種とする場合がある．空節は意味的には常に偽である論理式を表している．節形式は古典一階述語論理の自動証明である導出 (resolution) で用いられる形式でもある．ただしそこでは節は命題論理ではなく一階述語論理の論理式である．$\neg$ が付いていないリテラルを正リテラル，$\neg$ が付いたリテラルを負リテラルと言う．節のうち，さらに形を制限して正リテラルが高々一つであるとしたもののことをホーン節 (Horn clause) と言う．このように制限すると導出の手続きが非常に効率的に進むようになる．プログラミング言語 Prolog は，さらに導出を行う順序も 固定することで，論理式，即ちホーン節の列をプログラミング言語と見なしたものである．

CNF にはこのままでは冗長性がある場合がある．CNFX に 2 つの節 C と D があり，C に現れるリテラル

[*12] 厳密に言えば意味論的な同値で割る (同値類を作る) と結合法則と交換法則を満たす．
[*13] 後述のように完全性定理が成り立つので証明論的にも問題ない．

がすべて D に現れている場合，D を取り除いても論理的に同値となる (なぜか?)．ある節に同じリテラルが現れる場合，それらを一つに減らしても論理的に同値である．また，同じ命題変数の正負リテラルが両方現れる節 L はトートロジーであり，他のリテラルを取り除けるし，他にも節が存在する場合には L を取り除いても論理的に同値である．論理定数 T がある場合にはさらに L を T で置き換えられる．これらの操作を行えば，冗長性が取り除かれる．

演習問題

56. $(X_1 \vee \cdots \vee X_k \vee X_{k+1} \vee \cdots \vee X_n) \wedge (X_1 \vee \cdots \vee X_k) \models\!\models X_1 \vee \cdots \vee X_k$ を示せ．
57. リテラルの例を 3 つ挙げよ．
58. ホーン節の例を 3 つ挙げよ．
59. ホーン節でない節の例を 3 つ挙げよ．
60. CNF であるような論理式の例を 3 つ挙げよ．
61. CNF でない論理式の例を 3 つ挙げよ．
62*. 導出について調べよ．
63*. 論理型プログラミング言語 Prolog について調べよ．そこでは一つのプログラムが，函数型あるいは手続き型のプログラミング言語の場合には 2 通りの利用方法を持つ場合がある例を示せ (例えば 2 つのリストの結合と，1 つのリストを 2 つのリストに分解するなど)．

2.16.2 真理値表から論理積標準形を求める

任意に真理値表を与えられたときに，その真理値表を持つような CNF の論理式を与える手続きがある．これにより二値論理の場合，命題論理の論理式ですべての真理函数を表せることがわかる．また，論理式 X の真理値表を求めればそれから X の CNF を求めることができる．

入力の命題変数を $A_1, \ldots, A_n$ として，真理値表が与えられているとする．結果の真偽値が **F** となる各行 $l = 1, \ldots, m$ について，次のような節 C を一つずつ用意する．

$$C_l \stackrel{\text{def}}{=} (\neg)A_1 \vee \ldots \vee (\neg)A_n$$

ここで $\neg$ を付ける場合と付けない場合がある．その行の A_i の真偽値が **T** であれば $\neg$ を付け，**F** であれば付けない．するとその行が表している命題変数の真偽値の状態の場合のみに C_l は **F** となる．従って次の CNF の論理式 Y の真理値表は元の真理値表と一致する．

$$Y \stackrel{\text{def}}{=} C_1 \wedge \cdots \vee C_m$$

ただし $m = 0$ のとき，即ち X がトートロジーの時には $Y \stackrel{\text{def}}{\equiv} A_1 \vee \neg A_1$(あるいは論理定数 T がある場合には T) などとする．

このようにして求めた CNF には上の意味での冗長性はない．しかしさらに小さな CNF に置き換えられる場合がある．例えば $(P \vee A) \wedge (P \vee \neg A)$ は P に置き換えることができる．

演習問題

64. C_l が，l 行目の命題変数に対する真偽値割り当ての場合にのみ **F** となることを確認せよ．
65. 幾つかの論理式に対し，まず真理値表を求め，上の方法により CNF を求めよ．

2.16.3　論理積標準形への同値変形

論理式 X を次々に同値な論理式に置き換えてゆくことで X の CNF を求めることもできる．具体的には次のような操作を行う．ここで $A \Rightarrow B$ は，$C \equiv C'[D']$ の部分論理式 D' が左側の A の形をしている場合，それを右側の B の形に書き換えて $C'[B]$ とすることを表している．A と B が論理的同値であることに注意してほしい．また，各ステップでは左側の形をした全ての部分論理式に対し同様の操作を行う．

1. $X \to Y \Rightarrow \neg X \vee Y$
2. $\neg(X \vee Y) \Rightarrow \neg X \wedge \neg Y$,
   $\neg(X \wedge Y) \Rightarrow \neg X \vee \neg Y$
3. $\neg\neg X \Rightarrow X$
4. $(X \wedge Y) \vee Z \Rightarrow (X \vee Z) \wedge (Y \vee Z)$,
   $X \vee (Y \wedge Z) \Rightarrow (X \vee Y) \wedge (X \vee Z)$
5. $(X \vee Y) \vee Z \Rightarrow X \vee (Y \vee Z)$
6. $(X \wedge Y) \wedge Z \Rightarrow X \wedge (Y \wedge Z)$

以上の操作を若い番号からできる限り行うと，まず 1 により $\to$ はすべて消える．2 によりすべての $\neg$ は命題変数の左側に並ぶ．3 により $\neg$ は命題変数の直前に高々 1 つ付くのみとなる．4 により $\wedge$ が $\vee$ の外側に移動する．5 と 6 は $\vee$ と $\wedge$ の結合の仕方を標準的なものに変えるものである．

上記書き換えは各番号ごとに停止することがわかる．1 は一回書き換えるごとに $\to$ の個数が 1 減る．2 は $\neg$ が書き換えごとに $\wedge$ や $\vee$ の内側に移ってゆく (厳密にはもう少し正確に言う必要がある)．3 は $\neg$ の個数が 2 減る．4 は $\vee$ が $\wedge$ の内側に移る．5,6 は演算子の左側の深さが減る．

論理定数 T,F を導入している場合にはさらに以下の操作も必要である．

7. $\neg\mathrm{T} \Rightarrow \mathrm{F}, \neg\mathrm{F} \Rightarrow \mathrm{T}$
8. $\mathrm{T} \vee X \Rightarrow \mathrm{T}, X \vee \mathrm{T} \Rightarrow \mathrm{T}, \mathrm{F} \vee X \Rightarrow X, X \vee \mathrm{F} \Rightarrow X$
9. $\mathrm{T} \wedge X \Rightarrow X, X \wedge \mathrm{T} \Rightarrow X, \mathrm{F} \wedge X \Rightarrow \mathrm{F}, X \wedge \mathrm{F} \Rightarrow \mathrm{F}$

上記の書き換えの各ステップで，論理的な同値変形を行っているので，最終的に得られる論理式は X と論理的に同値である．この後，小節 2.16.1 に記述してあるようにして冗長性の削減を行う．すると，トートロジーであれば $A \vee \neg A$ などという形の論理式，あるいは T となる．そのため上記の手続きは，言い換えると論理式がトートロジーであることの決定手続きであるとも言える．

演習問題

66. 上の各変形の左右の論理式が論理的同値であることを確認せよ．
67. 幾つかの論理式の CNF を同値変形により求めてみよ．
68*. 論理式を記号列で表し，上記書き換えを DTM で行うことを考える．実行ステップ数の最大値が入力の大きさ n のどのような函数で抑えられるかを考察せよ．

2.16.4　論理和標準形

論理和標準形 (disjunctive normal form, DNF) は論理積標準形の双対である．即ち，1 レベル目がすべて

論理和 $\vee$ で結ばれていて，2 レベル目がすべて論理積 $\wedge$ で結ばれ，3 レベル目がリテラルになっているものである．

■例:

$(\underline{A} \wedge \neg \underline{B}) \vee (\neg \underline{C} \wedge \neg \underline{B} \wedge \underline{D}) \vee \underline{B}, (\neg \underline{BA}) \vee \neg \underline{D} \vee (\underline{A} \wedge \underline{B})$

■

CNF のときと双対的に，論理定数を導入していない場合には，常に **F** を表す DNF の論理式は $\underline{A} \wedge \neg \underline{A}$ などである．

論理和標準形の場合にも，対応する論理式 (を表す集合) を節と呼ぶ場合がある．

演習問題

69. $(X_1 \wedge \cdots \wedge X_k \wedge X_{k+1} \wedge \cdots \wedge X_n) \vee (X_1 \wedge \cdots \wedge X_k) \models\!\!\models X_1 \wedge \cdots \wedge X_k$ を示せ．
70. DNF であるような論理式の例を 3 つ挙げよ．
71. DNF でない論理式の例を 3 つ挙げよ．

2.16.5 真理値表から論理和標準形を求める

$\wedge$ と $\vee$ が双対の関係にあるので，CNF の場合と同様にして与えられた真理値表を持つ DNF の論理式を求めることができる．文章中や論理式の $\wedge$ と $\vee$，**T** と **F**，T と F を入れ替えることで同様の議論が成り立つ．

演習問題

72. DNF の場合の C_l(CNF の場合の双対) が，l 行目の命題変数に対する真偽値割り当ての場合にのみ **T** となることを確認せよ．
73. 上の方法で幾つかの論理式の DNF を求めてみよ．

2.16.6 論理和標準形への同値変形

こちらも同様である．

演習問題

74. 上の各変形の左右の論理式 (CNF の場合の双対) が論理的同値であることを確認せよ．
75. 幾つかの論理式の DNF を同値変形により求めてみよ．

2.17 形式的体系

以前の節で論理式の意味論的な正しさについて説明した．この節では構文論的な正しさを定義することになる．以下は古典命題論理に限定されない説明である．

証明図を与えて正しいことを示す対象をここでは証明対象[*14]と呼ぶことにする．例えば論理式やそれらな

[*14] 一般的な名称ではない．

どから成る記号列が証明対象だったりする．構文論的な正しさは，証明対象に対し，初めから正しいことがわかっている証明対象である公理 (axiom) と，正しいことがわかっている証明対象から別の正しい証明対象を導き出す推論規則 (inference rule) により与えられる．これらをまとめて形式的体系 (formal system)，あるいは単に体系と呼ぶ．「形式的」というのは，それらの意味まで考えずに単なる記号列として取扱うということを意味している．最終的には古典論理の場合，意味論的な正しさと構文論的な正しさが一致することを証明できる．これを指して完全性定理と呼ぶ．

多くの体系で公理は無限個あったりするが，それぞれ個々の公理として与えられるのではなく，通常は論理式等が入るメタ変数を含んだ有限個の公理スキーマとして与えられる．メタ変数を含んだ式にそこに入るべきもの，即ちこの場合は具体的な論理式を代入してできる式を一般にインスタンス (instance) と呼ぶ．このようにすると，公理の集合は (証明対象全体の集合の部分集合として) 決定可能な集合として与えられることになる．つまりある論理式が公理スキーマのインスタンスであるかどうかは計算機プログラムで有限時間内に (通常容易に[*15]) 判定できる．そうしておかないと，人間が判定するにしても証明対象が公理かどうかを誰もが同じ様に判定できるようにはならない．また推論規則はやはり有限個のスキーマとして与えられるのが普通である．正しいことがわかっている証明対象 (複数かもしれない) と，それらから導かれる証明対象が与えられたときに，その推論規則に当てはまるかどうかを (通常容易に) 決定可能な形で与えるのが普通である．

これら 2 つの制約，即ち公理であるかどうかが決定可能，推論規則が当てはまるかどうかが決定可能という条件が満たされる場合，正しいものとして推論される証明対象全体は一般には半決定可能な集合となる (条件を，決定可能より弱い条件である半決定可能としても同じ結論が成り立つ)．

ただし例えば古典命題論理の場合には決定可能なものになる (後述)．古典述語論理の場合には半決定可能なものにしかならない．Gödel の不完全性定理は，一階述語論理の体系に，自然数論を展開できるような公理や推論規則をさらに付け加えた場合，上記制約 (と若干の追加の制約) を満たしている限りどうやっても決定可能なものにはならないということを意味している．

形式的な体系には様々なものがあり，主なものを大まかに分類すると，Hilbert 流，自然演繹法，シーケント計算などに分類できる．Hilbert 流の体系では，証明対象は論理式である．公理スキーマの数が多めで，推論規則の数が少ない方式の体系である．ただしあまり自然に思えない手順で証明を与えることになる場合も多い．自然演繹法はやはり論理式が証明対象で，実際の人間の推論に近い，直観的に分かりやすい，推論規則の数が多い体系である．シーケント計算は，元々自然演繹法の分析のために考えられた体系である．証明対象は後述する様なシーケントであり，公理スキーマの数が少なく，論理演算子ごとに何個かの推論規則があるような体系である．技術的に取り扱いやすく，体系の性質を証明しやすい．歴史的にはこの順番で出現した．以前から使われていた体系に対する不満を解消したり，構文論的な無矛盾性などの証明を行ったりするために考え出されたのである．なおシーケント計算と現在よく使われている形式の自然演繹法は Gentzen により考案されたものである．

2.18 形式的体系の要素

以下では，「論理式」(formula) という言葉は古典 (classical, 二値 (two-valued)) 命題論理の論理式を指す (が，定義や補題，定理は，他の論理の場合にもそのままあるいは必要に応じて修正すれば適用・成立するものもある)．

[*15] ここで「容易に」というのが曖昧であれば，クラス P に属する問題であると読み替えてもよい．

■証明対象　本稿では証明する対象のことを証明対象ということにする (一般的な用語ではない)．論理式だったり，論理式と若干の記号でできた記号列または木だったりする．

■公理　初めから正しいとわかっている証明対象のことを公理 (axiom) という．論理式などを表すメタ変数を含んだ公理スキーマ (axiom schema) で与えられる場合も多い．公理は，次の推論規則の $n = 0$ となっている特殊ケースであると見ることもできる．

■推論規則　すでに正しいことがわかっているいくつかの証明対象 $o_1, \ldots, o_n$ から別の正しい証明対象 o を導く方法を記述するのが推論規則 (inference rule) である．通常はやはりスキーマの形で与えられる (ので推論規則スキーマとは言わない)．

$$\frac{o_1 \cdots o_n}{o}$$

■証明図　証明図 (proof figure) とは，公理から始まってその下に推論規則を連ねた木のことをいう．木全体を，木の一番下，すなわち根の位置にくる証明対象 o の証明図，あるいは証明木，単に証明という．o の証明図が存在するとき，o は証明可能 (provable) であるという．またそのような o を定理 (theorem)[*16] ということにする．

証明図や定理にメタ変数を含める場合があり，その場合にはそれらのメタ変数に具体的な論理式等を代入してできる証明対象についてすべて証明図を与えたことになる．そういった場合，本稿では証明図スキーマ，定理スキーマと呼ぶことにするのがよいように思われるが，省略して単に証明図，定理と呼ぶこともある．

証明図，あるいはそのスキーマがこのようなものなので，それらに関する性質を証明したい場合，それらの構成に関する帰納法を用いることができる．これは論理式の構成に関する帰納法に類似したものである．

証明図の一部を省略することもある．そういった場合，本稿では規則の横線を二重にして省略を表示する場合がある．

■派生規則　公理スキーマや推論規則を組み合わせて，別の推論規則を作ることができる．そういった規則を派生規則 (derived rule) と言う．言い換えると (複数の) 元の公理や推論規則に置き換えられるような規則が派生規則である．つまり派生規則も用いて作られた証明図を，元の公理や推論規則のみから成る証明図に変換できる．

以下で 3 種類 (aLK も入れると 4 種類) の形式的体系を紹介する．これらは見かけ上異なっているが，証明できる論理式の集合は同等となることを示すことができる[*17]

[*16] 文脈によってはメタレベルの定理と紛らわしい．証明されたのが対象レベルのスキーマであれば，「定理スキーマ」という言い方なら殆どの場合紛らわしくない．

[*17] 本稿では省略する．例えば其々の体系が健全かつ完全であることを示せば意味論を介して同等性が言える．

2.19 Hilbert 流

■公理 以下はポーランドの論理学者 Łukasiewicz[*18]による版である．これは，$\wedge$ と $\vee$ については $X \wedge Y \stackrel{\text{def}}{=} \neg(X \to \neg Y), X \vee Y \stackrel{\text{def}}{=} X \to Y$ として定義し，$\to$ と $\neg$ のみを論理演算子とする体系である．

$$X \to (Y \to X)\,, \tag{11}$$
$$(X \to (Y \to Z)) \to ((X \to Y) \to (X \to Z))\,, \tag{12}$$
$$(\neg X \to \neg Y) \to (Y \to X)\,. \tag{13}$$

$\to$ と $\neg$ 以外に $\wedge$ と $\vee$ も基本的な論理演算子とする Hilbert 流の形式的体系もあり，この場合には公理の数がさらに多くなる．

■推論規則 推論規則は一つだけである．

$$\frac{X \qquad X \to Y}{Y}$$

■証明図 証明図は，公理から始まって推論規則を積み重ねた木であり，すべての葉が公理でなければならない．例えば以下は定理スキーマ $X \to X$ の証明図である．

$$\dfrac{X \to (Y \to X) \qquad \dfrac{X \to ((Y \to X) \to X) \qquad X \to ((Y \to X) \to X) \to (X \to (Y \to X)) \to (X \to X)}{(X \to (Y \to X)) \to (x \to X)}}{X \to X}$$

なお本稿では Hilbert 流の体系の場合であっても証明図という概念を使用しているが，公理と論理式，あるいはそれらのスキーマの列を代わりに用いて議論を行うのが普通である (議論の道筋も異なったものとなる場合がある)．この場合，各論理式は公理のインスタンスか，列中その論理式より左側に現れる論理式から推論規則により正しいと導かれる論理式である．本稿では他の体系と統一的に記述するために上記のような説明を行っている．

上では基本的な論理演算子を $\neg$ と $\to$ のみであるとしていた．さらに $\neg$ を論理演算子から除くと論理演算子は $\to$ のみとなる．このような論理を含意命題論理 (implicational propositional logic) と呼ぶ．含意命題論理の Hilbert 流の公理系の一つは上の最後の公理スキーマ (13) を次のような公理スキーマ (14) に置き換えたものである．これを Peirce's law と呼ぶ．

$$((X \to Y) \to X) \to X. \tag{14}$$

他の論理演算子を表したい場合には，矛盾を表す $\bot$ を論理定数として追加し，$\neg X \stackrel{\text{def}}{=} X \to \bot, X \wedge Y \stackrel{\text{def}}{=} (X \to (Y \to \bot)) \to \bot, X \vee Y \stackrel{\text{def}}{=} (X \to Y) \to Y$ などとして他の論理演算子を定義することになる．ただしこれだけでは $\bot$ の性質を表すには不十分である．例えば次のような公理スキーマをさらに付け加える必要がある．

$$\bot \to X. \tag{15}$$

[*18] 日本語表記はウカシェヴィッツとする場合が多い．

■型理論との関係　この資料の本筋から見れば余談ながら，興味がある方もいらっしゃると思われるので簡単に書くことにする．但しこの部分の理解には函数型言語あるいは入計算の若干の事前知識が必要である．

上記体系の最初とその次の公理は，型付き入計算の K コンビネータと S コンビネータの型となっている．これは偶然ではなく，型理論と logic は実はほぼ同じものであることに由来する．

Hilbert 流の体系で，様々なトートロジーを証明できることと，K と S をうまく組み合わせると，与えられた λ 式と β 同値な λ 式となることが対応している．Peirce's law については，これは継続の型であり，継続は，古典論理を函数型言語に導入した場合の言語要素である．

演習問題

76. $X \vee Y \to Y \vee X$ の $\vee$ を上記の定義に従って展開し，さらに展開後の論理式の証明図を示 せ．論理演算子は $\neg$ と $\to$ のみ，論理定数なし，公理スキーマは (11),(12),(13) とする．
解答例: 展開すると $(\neg X \to Y) \to (\neg Y \to X)$．証明図は省略.
77. 二重否定除去 $\neg\neg X \to X$ の証明図を示せ．公理スキーマは (11),(12),(13) とする．
78. 含意命題論理で考える．排中律 $\neg X \vee X$ と $X \vee \neg X$ を展開し，それらの証明図をそれぞれ示せ．論理演算子は $\to$，論理定数は $\bot$，公理スキーマは (13) を展開した次のような (15')[*19]と，(11),(12),(15) とする．
$$((X \to \bot) \to (Y \to \bot)) \to (Y \to X)\,. \tag{15'}$$
79. 論理演算子は $\to$，論理定数は $\bot$ とする．(14),(15) が，(11),(12),(15') から導けることを示せ[*20].
80. 逆に (11),(12),(14),(15) から (15') を導けることを示せ.
81. 上の問題で (15) を除いた場合，(14) から (15') を導けないことを示せ．(ヒント: まず (11),(12),(14) と (2.19) から成る体系が健全，即ちこれらにより証明できる論理式が全てトートロジーであることをいう．次に，命題変数 A, B に対し $((A \to L) \to (B \to A)) \to (B \to A)$ を導けるとしたら，命題変数 C に対して $((A \to C) \to (B \to C)) \to (B \to A)$ が導けることを示し，それがトートロジーとならない場合があることを言う．)

2.20　自然演繹

以下は Gentzen による体系．論理定数 $\bot$ を導入する (F と同じだと思ってよい).

■公理　$X \vee \neg X_{\text{(excluded-middle)}}$
日本語では排中律と言う．名称は，論理式の値に中間の値がないという意味である[*21].

■推論規則

$$\frac{X \qquad Y}{X \wedge Y}\ (\wedge\text{-introduction}),$$

[*19] この問題の場合，実は (15') は必要ない.
[*20]「導けることを示せ」と「証明図を示せ」が，求めていることが異なることに注意せよ．前者では具体的な証明図を示す必要が必ずしもなく，証明図の存在を示せば十分である.
[*21] 但し，MV 代数あるいは Łukasiewicz 論理の意味論のように，中間の値があってもこの式の真理値は **T** となるような logic もある.

$$\frac{X \wedge Y}{X}, \quad \frac{X \wedge Y}{Y} \ (\wedge\text{-elimination1,2}),$$

$$\frac{X}{X \vee Y}, \quad \frac{Y}{X \vee Y} \ (\vee\text{-introduction1,2}),$$

$$\frac{\begin{array}{ccc} & [X] & [Y] \\ & \vdots & \vdots \\ X \vee Y & Z & Z \end{array}}{Z} \ (\vee\text{-elimination}),$$

$$\frac{\begin{array}{c} [X] \\ \vdots \\ Y \end{array}}{X \to Y} \ (\to\text{-introduction}),$$

$$\frac{X \qquad X \to Y}{Y} \ (\to\text{-elimination}),$$

$$\frac{\begin{array}{c} [X] \\ \vdots \\ \bot \end{array}}{\neg X} \ (\neg\text{-introduction}), \quad \frac{X \qquad \neg X}{\bot} \ (\neg\text{-elimination}),$$

$$\frac{\bot}{X} \ (\bot).$$

なお [] は対応する規則を適用する時に，作成途中の証明図の葉の部分に位置する論理式の出現の両側に書き入れるものであり，記号列として同じ論理式の出現 0 個以上について付ける．同じ論理式の出現のうちの何個，或いはどれに付けるかは自由であり，付けずに残すものがあってもよい．どの規則の適用時に [] を付けたのかをはっきりさせるためには，規則の出現 (横線の横など) と [] に番号を付けるなどして他と区別する必要がある．

■証明図　証明図はやはり公理から始まって推論規則を積み重ねてできる木である．葉の部分はすべて公理であるか，[] で囲まれていなくてはならない．

演習問題

82. Łukasiewicz の 3 つの公理スキーマを A_1, A_2, A_3 とする．それぞれについて自然演繹による証明図を示せ．
83. 本稿中に現れた他のトートロジーについて証明図を示せ．
84. 本稿中に現れた命題論理式で，トートロジーでないものについて，証明図を構成できないことを確認せよ．

2.21　シーケント計算 LK

Gentzen により構築された，シーケント計算の体系の一つである LK(Logischer Kalkül，論理計算) を紹介する．ここでは命題論理用に修正した版を導入する．

以下で出てくる論理式の列 $P_1, \ldots, P_n$ は，集合とは異なり，順序が付いていて同じものが複数回現れる場合もあることに注意してほしい．**0 個** 以上の有限個の論理式の列を Γ，Δ，Π，Θ などのメタ変数を使って表すことにする．Γ, Δ や Γ, P などは，二つの列 Γ と Δ を並べてできる列，列 Γ の最後に論理式 P をつけてできる列を表す．0 個の論理式から成る列のことを空列と呼ぶ．ここで「,」の左右の論理式の列が空列の時には，「,」は表記上除く場合があるので注意してほしい．例えば Γ が空列の時には Γ, Π は Π と等しい．また列 Γ に出てくる要素全体からなる集合を Γ_s と書くことにする．

$\Gamma \vdash \Delta$ の形をした記号列のことをシーケント (sequent) と呼ぶ (Gentzen は $\to$ という記号を用いたが，本稿では多少短くするために $\vdash$ を使っている．$\vdash$ は本来，他の形式的体系で用いられ，証明可能であることを表している．その文脈ではドイツ語の場合 Beweis と読むとのことである．ただし日本語・英語では turnstyle などと読んだりするようであるが，これは記号そのものの呼び方であろう．)．$\Gamma \vdash \Delta$ の直観的な意味は，Γ の論理式がすべて成り立つと仮定したときに，Δ の論理式を $\vee$ で結んだものが成り立つ，ということである．LK ではシーケントを証明対象とする．

2.21.1 LK の公理

LK で公理に相当するのは $P \vdash P$ という形のシーケントすべてである．つまり LK の公理は公理スキーマで与えられる．これを含めて以下での P や Q は，命題変数ではなく論理式を表すメタレベルの変数であることに注意．公理に相当するシーケントのことを始式 (initial sequent) という．

2.21.2 LK の推論規則

推論規則は構造に関する規則と論理演算子に関する規則に分けられる．規則の名称は異なるものを用いる場合もある．

■構造に関する規則

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

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

$$\frac{\Gamma, P, Q, \Delta \vdash \Lambda}{\Gamma, Q, P, \Delta \vdash \Lambda}\ \text{(exchange-left)}, \qquad \frac{\Gamma \vdash \Delta, P, Q, \Lambda}{\Gamma \vdash \Delta, Q, P, \Lambda}\ \text{(exchange-right)},$$

$$\frac{\Gamma \vdash \Delta, P \qquad P, \Lambda \vdash \Theta}{\Gamma, \Lambda \vdash \Delta, \Theta}\ \text{(cut)}$$

以上は論理演算子についての規則ではなく，シーケントの構造に関する規則なので構造規則 (structural rule) と呼ぶ．今の場合，構造に関する規則は左右対称なものになっている．

■論理演算子に関する規則

$$\frac{P, \Gamma \vdash \Delta}{P \wedge Q, \Gamma \vdash \Delta}\ (\wedge\text{-left1}), \qquad \frac{Q, \Gamma \vdash \Delta}{P \wedge Q, \Gamma \vdash \Delta}\ (\wedge\text{-left2})$$

$$\frac{\Gamma \vdash \Delta, P \qquad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q}\ (\wedge\text{-right})$$

$$\frac{\Gamma \vdash \Delta, P}{\Gamma \vdash \Delta, P \vee Q}\ (\vee\text{-right1}), \qquad \frac{\Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \vee Q}\ (\vee\text{-right2})$$

$$\frac{P, \Gamma \vdash \Delta \qquad Q, \Gamma \vdash \Delta}{P \vee Q, \Gamma \vdash \Delta}\ (\vee\text{-left})$$

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

$$\frac{\Gamma \vdash \Delta, P \qquad Q, \Lambda \vdash \Theta}{P \to Q, \Gamma, \Lambda \vdash \Delta, \Theta}\ (\to\text{-left}), \qquad \frac{P, \Gamma \vdash \Delta}{\Gamma \vdash \Delta, P \to Q}\ (\to\text{-right}),$$

2.21.3 LK の証明図

LK における証明図は，始式から始まってその下に推論規則を連ねた木である．証明図の一番下，すなわち根の位置にくるシーケントのことを終式 (end sequent) という．木全体が終式の証明図となる．シーケント s に対し，s の証明図が存在するとき，s は証明可能であるという．またそのような s を定理ということにする．

証明図の一部を省略することもある．特に exchange，あるいは weakening や contraction も省略する場合もある．そういった場合，本稿では規則の横線を二重にして省略を表示する場合がある．

■証明図の例:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\underline{B} \vdash \underline{B}}{\underline{B} \vdash \underline{A} \vee \underline{B}}\ (\vee\text{-right2}) \qquad \dfrac{\underline{A} \vdash \underline{A}}{\underline{A} \vdash \underline{A} \vee \underline{B}}\ (\vee\text{-right1})}{\underline{B} \vee \underline{A} \vdash \underline{A} \vee \underline{B}}\ (\vee\text{-left})}{\underline{B} \vee \underline{A} \vdash (\underline{A} \vee \underline{B}) \vee \underline{C}}\ (\vee\text{-right1})}{\vdash \underline{B} \vee \underline{A} \to (\underline{A} \vee \underline{B}) \vee \underline{C}}\ (\to\text{-right})$$

上は具体的な証明図であり，スキーマではない．

$$\frac{\dfrac{P \vdash P}{}}{\Gamma, P, \Delta \vdash \Pi, P, \Sigma}\ (\text{weakening-left, weakening-right, exchange})$$

上は P に論理式，Γ 等に論理式の列を入れたら具体的な証明図となるスキーマである．また $\Gamma, P, \Delta \vdash \Pi, P, \Sigma$ は定理スキーマである (本稿では $\Gamma, \Delta, \Pi, \Sigma$ に現れる各論理式と P を命題変数に限ったこの形式のスキーマを TC(tautology clause) と名付けることにする)．シーケントが TC であるかどうか (シーケント全体の集合の中で，TC であるようなシーケント全体の集合) は (簡単に) 決定可能である．

定理スキーマを公理スキーマの代わりに使った木があったとする．この時，定理スキーマをそれの証明図で置き換えれば木の根の部分の証明図となる．

2.22 LK の派生規則

LK の推論規則をいくつか繋げて例えば次の様に証明図の断片を作って派生規則を作ることができる．これらの葉は公理とは限らないことに注意してほしい．

$$\dfrac{\dfrac{\dfrac{\dfrac{P,Q,\Gamma\vdash\Delta}{P\wedge Q,Q,\Gamma\vdash\Delta}}{Q,P\wedge Q,\Gamma\vdash\Delta}}{P\wedge Q,P\wedge Q,\Gamma\vdash\Delta}}{P\wedge Q,\Gamma\vdash\Delta} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\Gamma\vdash\Delta,P,Q}{\Gamma\vdash\Delta,P,P\wedge Q}}{\Gamma\vdash\Delta,P\vee Q,P}}{\Gamma\vdash\Delta,P\vee Q,P\vee Q}}{\Gamma\vdash\Delta,P\wedge Q}$$

今の場合，以下のような新しい規則を導入して証明図を作っても，それらの規則を上のような証明図の断片と置き換えれば，元々の規則だけから成る証明図を得られる．

$$\dfrac{P,Q,\Gamma\vdash\Delta}{P\wedge Q,\Gamma\vdash\Delta}\ (\wedge\text{-left}) \qquad \dfrac{\Gamma\vdash\Delta,P,Q}{\Gamma\vdash\Delta,P\vee Q}\ (\vee\text{-right})$$

また同様に，以下の様にして新しい規則を導入する．

$$\dfrac{\dfrac{\Gamma\vdash\Delta,P \qquad Q,\Gamma\vdash\Delta}{P\to Q,\Gamma,\Gamma\vdash\Delta,\Delta}\ (\to\text{-left})}{\dfrac{}{P\to Q,\Gamma\vdash\Delta}}\ (\text{exchange,contraction-left,contraction-right})$$

これによって以下の派生規則ができる．

$$\dfrac{\Gamma\vdash\Delta,P \qquad Q,\Gamma\vdash\Delta}{P\to Q,\Gamma\vdash\Delta}\ (\to\text{-left'})$$

$$\dfrac{P,\Gamma,\Delta\vdash\Lambda}{\Gamma,P,\Delta\vdash\Lambda}\ (\text{exchange-left}), \qquad \dfrac{\Gamma\vdash\Delta\Lambda,P}{\Gamma\vdash\Delta,P,\Lambda}\ (\text{exchange-right})$$

これによって以下の派生規則ができる．

$$\dfrac{P,\Gamma,\Delta\vdash\Lambda}{\Gamma,P,\Delta\vdash\Lambda}\ (\text{exchange-left'}), \qquad \dfrac{\Gamma\vdash\Delta,\Lambda,P}{\Gamma\vdash\Delta,P,\Lambda}\ (\text{exchange-right'})$$

演習問題

85. Łukasiewicz の 3 つの公理スキーマを A_1, A_2, A_3 とする．$\vdash A_1$,$\vdash A_2$,$\vdash A_3$ がそれぞれ LK で証明可能であることを示せ．
86. 排中律のスキーマ $X\vee\neg X$ が LK で証明可能であることを示せ．
87. 二重否定除去のスキーマ $\neg\neg X\to X$ が LK で証明可能であることを示せ．
88. 上で導入されたそれぞれの派生規則を示す証明図の断片で省略されている規則の数はいくつか．

2.22.1 LK の特徴

T を命題論理のシーケントの LK における証明図で，規則 cut が現れないものとする．すると，各規則の上の段に現れる論理式はすべて下の段に現れる論理式のうちどれかの部分論理式である．この事実を指して部分論理式性 (subformula property) が成り立つと言う．しかも論理演算子に関する規則の場合には，真に小さい論理式となっている．これらは，各規則の形から直ちに確かめられることである．部分論理式性は重要な性質であり，命題論理の論理式が LK で証明可能であるかどうかを決定する決定手続きを，LK の推論規則たちを元にして作れることを意味している (後述)．このことと (別の) 後述の LK の完全性と併せると，命題論理の

論理式が恒真であるかどうかが決定可能であることの別証明を与える[*22].

また後述のカット除去定理は，もしそれを構文論的に証明すれば，LK の無矛盾性の構文論的な証明を与える．カット除去定理とは，規則 cut を含む証明図が存在する場合，それを元にして cut を含まない証明図を得られる手続きがあるという定理である．Gentzen により一階述語論理の LK について証明された．LK の無矛盾性，即ち $\vdash$ の証明図が存在しないことの証明は，これが下段のシーケントとなりえる規則が cut 以外に存在しないことから導かれる．

これらの事実はシーケント計算の美点となっている．本稿でも cut を含む証明図が存在する場合，cut を含まない証明図が存在することを証明する．ただしその証明では恒真性という意味論的な概念を利用するし，cut を含む証明図を手続き的に変換するという，純粋に構文論的な証明にはなっていない．

2.23 形式的体系 aLK

ここで，TC を公理スキーマとし，exchange-left', exchange-right', $\wedge$-left', $\wedge$-right, $\vee$-right', $\vee$-left, $\neg$-left, $\neg$-right, $\neg$-left', $\neg$-right を推論規則とするシーケントに対する命題論理の体系を考え，それを aLK と呼ぶことにする．LK と aLK は，実は定理，即ち証明図を持つシーケントの集合が同じであることが後でわかる．LK があるのにわざわざ aLK を考えるのは aLK の方が解析しやすい性質を持っているからである．

以下に形式的体系 aLK について纏める．

■aLK の公理 $\Gamma, A, \Delta \vdash \Pi, A, \Sigma$ (TC)

これは LK の定理スキーマになっている．前述のようにシーケント中のすべての論理式を命題変数のみに制限している．

■aLK の構造に関する規則

$$\frac{P, \Gamma, \Delta \vdash \Lambda}{\Gamma, P, \Delta \vdash \Lambda}\ (\text{exchange-left'}), \qquad \frac{\Gamma \vdash \Delta, \Lambda, P}{\Gamma \vdash \Delta, P, \Lambda}\ (\text{exchange-right'}),$$

■aLK の論理演算子に関する規則

$$\frac{P, Q, \Gamma \vdash \Delta}{P \wedge Q, \Gamma \vdash \Delta}\ (\wedge\text{-left'}), \qquad \frac{\Gamma \vdash \Delta, P \qquad \Gamma \vdash \Delta, Q}{\Gamma \vdash \Delta, P \wedge Q}\ (\wedge\text{-right}),$$

$$\frac{P, \Gamma \vdash \Delta \qquad Q, \Gamma \vdash \Delta}{P \vee Q, \Gamma \vdash \Delta}\ (\vee\text{-left}), \qquad \frac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \vee Q}\ (\vee\text{-right'}),$$

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

$$\frac{\Gamma \vdash \Delta, P \qquad Q, \Gamma \vdash \Delta}{P, \Gamma \vdash \Delta, Q}\ (\rightarrow\text{-left'}), \qquad \frac{\Gamma \vdash \Delta, P, Q}{\Gamma \vdash \Delta, P \rightarrow Q}\ (\rightarrow\text{-right})$$

' が付いた規則は LK の派生規則になっており，付いていない規則は LK の規則と同じである．つまり aLK で証明図を持つシーケントは LK でも証明図を持つ．

[*22] 一つ目の証明は，論理式の真偽値表を書いて結果の真偽値が全て **T** であれば yes，そうでなければ no を答えるというものであった．

2.24 形式的体系の性質

ここで証明対象をシーケントとする形式的体系の性質を幾つか定義し，LK と aLK についてそれらが成り立つことを示す．証明対象を論理式とする形式的体系の場合にもほぼ同様の定義を行い，示すことができる．

2.24.1 無矛盾性

シーケントについての形式的体系 S で，$\Gamma \vdash P$ と $\Gamma \vdash \neg P$ が両方証明図を持つような論理式 P が存在するとき，Γ は S で矛盾している (inconsistent) という．そうでないとき，Γ は S で無矛盾 (consistent) であるという．特に Γ として空列が矛盾している時に S は矛盾しているという．

LK の場合，以下のように $\Gamma \vdash$ が証明図を持つことと，$\Gamma \vdash P$ と $\Gamma \vdash \neg P$ の両方が証明図を持つことは同値であり，特に LK が無矛盾であるというのはシーケント $\vdash$ が証明図を持たないことと同値である．$\vdash$ が矛盾しているとどのような論理式 Q についても $\Gamma \vdash Q$ が証明図を持つこともわかる．

$$\frac{\Gamma \vdash}{\Gamma \vdash Q}\ (\text{weakening-right},Q \equiv P, \neg P \text{ の場合も OK})$$

$$\dfrac{\dfrac{\Gamma \vdash P}{\Gamma \vdash P \wedge \neg P}\ (\wedge\text{-right}) \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{P \vdash P}{\neg P, P \vdash}\ (\neg\text{-left})}{P, \neg P \vdash}\ (\text{exchange-left})}{P \wedge \neg P, \neg P \vdash}\ (\wedge\text{-left1})}{\neg P, P \wedge \neg P \vdash}\ (\text{exchange-right})}{P \wedge \neg P, P \wedge \neg P \vdash}\ (\wedge\text{-left2})}{P \wedge \neg P \vdash}\ (\text{contraction-left})}{\Gamma \vdash}\ (\text{cut})$$

2.24.2 LK の健全性

次に体系 S の健全性を定義する．

以下で $\Gamma_\wedge$ は Γ のすべての論理式を $\wedge$ で結んでできる論理式である．また，$\Gamma|\vee$ は $\vee$ で結んでできる論理式を表す．ただし，Γ が空列の時には $\Gamma_\wedge \equiv \underline{A} \to \underline{A}, \Gamma_\vee \equiv \underline{A} \wedge \neg \underline{A}$ としておく．

■例: $\Gamma \equiv P, Q, R$ のとき，$\Gamma_\wedge \equiv P \wedge Q \wedge R$ となる．

$\Gamma \equiv P$ のときには $\Gamma_\wedge \equiv P$ である．

定義 シーケントの充足

付値 ν とシーケント $s \equiv \Gamma \vdash \Delta$ があるとする．$\nu(s) \stackrel{\text{def}}{=} \nu(\Gamma_\wedge \to \Delta_\vee)$ として ν による s の真理値を定める．付値 ν が論理式 $\Gamma_\wedge \to \Delta_\vee$ を充足するとき，ν はシーケント s を充足するという．どのような付値によっても充足されるような古典命題論理のシーケントを恒真であるという．

古典命題論理の恒真なシーケントをやはりトートロジーと呼ぶ．

■例 $\nu(P) = \mathbf{T}, \nu(Q) = \mathbf{F}$ とすると，$\nu \models P \wedge Q \to \neg P \vee Q$ なので，ν はシーケント $P, Q \vdash \neg P, Q$ を充足する．

補題 7. 命題論理の論理式 X が恒真であることと，シーケント $\vdash X$ が恒真であるのは同値である．

証明: 定義より直ちに成り立つ． □

定義 健全性

形式的体系 S の定理, 即ち証明図を持つシーケントがすべて恒真である時, S は (弱い意味で) 健全 ((weakly) sound) であると言う.

健全でない体系 S を考えてもあまり意味はない. 正しくない場合があるシーケントに証明図があるからである.

補題 8. シーケントに関する体系 S が弱い意味で健全であれば無矛盾である.

証明: X と $\neg X$ が両方ともトートロジーとなることはない. □

ただしこれは意味論的な概念を利用した無矛盾性の証明である.

定義 完全性

恒真なシーケントがすべて形式的体系 S の定理である, 即ち証明図を持つ時, S は (弱い意味で) 完全 ((weakly) complete) であると言う.

健全かつ完全であることを指して完全と言う場合もある. 最終的には後述するように LK も aLK も弱い意味で完全でもある.

定理 9 (LK の健全性). LK は弱い意味で健全である. すなわち, LK でシーケント s が証明図を持つとすれば, s はトートロジーである.

証明: (概略) 証明図の構成による帰納法によって証明できる. まず各公理はトートロジーである. 加えて個々の推論規則について, 上のシーケントがすべてトートロジーであれば, 下のシーケントがトートロジーであることを示す. □

系 10. LK は無矛盾である.

証明: $\vdash$ はトートロジーではないからである. □

一般に, 体系 S' の公理 (スキーマ) がすべて別の健全な体系 S の定理 (スキーマ) であり, S' の推論規則が S の規則か派生規則であるとする. このとき, S が健全であれば S' も健全となる. 逆に $S6\prime$ が完全であれば S も完全である.

補題 11 (aLK の健全性). aLK は弱い意味で健全である.

証明: TC は LK の定理スキーマであり, LK は健全なのでこれらはトートロジーのスキーマである. aLK の各推論規則は LK の派生規則なので, 上のシーケントがすべてトートロジーであれば下のシーケントもそうである. □

しかし aLK についてはもう少し強い性質が成り立つ.

補題 12. 付値 ϕ のと aLK の各推論規則について, 上の段のシーケントを $s_1, \ldots, s_n$, 下の段のシーケントを s とすると, $\phi(s_i)$ がすべて $\mathbf{T}$ であることと, $\phi(s)$ が $\mathbf{T}$ であることは同値である.

証明: aLK の各推論規則ごとに同値性を示す. 例えば $\neg$-left の場合, $\phi(\neg P \wedge \Gamma_\wedge \to \Delta_\vee) = \mathbf{T}$ となるのは $\phi(\neg P) = \mathbf{F}$, $\phi(\Gamma_\wedge) = \mathbf{F}$, $\phi(A) = \mathbf{T}$ の 3 つの場合である. いずれの場合にも $\phi(\Gamma_\wedge \to \Delta_\wedge \vee P) = \mathbf{T}$ であ

る．$\phi(\Gamma_\wedge \to \Delta_\vee \vee P) = \mathbf{T}$ となるのは $\phi(P) = \mathbf{T}$，$\phi(\Gamma_\wedge) = \mathbf{F}$，$\phi(\Delta_\vee) = \mathbf{T}$ の 3 つの場合である．
$\phi(\neg P) = \mathbf{F}$ と $\phi(P) = \mathbf{T}$ は同値なので求める同値性が成り立つことがわかる．
他の規則の場合も類似の議論により示される． □

演習問題

89. 定理 9 の証明の細部を完成せよ．
90. 補題 12 の証明で，本文中で記述されていない他の規則の場合について補完せよ．

2.25 証明図の構成手続き

ここで，シーケント s に対して証明図の候補を作ってゆく手続きを与える．

■手続き W(Wang のアルゴリズムの変形)[*23] 最初 s を木 T の唯一の葉=根として出発する．

1. T のいずれの葉のシーケントにも論理演算子が現れていなければ，手続き W を停止し，T を結果とする．
2. そうでない場合，一つ以上の論理演算子が現れている葉のシーケントを選び，s' とする (例えば一番左側のものを選んでもよい)．$s' \equiv \Gamma \vdash \Delta$ として，演算子が Γ または Δ のどれかの論理式に現れている．それらから一つ選ぶ (やはり例えば一番左側のものとしてもよい)．
3. Γ から選んだ場合，その論理式が一番左になければ exchange-left' で一番左に移動する．論理演算子に応じて ¬-left, ∧-left', ∨-left, →-left' のうちいずれか一つを適用し，それらのインスタンスを選んだシーケントの上に繋いで T を延長する．
4. Δ の場合には同様に exchange-right' で一番右に移動する．その後は Δ の場合と同様にして T を延長する．
5. 1 に戻る．

■停止性 ここで，s 中の論理演算子の数は有限であることと，3 あるいは 4 で，推論規則の上の段の各シーケント中の論理演算子の数は下の段のシーケント中の数よりも 1 以上減ることに注意してほしい．

補題 13. 手続き W は停止する．

証明: 対象とするシーケント s 中の論理演算子の個数を d 個とする．上の注意の内容から，T の深さは一定値 $2d$ 以下にしかならない．T の分岐の仕方が 2 以下であることと併せると，T の大きさ (ノードの数) は高々 2^{2d+1} であり有限である．手続き W の各ループで T に 1 個あるいは 2 個の規則のインスタンスが追加されるので，ループの回数も有限である． □

停止時の T の exchange-(left',right') を除いた深さは一定で，s 中の論理演算子の数と等しい．今示したように手続き W は停止するが，できた木が aLK の証明図になっているとは限らない．葉の部分のシーケントがすべて aLK の公理だとは限らないからである．

[*23] 日本語表記は「ワン」が近いと思われる．

補題 14. ϕ を付値とする．上の手続き W の途中で出てくる各 T において，葉の部分に現れるシーケントを $s_1, \ldots, s_n$ とする．$\phi(s) = \mathbf{T}$ であるための必要十分条件は，すべての $i = 1, \ldots, n$ に対し $\phi(s_i) = \mathbf{T}$ である．

証明: $(\Rightarrow)$ $\phi(s) = \mathbf{T}$ であるとする．ループを回った回数 $l = 0$ のときには $\phi(s) = \mathbf{T}$ は自明である．

手続き W のあるステップで $i = 1, \ldots, n$ に対し $\phi(s_i) = \mathbf{T}$ であるとする．次にどこかの葉を延長することになる．補題 12 より，延長した先のどの葉のシーケント s' でも $\phi(s') = \mathbf{T}$ となるので，その時点でのすべての葉のシーケント s'' について $\phi(s'') = \mathbf{T}$ である．

$(\Leftarrow)$ $\phi(s) = \mathbf{F}$ であるとする．$l = 0$ のときには $\phi(s) = \mathbf{F}$ は自明である．

手続き W のあるステップで $1 \leq i \leq n$ である i に対し $\phi(s) = \mathbf{F}$ であるとする．次にどこかの葉が延長される．延長した葉 l のシーケントの ϕ による値が $\mathbf{T}$ の場合には，シーケントの中による値が $\mathbf{F}$ である葉はそのままである．$\mathbf{F}$ の場合には，l の部分で延長されて l は葉ではなくなるが，補題 L1 より，やはり延長した先の一つの葉のシーケント s' で $\phi(s') = \mathbf{F}$ となる． □

上の手続き W により aLK での s の証明図の候補が作られ，葉の部分に現れるシーケントは全て命題変数の列から成るものとなっている．そうでなければ停止していない．これが aLK の証明図であるためには，それらのシーケントが全て TC のインスタンスとなっていればよい．’

補題 15. 全て命題変数からなるシーケントを $s \equiv \Gamma \vdash \Delta$ とする．s がトートロジーであることと，Γ と Δ に共通変数がある，即ち TC のインスタンスであることは同値である．

証明: 付値を ϕ とする．共通変数 A があるとする．$\phi(A) = \mathbf{T}$ の場合には $\phi(\Delta_\vee) = \mathbf{T}$ であり，$\phi(\Gamma_\wedge \to \Delta_\vee) = \mathbf{T}$ となる．$\phi(A) = \mathbf{F}$ の場合には $\phi(\Gamma_\wedge) = \mathbf{F}$ であり，やはり $\phi(\Gamma_\wedge \to \Delta_\vee) = \mathbf{T}$ となる．共通変数がないとする．$\Gamma \equiv= A_1, \ldots, A_n, \Delta \equiv B_1, \ldots, B_m$ とする $(n, m \geq 0)$．$1 \leq i \leq n, 1 \leq j \leq m$ であるようなすべての i, j に対し $\phi(A_i) = \mathbf{T}, \phi(B_j) = \mathbf{F}$ となる ϕ により $\phi(\Gamma_\wedge \to \Delta_\vee) = \mathbf{F}$ となる． □

定理 16 (aLK の完全性). シーケント s がトートロジーであれば体系 aLK で証明図を持つ．

証明: s に対し上の手続き W を適用する．補題 14 により葉の部分に現れるシーケントはすべてトートロジーである．補題 15 により，それらはすべて aLK の公理である． □

系 17 (LK の完全性). シーケント s がトートロジーであれば体系 LK で s の証明図を作れる．

証明: aLK は完全なので aLK の公理と推論規則で s の証明図 T を構成できる．aLK の公理スキーマは LK の定理スキーマ，aLK の推論規則は LK の推論規則か派生規則なので T を LK の証明図に変換できる． □

系 18 (LK の cut 除去定理). シーケント s の LK での証明図があるとする．すると規則 cut を使わない LK での s の証明図を作れる．

証明: LK は健全なので s はトートロジーである．上の系の証明に出てくる方法で s の LK での証明図を作れる．その際，aLK の規則には cut は含まれないし，aLK の公理や派生規則の証明にも cut を使用していない．また，変換により cut が出てくることもないためである． □

LK の cut 除去定理は述語論理版を Gentzen が初めて証明した．その証明は cut を使った証明図を変形して cut を使わなくする手続きを与えるものである．その手続きは必ず停止するし，証明は純粋に構文論的なものである．本稿で説明した証明はそうではなく，元の証明図を変形するものではないし，証明の過程でトート

ロジーなどの意味論的な概念を用いている．また，命題論理の場合にしか適用できないものである．

LK の cut 除去定理から LK の無矛盾性を証明できる．LK の無矛盾性は $\vdash$ が証明図を持たないということと同値であった．LK で $\vdash$ が証明図を持つことを仮定する．すると cut 除去定理により LK で cut を含まない $\vdash$ の証明図が存在する．しかし $\vdash$ が下段に来ることができる規則は cut 以外に存在しない．これは矛盾である．結局，Gentzen の証明の場合には，LK の純粋に構文論的な無矛盾性の証明を与えているといえる．

シーケント s への手続き W の適用後に葉の部分の各論理式 s_i がすべて TC のインスタンスであるかどうかをテストするという手続きは，シーケント s がトートロジーであることの決定手続きとなる．補題 14 により s がトートロジーであることと，s_i がすべてトートロジーであることが同値となるからである．論理式がトートロジーでない時に yes と答えるという問題は NP 完全問題なので，この手続きは (少なくとも今の所) 多項式時間で実行できないはずである[*24]．

手続き W は，論理式 X の CNF を与えるものでもある．具体的には，シーケント $\vdash X$ に手続き W を適用して得られた木の葉の部分のシーケントを $s_1 \equiv \Gamma_1 \vdash \Delta_1, \ldots, s_n \equiv \Gamma_n \vdash \Delta_n$ とする．$C_i \stackrel{\text{def}}{\equiv} (\neg\Gamma_i, \Delta_i)$ として節 C_i を定義し，$Y \stackrel{\text{def}}{\equiv} C_1 \wedge \cdots \wedge C_n$ とすれば補題 14 により Y は X の CNF である．ここに $\neg\Gamma$ は Γ の各論理式に $\neg$ をつけた列である．ただし，Γ_i と Δ_i に同じ命題変数が複数回出現していたら一つに減らす，Γ_i と Δ_i に共通変数がある C_i は除くなど，単純化を行ってもよい．

演習問題

91. Łukasiewicz の 3 つの公理スキーマを A_1, A_2, A_3 とし，それらの異なるメタ変数にそれぞれ異なる命題変数を代入したものを A'_1, A'_2, A'_3 とする．$\vdash A'_1$,$\vdash A'_2$,$\vdash A'_3$ に対し手続き W を適用し，葉の部分のシーケントがすべて TC のインスタンスであるかどうか調べよ．
92. 排中律のスキーマ $X \vee \neg X$ に対し 91 と同様のことを行え．
93. $X \stackrel{\text{def}}{\equiv} \vdash \underline{A} \vee \underline{B} \to \neg\underline{A} \wedge \underline{C}$ に対し手続き W を適用し，葉の部分のシーケント s_i で TC のインスタンスでないものがもしあれば，$\nu(s_i) = \mathbf{F}$ となる付値 ν を与え，$\nu(X)$ を求めよ．ただし付値 ν は X に出現する命題変数についてのみ真偽値を記述すれば十分である．

2.26 形式的体系についてのまとめ

1. 形式的体系により論理式の構文論的な正しさを定義する．
2. 形式的体系は，言語，公理 (スキーマ)，推論規則から成る．
3. 古典命題論理の形式的体系の例として Hilbert 流，自然演繹，シーケント計算を挙げた．
4. 公理から始まる推論規則のインスタンスを連ねた木を証明図という．
5. 証明図を持つ論理式，シーケントを定理と呼ぶ．
6. 論理式，シーケントが定理かどうかは一般には半決定可能であるように形式的体系を作る．
7. 恒真式 (トートロジー) のみに証明図がある形式的体系を (弱い意味で) 健全であると言う．
8. 逆にどの恒真式にも証明図があるとき (弱い意味で) 完全であると言う．
9. 形式的体系 LK(と他の 2 つの体系) は (弱い意味で) 健全かつ完全である．
10. シーケントあるいは論理式がトートロジーであるかどうかの構文論に基づく決定手続きがある．
11. 同じ手続き (と若干の付加的な操作) により論理式の CNF を求めることができる．

[*24] P≠NP である場合か，P=NP だが NP 完全問題に多項式時間で答えるアルゴリズムが求められるまでは．

12. 古典命題論理の場合には定理の集合は決定可能となる．

2.27 命題論理のコンパクト性

ここで命題論理の場合のコンパクト性を示しておく．これは無限かもしれない命題論理式の集合 $G \subseteq \mathbf{Pro}$ について，その任意の有限部分集合 G' に対し G' に依存する一つの付値によって G' の要素すべてが充足可能であれば，G の要素すべてが一つの付値によって充足可能になるという性質である．これは後で出てくる一階述語論理についての定理であるエルブランの定理 (3.10 節) を示すのに使うことができる．エルブランの定理は命題論理と述語論理の橋渡しをする定理であり，形式的体系の定理の計算機による自動証明の基礎となる定理でもある．

■論理式の集合からの演繹 $G \subseteq \mathbf{Pro}$ とするとき，G の元から成る有限列 Π があって，$\Pi, \Gamma \vdash \Delta$ が LK で証明できるとき，$\Gamma \vdash \Delta$ は G から演繹されるといい，$G, \Gamma \vdash \Delta$ あるいは $\Gamma \vdash_G \Delta$ と書く (表記法の細かい部分の流儀はいろいろあるので授業で用いた表記法が使われていない場合もある)．つまり，

$$\Gamma \vdash_G \Delta \text{ iff ある有限な } \Pi \subseteq G \text{ により } \Pi, \Gamma \vdash \Delta$$

である．$X \in \mathbf{Pro}$ があって $G \vdash X$ かつ $G \vdash \neg X$ のとき，G は矛盾していると言う．そうでないとき，G は無矛盾であると言う．

補題 19. $G \subseteq \mathbf{Pro}$ とする．G が充足可能であれば G は無矛盾である．

証明: G が充足可能であるとすると，ある付値 ν が存在して $\nu \models G$ であり，従って $\Gamma_s \subseteq G$ であるような任意の有限列 Γ に対しても $\nu \models \Gamma_\wedge$ である．

すると $G \vdash X$ であるような任意の $X \in \mathbf{Pro}$ について，LK の健全性より $\nu \models X$ である．どのような ν によっても $\nu \models X$ かつ $\nu \models \neg X$ であることはないので，G は無矛盾である． □

無矛盾な $G \subseteq \mathbf{Pro}$ が，集合の包含関係で極大，即ち $G' \supsetneq G$ である任意の $G' \subseteq \mathbf{Pro}$ が矛盾しているとき，G は極大無矛盾 (maximal consistent) であると言う．

補題 20. $G \subseteq \mathbf{Pro}$ が無矛盾かつ $X \in \mathbf{Pro}$ とする．

1. $G \vdash X$ ならば $G' \stackrel{\text{def}}{=} G \cup \{X\}$ も無矛盾である．
2. $G \nvdash X$ ならば $G' \stackrel{\text{def}}{=} G \cup \{\neg X\}$ も無矛盾である．

証明: 1. $\Gamma'_s, \Delta'_s \subseteq G'$ である有限列 Γ', Δ' について $\Gamma' \vdash Y$ と $\Delta' \vdash \neg Y$ の証明図があるとする．Γ', Δ' から X を除いたものを Γ, Δ とすると，$\Gamma, \Delta, X \vdash Y \wedge \neg Y$ の証明図が存在する．$G \vdash X$ なので $\Pi_s \subseteq G$ である有限列 Π により $\Pi \vdash X$ である．従って $\gamma, \delta, \pi \vdash Y \wedge \neg Y$ の証明図が存在する．$\Gamma_s, \Delta_s \subseteq G$z なので G は矛盾している．

2. 対偶を示す．$\Gamma'_s, \Delta'_s \subseteq G'$ である有限列 Γ', Δ' について $\Gamma \vdash Y$ と $\Delta' \vdash \neg Y$ の証明図があるとする．Γ', Δ' から $\neg X$ を除いたものを Γ, Δ とすると，$\Gamma, \Delta, \neg X \vdash Y \wedge \neg Y$ の証明図が存在し，ゆえに $\Gamma, \Delta, \neg X \vdash$ の証明図が存在し，従って $\Gamma, \Delta \vdash X$ の証明図が存在する．$\Gamma_s, \Delta_s \subseteq G$ なので $G \vdash X$ となる．

□

補題 21. $G \subseteq \mathbf{Pro}$ が極大無矛盾の時，任意の $X \in \mathbf{Pro}$ に対し次が成り立つ．

1. $G \vdash X$ なら $X \in G$ である．
2. $X \in G$ かつ $\neg X \in G$ ということはない．
3. $X \in G$ 又は $\neg X \in G$ である．

証明: 1. $G \vdash X$ かつ $X \notin G$ とする．$G' \stackrel{\text{def}}{=} G \cup \{X\}$ とすると補題 20 の 1 により $G' \supsetneq G$ は無矛盾である．従って G は極大ではない．

2. もしそうであれば G は矛盾している．
3. $X \notin G$ であるとする．1 より $G \nvdash X$ である．$G' \stackrel{\text{def}}{=} G \cup \{\neg X\}$ とすると補題 20 の 2 より G' は無矛盾である．G の極大性から $G' = G$ であり，$\neg X \in G$ である．

□

補題 22. $G \subseteq \mathbf{Pro}$ が無矛盾であるとする．任意の $X \in \mathbf{Pro}$ に対し $X \in G$ あるいは $\neg X \in G$ であれば，G は極大無矛盾である．

証明: $Y \notin G$ とすると仮定より $\neg Y \in G$ なので $G \cup \{Y\}$ は矛盾する． □

補題 23. $G \subseteq \mathbf{Pro}$ が無矛盾であるとき，$G' \supseteq G$ となる極大無矛盾な $G' \subseteq \mathbf{Pro}$ が存在する．

証明: **Pro** を整列して番号をつけ，$X_1, X_2, \ldots$ とする．これは命題変数の集合 V が高々可算無限個という本稿のような定式化の場合にはそのようにできる．まず $G_0 \stackrel{\text{def}}{=} G$ とする．これは無矛盾である．$i \geq 1$ について以下のようにして G_i を定める．

$$G_i \stackrel{\text{def}}{=} \begin{cases} G_{i-1} \cup \{\neg X_i\} & (G_{i-1} \nvdash X_i) \\ G_{i-1} \cup \{X_i\} & (G_{i-1} \vdash X_i) \end{cases}$$

G_{i-1} が無矛盾であれば補題 20 により G_i も無矛盾であり，数学的帰納法によりすべての i について G_i は無矛盾である．$G' \stackrel{\text{def}}{=} \bigcup_{i=0}^{\infty} G_i$ として G' を定義する．G' が矛盾しているとするとその矛盾を導く有限個の前提を含む G_i があり，G_i も矛盾していることになるので G' は無矛盾である．列 $X_1, X_2, \ldots$ にはすべての論理式が現れているので，G' の作り方から任意の $Y \in \mathbf{Pro}$ について $Y \in G'$ または $\neg Y \in G'$ である．従って補題 22 により G' は極大である． □

結果として得られる G' は論理式の列の与え方に依存することに注意してほしい．ちょうど一回ずつ **Pro** の各論理式を $X_1, X_2, \ldots$ として出力してゆく手続きが存在する．つまり **Pro** は帰納的可算である．有限で無矛盾な G を与えると，それを含む極大無矛盾かつ決定可能な集合 G' を与えられることもわかる．G が有限集合であれば各 G_i も有限であり，$G_i \vdash X$ は命題論理なので決定可能である．G' に含まれるかどうか判定するために与えられた X が X_i として現れるまで G_i を生成し，X が G_i に含まれるか判定すればよい．

補題 24. $G \subseteq \mathbf{Pro}$ を論理式の無矛盾な集合とする．

- G が極大無矛盾であれば，$A \in V$ に対し

$$\nu_G(A) \stackrel{\text{def}}{=} \begin{cases} \mathbf{T} & (A \in G) \\ \mathbf{F} & (\neg A \in G) \end{cases}$$

として付値 ν_G を定義すると，$\nu_G \models G$ である．

- G のモデルが存在する．

証明: 1. まず，G が極大無矛盾だから $A \in V$ に対し $A \in G$ か $\neg A \in G$ のどちらかなので $\nu_G : V \to \{\mathbf{T}, \mathbf{F}\}$ が定義される．$X \in \mathbf{Pro}$ の構成による帰納法により $X \in G$ iff $\nu_G(X) = \mathbf{T}$ を示す．
まず，$X \in V$ のときには ν の定義から $X \in G$ iff $\nu_G(X) = \mathbf{T}$ である．$X, Y \in \mathbf{Pro}$ について $X \in G$ iff $\nu_G(X) = \mathbf{T}$ と $Y \in G$ iff $\nu_G(Y) = \mathbf{T}$ が成り立つと仮定すると以下が成り立つ．ただし iff が一番結合力が弱い．
$\neg X \in G$ iff $X \notin G$ iff $\nu_G(X) \neq \mathbf{T}$ iff $\nu_G(X) = \mathbf{F}$ iff $\nu_G(\neg X) = \mathbf{T}$ である．
$X \wedge Y \in G$ iff $X, Y \in G$ iff $\nu_G(X) = \nu_G(Y) = \mathbf{T}$ iff $\nu_G(X \wedge Y) = \mathbf{T}$ である．
$X \vee Y \in G$ iff $X \in G$ または $Y \in G$ iff $\nu_G(X) = \mathbf{T}$ または $\nu_G(Y) = \mathbf{T}$ iff $\nu_G(X \vee Y) = \mathbf{T}$ である．
$X \to Y \in G$ iff $X \notin G$ または $Y \in G$ iff $\nu_G(X) = \mathbf{T}$ または $\nu_G(Y) = \mathbf{T}$ iff $\nu_G(X) = \mathbf{F}$ または $\nu_G(Y) = \mathbf{T}$ iff $\nu_G(X \to Y) = \mathbf{T}$ である．
ここでそれぞれの最初の同値は G の極大無矛盾性から導かれる．
2. 補題 23 より G を含む極大無矛盾な $' \subseteq \mathbf{Pro}$ が存在する．1 より $\nu_{G'} \models G'$ であり，ゆえに $\nu_{G'} \models G$ である．

□

定理 25 (命題論理におけるコンパクト性). $G \subseteq \mathbf{Pro}$ とする．すべての有限な $G' \subseteq G$ について G' が充足可能であれば，G は充足可能である．

証明: G が矛盾しているとすると，矛盾を導く論理式の有限集合 $G' \subseteq G$ が存在するが，それは補題 19 から充足不能であり，仮定に反する．従って G は無矛盾である．すると補題 24 より G は充足可能である． □

この定理をコンパクト性定理と呼ぶのは，位相空間論におけるコンパクト性に対応するからである．この部分の理解には位相空間論の初歩の知識が必要である．ただし理解できなくても本稿の他の部分には影響しない．命題変数全体の集合 V からの附値の $\phi : V \to = \mathbf{TV}$ 全体の集合 $\mathbf{TV}^V$ にカントール位相を入れる．即ち $\mathbf{TV} = \{\mathbf{T}, \mathbf{F}\}, V = \{\underline{A_1}, \underline{A_2}, \ldots\}$ とし，$s \in \mathbf{TV}^*$ に対し $B_s \stackrel{\text{def}}{=} \{\phi : V \to \mathbf{TV} \mid \phi(\underline{A_i}) = s_i \ for\ 1 \leq \forall i \leq \text{length}(s)\}, \mathcal{B} \stackrel{\text{def}}{=} \{B_s \mid s \in \mathbf{TV}^*\}$ とする．ここで s_i は列 s の i 番目の文字 (この場合は $\mathbf{TV}$ の要素)，length(s) は列 s の長さである．$\mathcal{B}$ を基本開集合として $\mathbf{TV}^V$ に位相を入れると，コンパクトとなることが知られている．この場合各基本開集合は開閉 (開集合かつ閉集合) となっている．

各命題論理式 $X \in \mathbf{Pro}$ は有限個の命題変数を含むので $S_X \stackrel{\text{def}}{=} \left\{\phi \in \mathbf{TV}^V \mid \phi(X) = \mathbf{F}\right\}$ とすると開閉となる．式中の $\mathbf{F}$ を $\mathbf{T}$ としても開閉である．$G \subseteq \mathbf{Pro}$ が充足不能 iff $\bigcup_{X \in G} S_X = \mathbf{TV}^V$ である．iff の右側は $\mathbf{TV}^V$ がコンパクトなので $\exists G' \subseteq G\ G' : \text{finite} \wedge \bigcup_{X \in G'} S_X = \mathbf{TV}^V$ と同値である．$\bigcup_{X \in G'} S_X = \mathbf{TV}^V$ iff G' が充足不能なので，結局 G が充足不能 iff G のある有限集合 G' が充足不能を示した．これの $\Rightarrow$ 方向の対偶をとったものが上の定理である．

2.28 一般の SAT 問題の 3SAT への帰着

この小節では古典命題論理式 X を，充足可能性を保存しながら CNF S_3 に変換する手続きの例を示す．ここで示す例では S_3 の各節が高々 3 つのリテラルから成る．各節のリテラルが 3 個以下の CNF に限った決定問題を 3SAT と呼ぶ．即ちここで示すのは，CNF に限定しない一般の SAT から 3SAT への変換手続きの例

である．

S_3 の大きさは X の大きさの或る線形函数で抑えられる．実現方法を工夫すれば変換の実行時間はやはり線形函数で抑えられる．なお SAT 問題という言い方をすると CNF である場合に限っている場合も多いが，ここでは X は一般の古典命題論理式であり，更に $\wedge, \vee, \rightarrow$ 以外の二項論理演算子を含んで構わない (次段落の $\rightleftarrows$ を除く．但し同じ論理函数の演算子，例えば $\leftrightarrow$ を含んでよい)．但し X は論理定数を含まないものとしている．

もしも含む場合を考えたいのであれば多項式時間の前処理により含まないように簡単化しておくことで対応できるし，あるいは以下の手続きを拡張することでも対応できるだろう．

手続きを示す前に論理演算子を新たに 1 つ導入して論理式の文法を拡張する．まず新しい二項論理演算子 $\rightleftarrows$ を付け加え元の文法を拡張する．但しこの演算子は本小節内でのみ変換目的で用い，トップレベルにしか出現せず，しかも $A \rightleftarrows X$ という形 (A は命題変数で，X は拡張前の文法での任意の論理式) でのみ現れるとする．演算子としての表示上の優先順位は最も低いとする．$\rightleftarrows$ に対応する真理函数は $\leftrightarrow$ と同じであるとする．

■変換の手続き: 最初に拡張前の文法の論理式 ($\rightleftarrows$ が出現しない論理式)X に対し集合 $S_0 \stackrel{\text{def}}{=} \{X\}$ とする．S_0 に対し以下の 3 つの段階の変換手続きを順に適用する．

第一段階では $S = S_0$ に対して次のような書き換え (其々を書き換えの 1 ステップとする) を，当てはまるパターンがなくなるまで繰り返し行う (1 以外の記述では{と }$\cup S'$ の部分を略している)．op は $\rightleftarrows$ 以外の二項演算子，A, B は命題変数，A', B' は新しい (即ち S 中に出現しない) 互いに異なる命題変数を表している．4 の $\Rightarrow$ の左側はリテラル数 3 以下の節である場合を除く．例えば $A \vee (\neg B \vee C)$ や $\neg A$ は書き換えられないが，$A \vee (B \vee (C \vee \neg D))$ は書き換えられる．上方の行の規則の方が優先適用される．

1. $\{X \wedge Y\} \cup S' \Rightarrow \{X, Y\} \cup S'$,($S = \{X \wedge Y\} \cup S'$ 全体を $\{X, Y\} \cup S'$ に書き換える．略記すると $X \wedge Y \rightarrow X, Y$ である),
2. $\neg\neg X \Rightarrow X$,
3. $\neg X \Rightarrow \neg A', A' \rightleftarrows X$,
4. $X \ op \ Y \Rightarrow A', A' \rightleftarrows X \ op \ Y$.

書き換えは S 中の各論理式のトップレベルにのみ適用される．$S_1 \stackrel{\text{def}}{=}$ 第一段階終了時の S とする．

第二段階では，$S = S_1$ に対し以下の書き換えを同様に行う．但し $\mathrm{cnf}(X)$ は X の CNF(節の集合扱いし，やはり列として表記している) を表す．$\mathrm{cnf}(X)$ には $\rightleftarrows$ が現れないことに注意．

1. $A \rightleftarrows \neg X \Rightarrow \neg \vee B', A \vee B', B' \rightleftarrows X$,
2. $A \rightleftarrows X \ op \ Y \Rightarrow \mathrm{cnf}(A \leftrightarrow (B' \ op \ C')), B' \rightleftarrows X, C' \rightleftarrows Y$

第二段階を終了すると，S 中の $A \rightleftarrows X$ というパターンの論理式について，X は命題変数である (それ以外のパターンは 1,2 どちらかにマッチするため)．$S_2 \stackrel{\text{def}}{=}$ 第二段階終了時の S とする．第三段階では $S = S2$ に次の書き換えを同様に行う．終了時の S を変換結果 S_3 とする．

1. $A \rightleftarrows B \Rightarrow \neg A \vee B, A \vee \neg B$,

以下に第一段階の変換の例を示す．命題変数のアンダーラインを略している．

$$
\begin{aligned}
&(A \vee B) \wedge ((\neg\neg(\neg A \vee (B \vee C))) \wedge (\neg A \vee (B \vee (C \vee D))))\\
\Rightarrow &A \vee B, (\neg\neg(\neg A \vee (B \vee C))) \wedge (\neg A \vee (B \vee (C \vee D)))\\
\Rightarrow &A \vee B, \neg\neg(\neg A \vee (B \vee C)), \neg A \vee (B \vee (C \vee D))\\
\Rightarrow &A \vee B, \neg A \vee (B \vee C), \neg A \vee (B \vee (C \vee D))\\
\Rightarrow &A \vee B, \neg A \vee (B \vee C), E, E \rightleftarrows \neg A \vee (B \vee (C \vee D))
\end{aligned}
$$

続いて第二段階では $\rightleftarrows$ を含む式のみ変換される．その部分のみ表記する．含まない式はそのまま S_2(及び S_3) の要素となる．[] 内はそのステップで確定し，結果の S_2 が含む部分である．

$$
\begin{aligned}
&[A \vee B, \neg A \vee (B \vee C), E], E \rightleftarrows \neg A \vee (B \vee (C \vee D))\\
\Rightarrow &[\neg E \vee (F \vee G), E \vee \neg F, E \vee \neg G], F \rightleftarrows \neg A, G \rightleftarrows B \vee (C \vee D)\\
\Rightarrow &[\neg F \vee \neg H, F \vee H, H \rightleftarrows A], G \rightleftarrows B \vee (C \vee D)\\
\Rightarrow &[\neg G \vee (I \vee J), G \vee \neg I, G \vee \neg J, I \rightleftarrows B], J \rightleftarrows C \vee D\\
\Rightarrow &[\neg J \vee (K \vee L), J \vee \neg K, J \vee \neg L, K \rightleftarrows C, L \rightleftarrows D]
\end{aligned}
$$

S_2 は上記の [] の中身を全て合併したものである. 第三段階では 2 つの命題変数が $\rightleftarrows$ で結合された論理式が全て書き換えられ, 以下になる. $S_3 = \{A \vee B, \neg A \vee (B \vee C), E, \neg E \vee (F \vee G), E \vee \neg F, E \vee \neg G, \neg F \vee \neg H, F \vee H, \neg H \vee A, H \vee$ である．例を見てわかる様に，改良すれば効率化 (即ち少ない追加命題変数，少ない結果の節) 可能である．

補題 26. (上の) 第一段階は停止する．

証明: S 中の，$\rightleftarrows$ を含まない論理式で，なおかつリテラル個数が 3 以下の節以外のものについて，それらの中の演算子の出現個数の合計を n とすると有限であり，1 ステップ毎に 1 以上減る． □

補題 27. 第一段階の各ステップで充足可能性が保存される．

証明: (省略していない)1 ステップ書き換え $\{X\} \cup S' \Rightarrow S'' \cup S'$ に対し充足可能性について ($\Rightarrow$) を示す時 $\phi \models \{X\} \cup S'$，($\Leftarrow$) の時 $\psi \models S'' \cup S'$ とする．
(1,2 の場合) そのまま $\phi \models S'' \cup S'$ あるいは $\psi \models \{X\} \cup S'$ である．
(3$\Rightarrow$) $X \equiv \neg X'$ とすると $\phi[\mathbf{F}/A'] \models \{\neg A', A' \rightleftarrows X'\} \cup S'$ である．ここで

$$
\phi[v/A](B) \stackrel{\text{def}}{=} \begin{cases} v & (B \equiv A) \\ \phi(B) & (B \not\equiv A) \end{cases}
$$

である．
(3$\Leftarrow$) そのまま $\psi \models \{X\} \cup S'$ である．
(4$\Rightarrow$) $X \equiv X' \mathit{op}\, Y'$ とすると $\phi[\mathbf{T}/A'] \models \{A', A' \rightleftarrows X' \mathit{op}\, Y'\} \cup S'$ である．
(4$\Leftarrow$) そのまま $\psi \models \{X\} \cup S$ である． □

補題 28. S_1 の要素はリテラル数 3 以下の節であるか，$A \rightleftarrows X$ の形の式のどちらかである．

証明: それら以外の論理式があると書き換えの対象となるため手続きは終了していない． □

補題 29. 第二段階は停止する．

証明: S 中の $A \rightleftarrows X$ の形の論理式の X の長さの合計値がステップ毎に減るため． □

補題 30. 第二段階の各ステップで充足可能性が保存される．

証明: 補題 27 と同様に証明する．
(1⇒) $X \equiv A \rightleftarrows \neg X'$ とすると $\phi[\neg\phi(A)/B'] \models \{\neg A \vee \neg B', A \vee B, B' \rightleftarrows X'\} \cup S'$ である．
(1⇐) そのまま充足する．
(2⇒) $X \equiv A \rightleftarrows X' \mathit{op} Y'$ とすると $\phi[\neg\phi(X')/B'][\phi(Y')/C'] \models \{\mathrm{cnf}(A \rightleftarrows (B' \mathit{op} C')), B' \rightleftarrows X', C' \rightleftarrows Y'\} \cup S'$ である．
(2⇐) そのまま充足する． □

補題 31. 命題変数が n 個の古典命題論理式 X の CNFX' の各節は，高々 n 個のリテラルから成る或る節と同値である．

証明: まず $\wedge, \vee$ 以外の二項演算子を $\neg, \wedge, \vee$ により表し，CNFX' を求められる．$n+1$ 個以上のリテラルを含む X' の節を C とする．すると少なくとも一つの命題変数 A について A と $\neg A$ 両方が C に出現するか，あるいはどちらかが重複して C に出現している．それゆえ $C \models\!\models T$ であるか，C のリテラルを減らして同じ推論を繰り返せる． □

補題 32. S_2 はリテラル数 3 以下の節と $A \rightleftarrows B$ の形の式から成る $(A, B \in V)$.

証明: 補題 28 と，第二段階の書き換え 1.2 の結果中の，$A \rightleftarrows X$ 以外の形の論理式がリテラル数 2 の節形式であること，3 の場合には cnf の部分の命題変数の数が 3 に対し補題 31 を適用すること，$A \rightleftarrows X (X \notin V)$ の形の論理式が S 中にある場合には第二段階が終了していないことによる． □

補題 33. S_3 はリテラル数 3 以下の節から成る．

証明: 上の補題と，第三段階で $A \rightleftarrows B$ が $\neg A \vee B, A \vee \neg B$ に書き換えられるため． □

以下では論理式を計算機中でポインタにより木の形で表現するとする．具体的には C 言語のポインタと struct を利用するか，あるいは LISP を利用する場合を考えればよい．すると論理式のトップレベルから定数レベル下までのパターンを分類したり，論理式をトップレベルの論理演算子の引数に分解したり，逆に特定の演算子の引数を与えて組み立てたりするのはどれも定数時間で実行可能である．ここではそのような実現方法と採っているとして各段階の実行時間と結果の大きさについて考える．なお定数分余分に必要な点については省略して記述している．例えば定数倍で抑えられる，などと書いた場合，正確には或る一次式で抑えられる，という意味である．

補題 34. 第一段階の実行時間は S_0 の大きさの定数倍で抑えられる．結果のデータ S_1 の大きさ (複数の論理式を表すデータをリストとして表すのに必要なメモリも含める) は S_0 の大きさの定数で抑えられる．

証明: 第一段階の書き換えの回数は S_0 中の演算子の数以下である．パターンマッチにかかる時間の合計は S_0 の大きさの定数倍である．追加で利用するメモリの領域は書き換えの回数の定数倍である． □

補題 35. 第二段階の実行時間は S_1 の大きさの定数倍で抑えられる．結果のデータ S_2 の大きさは S_1 の大きさの定数で抑えられる．

証明: 上と同様である． □

補題 36. 第三段階の実行時間は S_2 の大きさの定数倍で抑えられる．結果のデータ S_3 の大きさは S_2 の大きさの定数で抑えられる．

証明: これも同様である． □

結果的に以下が証明された．

系 37. 一般の論理式 X についての SAT 問題を 3SAT に変換する上の手続きについて，変換結果の大きさは X の大きさの一次式で抑えられる．仮定した実現方法を採った場合には，実行時間も同様に一次式で抑えられる．

系 38. 3SAT は NP 完全である．

証明: 3SAT の問題は SAT 問題なので 3SAT はクラス NP に属する．NP 困難な SAT が上の系により SAT に線形時間帰着可能ゆえ 3SAT も NP 困難であり，併せると NP 完全である． □

3 古典一階述語論理

この節で古典一階述語論理 (classical first-order predicate logic) に付いて述べる (以下，一階述語論理と記す)．この論理を用いると，ある種の数学を記号的・形式的に取り扱うことが可能になる．実際にはその領域の数学と議論を全て枠内で取り扱えるという保証があるわけではないが，経験的には問題がないし，そうできると信じられてもいる．

一階述語論理を用いると，例えば以下のような論理式を取り扱うことが可能になる．

$$\forall x, y, z\ x \cdot (y \cdot z) = (x \cdot y) \cdot z, \tag{16}$$
$$\forall x \forall \epsilon (\epsilon > 0 \to \exists \delta\ \delta > 0 \wedge \forall x' (|x' - x| < \epsilon \to |f(x') - f(x)| < \epsilon)), \tag{17}$$
$$\forall x, y (x \leq y \wedge y \leq x \to x = y), \tag{18}$$
$$\forall x, y, z (x \in y \cup z \leftrightarrow x \in y \vee x \in z), \tag{19}$$

述語というのは，引数を与えられると真偽値が定まるもののことで，命題論理の命題が引数によらずに真偽 値が定まったのとは対照的である．例えば等号 $=$ は両側の 2 つの引数が定まると真 (等しい時) か偽 (それ以外の時) かが定まるような述語である．ただし述語論理と命題論理の違いは，函数記号を扱うことや，命題ではなく対象の上を動く変数があることなど他にもある．古典論理というのは，もともと数学で広く使われている自然言語での論理に対応するような，研究の歴史が長い論理ということで，その基本的な意味論で真偽値として真と偽 2 つの値を取るものを考えるので二値論理 (two-valued logic) とも言う．一階 (first-order) というのは，数学の理論が取り扱う領域の要素が変数の値となるし，取り扱う函数や述語もそういった要素に対して値や真偽値が定まるという意味である．領域上の函数や述語 (領域の部分集合) の上を動くような変数がある場合には二階 (second-order) と言う[*25]．さらにそれらの集合の上の函数や関係の上を動く変数があり，さらに... というようなものを高階 (higher-order) と言う．

一階述語論理により，集合，自然数や，半群，モノイド，群，環，体などの代数的な数学的対象に付いてそれらの性質を論理式で表現し，証明を行えるようになる．例えばそのようにして集合論や自然数論を一階述

[*25] 一階述語論理に基づいた数学の例に集合論があり，その場合には変数は集合の上を動くと考えられるが，領域の元が集合であるため一階の範疇に入る．

語論理の中で展開できる．前者を公理的集合論 (axiomatic set theory) と呼び，後者にはペアノ数論 (Peano arithmetic) という標準的な方法がある．

これらに対し，解析学 (に相当する数学) を展開するには二階の論理を用いる必要がある．ただし，集合論の中で実数に対応する集合を定義して解析学を行うことは可能なので，その意味では一階述語論理で解析学を取り扱えるとも言える．

以下，命題論理の場合とある程度同じように，一階述語論理の言語，意味論，形式的体系，標準形たちとそれらへの変換手続き，命題論理との関係 (エルブランの定理)，体系の健全性と完全性について説明する．

3.1 一階述語論理の言語

古典一階述語論理で使用する記号と項，論理式とは以下のようなものである．これらを古典一階述語論理の言語と呼ぶ．古典命題論理の場合には命題変数全体の集合を固定すれば言語は 1 つしかなかった．また命題変数全体の集合を可算無限集合の範囲で取り替えても本質的には同じである．そのため言語は事実上 1 つであると考えてよい．しかし古典一階述語論理の場合には言語そのものが変化することに注意してほしい．具体的には定数記号，函数記号，述語記号の集合に依存して変化する．

■論理記号 $\wedge\ \vee\ \neg\ \to\ \forall\ \exists$

命題論理と比べて, $\forall, \exists$ の 2 つが新しい記号である．これらを量化記号 (quantifier symbol) と言う．$\forall$ を全称量化記号 (universal quantifier symbol)，全称記号，$\exists$ を存在量化記号 (existential quantifier symbol)，存在記号などという．

■変数記号 V: 可算無限個の変数記号 (variable symbol) の集合．

対象言語の変数記号: $\underline{x_1}, \underline{x_2}, \ldots, \underline{y_1}, \underline{y_2}, \ldots, \underline{z_1}, \underline{z_2}, \ldots$．変数記号を表すメタ変数としては $x_1, x_2, \ldots, y_1, y_2, \ldots, z_1, z_2, \ldots$ を用いる．

以下，K, F, P についての対象言語の記号も，それらの方に下線をつけて表す．また，誤解がない場合には下線を省略する場合もある．

■定数記号 K: 定数記号 (constant symbol) の集合．

対象言語の定数記号:$\underline{c_1}, \underline{c_2}, \ldots, \underline{d_1}, \underline{d_2}, \ldots$．定数記号を表すメタ変数としては $c_1, c_2, \ldots, d_1, d_2, \ldots$ を用いる．定数記号を arity(引数の数) が 0 の函数記号として扱う流儀もある．

■函数記号 F: 函数記号 (function symbol) の集合．

arity: $F \to \boldsymbol{N}\backslash\{0\}$ が定義され，引数の数を表すとする．

対象言語の函数記号:$\underline{f_1}, \underline{f_2}, \ldots, \underline{g_1}, \underline{g_2}, \ldots, \underline{h_1}, \underline{h_2}, \ldots$．函数記号を表すメタ変数としては $f, g, h, f_1, f_2, \ldots, g_1, g_2, \ldots, h_1, h_2, \ldots$ を用いる．

$F = \{\underline{f}/2, \underline{g}/1\}$ などと表記して，$\mathrm{arity}(\underline{f}) = 2, \mathrm{arity}(\underline{g}) = 1$ を表すことにする．

■述語記号 P: 述語記号 (predicate symbol) の集合．空でないとする．arity:$P \to \boldsymbol{N}$ が定義されているとする．対象言語の述語記号:$\underline{p_1}, \underline{p_2}, \ldots, \underline{q_1}, \underline{q_2}, \ldots, \underline{r_1}, \underline{r_2}, \ldots$．述語記号を表すメタ変数として $p, q, r, \ldots, p_1, p_2, \ldots, q_1, q_2, \ldots$ を用いる．F の場合と同様にして arity を表すことにする．

上記 3 種類の定数・函数・述語記号は，述語記号が空集合でないという条件が付いている以外，一般には追加条件はない．述語記号が空集合だと，後で定義する論理式の集合が空集合となってしまうため，意味がない．但し一階述語論理で取り扱える一般の数学の場合，これら 3 種類の記号は通常有限集合とする場合がほと

んどである．無限集合の場合も考えるのは，そのように一般化してもそれほど問題がないことと，理論展開上必要な場合が出てくるのに備えてのことである．

■補助記号　(,) などの補助的に用いる記号．

■一階述語論理の言語　V, K, F, P がすべて定まると一階述語論理の構成要素の記号やそれらの列あるいは木がすべて定まるので，$L = \langle V, K, F, P\rangle$ で一階述語論理の言語を表すことにする．逆に，L の V, K, F, P をそれぞれ V_L, K_L, F_L, P_L と書いて，言語 L の変数記号，定数記号，函数記号，述語記号の集合を表すことにする．以下の項，原子論理式，論理式の集合は，組 L が定まると決まるので，それぞれ添え字 L を付けている．

■項　以下のように帰納的に定義される集合 TT_L の元を項 (term) という．

(1) 変数記号と定数記号は TT_L の元である．
(2) $t_1, \ldots, t_n$ を TT_L の元とし，f を arity n の函数記号とするとき，$f(t_1, \ldots, t_n)$ も TT_L の元である．

$s, t, s_1, s_2, \ldots, t_1, t_2, \ldots$ を項のメタ変数として用いる．

函数記号を四則演算の $+$ や-, $\cdot$ のように演算子として項を表記する場合がある．演算子の優先順位は，通常は単項の方が他よりも優先順位が高いとする．

■原子論理式　$t_1, \ldots, t_n$ を項とし，p を arity が n の述語記号とするとき，$p(t_1, \ldots, t_n)$ は原子論理式 (atomic formula) の集合 AA_L の元である．

原子論理式も，述語記号を演算子として表記する場合がある．例えば等号 $= /2$ は二項演算子として表記する場合が多い．

■量化子　$x \in V_L$ のとき，$\forall x$ を全称量化子 (universal quantifyer)，$\exists x$ を存在量化子 (existential quantifyer) と言い，両者を併せて量化子 (quantifyer) と呼ぶ．本稿では x をその量化子の変数記号と言う．

■論理式　次のように定義される集合 $\mathbf{Pre}_L$ の元を論理式 (formula) という．論理式を表すメタ変数として E, F などを用いる．

1. 原子論理式は $\mathbf{Pre}_L$ の元である．
2. E, F を $\mathbf{Pre}_L$ の元，x を変数記号とするとき，以下は論理式である．
   (a) $\neg E$
   (b) $E \wedge F$
   (c) $E \vee F$
   (d) $E \to F$
   (e) $\forall x E$
   (f) $\exists x E$

$\forall x_1 \forall x_2 \ldots \forall x_n$ を省略して $\forall x_1, x_2, \ldots, x_n$ と書く場合がある．$\exists$ についても同様である．量化子の右側の論理式に括弧が付けられていない場合，切れ目がわかりにくくなりやすい．そこに若干の空白を入れる，あるいは. を入れるなどして切れ目をわかりやすくしたりする場合がある．

原子論理式と，その前に $\neg$ を一つ付けたものをリテラル (literal) と呼ぶ．リテラルを $\vee$ で有限個結んだものを節 (clause) と呼ぶ．これらは命題論理の場合と同じである．

() の使用を減らすため，論理記号の間に優先順位をつける．本稿では $\neg, \wedge, \vee, \to$ についてはお互いの優先

順位は命題論理の場合 (2.1 節の最後の部分を参照) と 同じで，量化子，即ち $\forall x, \exists x$ の優先順位は $\wedge, \vee$ と $\to$ の間であるとする．即ち本稿では以下の 2 つの論理式は同等である．

$$\forall x\ p(x) \wedge q(x) \to \neg\exists y \neg p(y) \vee q(y)$$
$$(\forall x(p(x) \wedge q(x))) \to \neg\exists y((\neg p(y)) \vee q(y))$$

量化子の優先順位を $\neg$ と同じとする流儀もあるし，$\to$ が連なった場合に右側の $\to$ の式が括弧で囲まれていなくてもそちらが優先される流儀もあるので，注意してほしい．

K_L, F_L, P_L が高々可算無限集合のとき，$\mathbf{Pre}_L$ は可算無限集合，TT_L, AA_L は高々可算無限集合となる．このような言語 L を可算な言語 (countable language) という．

2 つの言語 L_1, L_2 があり，$V_{L_1} = V_{L_2}, K_{L_1} \subseteq K_{L_2}$，さらに arity を含めて $F_{L_1} \subseteq F_{L_2}, P_{L_1} \subseteq P_{L_2}$ のとき，L_2 を L_1 の拡大 (extension) という．このとき，TT_{L_1} の元はそのまま TT_{L_2} の元とみなせるし，$\mathbf{Pre}_{L_1}$ の元はそのまま $\mathbf{Pre}_{L_2}$ の元とみなせる．

なおここで $E \leftrightarrow F \stackrel{\text{def}}{\equiv} (E \to F) \wedge (F \to E)$ として，同値の記号 $\leftrightarrow$ を命題論理と同じように略記法として導入することができる．$\leftrightarrow$ の優先順位は $\to$ と同じである．

以下では, 特に別の言語や言語の拡張を考えない場合には，一つの言語 L を固定し，添え字の L を適宜略す．

■出現　論理式あるいは項の中に変数記号や定数記号，函数記号，述語記号が現れている場合，それらをそれぞれの出現 (occurrence) と呼ぶ．出現と言う場合には，同じ記号が複数回出現している場合，それぞれを区別する．本稿では，量化子 $\forall x, \exists x$ の変数記号は変数の出現と言わないことにする．ただし (変数記号を x とする) 量化子の出現ではあるとする．

変数が出現しない項を基礎項 (ground term)，あるいはグラウンドな項と言う (分野によってはこういった項を閉 (closed) であるという場合もある)．

■部分項　項は以上のように帰納的に定義される．ある項 t を帰納的に作っていく際に途中で出てくる項 s，言い換えると t の一部になっている項 s を t の部分項 (subterm) と呼ぶ．本稿では t 自身も t の部分項であるとする．

■部分論理式　論理式も以上のように帰納的に定義される．ある論理式 E を帰納的に作っていく際に途中で出てくる論理式 F，言い換えると E の一部になっている F を E の部分論理式 (subformula) と呼ぶ．本稿では E 自身も E の部分論理式であるとする．

■変数の束縛と自由な出現　変数 x が量化子 $\forall x$ または $\exists x$ の内側に出現している場合，その出現は束縛されている (bound) と言う．以下で例えば括弧内の x は左側の $\forall x$ や $\exists x$ により束縛されている．ここで $\forall x(\cdots x \cdots)$ は E の部分論理式である．

$$E \equiv \cdots \forall x(\cdots x \cdots) \cdots$$
$$F \equiv \cdots \exists x(\cdots x \cdots) \cdots$$

束縛されている変数の出現からみて，最も内側にある同じ変数記号の量化子の出現が，その変数の出現を束縛する．つまり束縛されている変数の出現には，一意的にそれを束縛する量化子の出現が対応する．量化子の同じ出現に束縛されている変数は，それらと，対応する量化子の変数記号を組織的に別の変数記号に置き換えて

も，後でわかるように論理式の意味や証明可能性は変わらない[*26]．これはプログラムで仮引数や局所変数の名前を変えてもプログラムの動作や意味に変化がないことに相当する．論理式 E 中で，いずれの量化子の出現によっても束縛されない変数 x が出現している時，その出現を x の自由 (free) な出現と呼び，x は E に自由に出現する，と言う．

例:

$$\underline{p}(\underline{f}(\overset{1}{\underline{x}},\overset{2}{\underline{y}}))\wedge\overset{3}{\forall}\underline{x}\ \ (\underline{q}(\overset{4}{\underline{x}},\overset{5}{\underline{y}})\rightarrow(\overset{6}{\exists}\underline{x}\ \ \underline{r}(\overset{7}{\underline{x}},\overset{8}{\underline{y}})\vee\underline{p}(\overset{9}{\underline{x}}))\wedge\underline{p}(\overset{10}{\underline{x}}))$$
$$\underline{p}\big(\underline{f}\big(\underline{x},\underline{y}\big)\big)\wedge\forall\underline{x}\ \big(\underline{q}\big(\underline{x},\underline{y}\big)\rightarrow\big(\big(\exists\underline{x}\big(\underline{r}\big(\underline{x},\underline{y}\big)\vee\underline{p}(\underline{x})\big)\big)\wedge\underline{p}(\underline{x})\big)\big)$$

まず，上式は各論理演算子等の優先順位により，括弧付きで表すと下のような式である．1 は $\underline{x}$ の自由な出現であり，2,5,8 は $\underline{y}$ の自由な出現である．4 と 10 は 3 に，7 と 9 は 6 に束縛されている．つまり 4,7,9,10 は $\underline{x}$ の束縛された出現である．

後でわかるように，3 の変数記号と 4, 10 をすべて $\underline{z}$ に，6 の変数記号と 7,9 をすべて $\underline{x_1}$ に置き換えても論理式の意味は変わらない．ただしそれらの一部のみを置き換えたり，$\underline{z}$ ではなく $\underline{y}$ に置き換えたりするのはだめである．

$\mathrm{FV}(E)\stackrel{\mathrm{def}}{=}\big\{x\in V\mid x$は式$E$中に自由に出現する$\big\}$ として $\mathrm{FV}:\mathbf{Pre}_L\rightarrow 2^V$ を定義する．$\mathrm{FV}(E)=\emptyset$ の時，E は閉 (closed) であると言う．あるいは文 (sentence) とも言う．$\mathbf{Pre}_L^c\subset\mathbf{Pre}_L$ を閉論理式全体の集合とする．変数も量化子も出現しない論理式をグラウンドであると言うことにする．以下，幾つか定義を行う．

■閉包　$\mathrm{FV}(E)=\{x_1,\ldots,x_n\}$ のとき $\forall x_1,\ldots,x_n\ E$ を E の全称閉包 (universal closure)，$\exists x_1,\ldots,x_n\ E$ を E の存在閉包 (existencial closure) と呼ぶ．閉包は両方とも閉論理式となる．

■代入　論理式 E 中の変数 x の自由な出現をすべて項 t で置き換える操作を，変数 x への t の代入 (substitution) と言い，代入の結果を $E[t/z]$ と表記する．ただし，t に出現する変数達が E の量化子によって束縛されてはならない (そのような場合には，t に現れる変数記号と衝突する，E の量化子の変数記号達と，それに束縛される変数の出現達をまったく新しい (つまり E に出現しない) 変数記号達に置き換える操作を行ってから代入することになる)．$E[x/x]\equiv E$ であり，x が E に自由に出現しない場合には $E[t/x]\equiv E$ であることに注意してほしい．

■置き換え　i ごとにそれぞれ定数記号あるいは述語記号として K_L に $[]_i$，あるいは P_L に $[]_i/0$ を $i=1,\ldots,k$ まで付け加えて言語 L' とし，論理式の集合 $\mathbf{Pre}_L$ を改めて定義し，$\mathbf{Pre}_{L,k}$ と記す．また $\mathbf{Pre}_{L,k}$ の元を $E[]_k$ などと表記し，$E[]_k$ の中の記号 $[]_i$ をすべて L の項または論理式 u_i で置き換えたものを $E[u_1,\ldots,u_k]$ と表記することにする．ただし u_i は $[]_i$ が定数記号であれば項，述語記号であれば論理式でなければならない．$E[u_1,\ldots,u_k]\in\mathbf{Pre}_L$ となる．u_i が自由変数を含んでいる場合には，それらの変数が代入によって束縛されてはならない (u_i が項の場合は u_i 中のすべての変数が自由変数である)．

なお，本稿では記号 $[]_i$ を導入して論理式の置き換えを以上のように定義し，記法 $E[u_1,\ldots,u_k]$ を定義したが，直観的な説明で $E[u_1,\ldots,u_k]$ を導入してもよいかもしれない．

[*26] 但し他に出現している変数名に置き換えて，置き換え後の量化子でその出現を束縛するのはまずい．

演習問題

94. 以下の言語を L_{peano} と表記する．

$$K_{peano} \stackrel{\text{def}}{=} \{0\}, \qquad F_{peano} \stackrel{\text{def}}{=} \{s/1, +/2, \cdot/2\}, \qquad P_{peano} \stackrel{\text{def}}{=} \{= /2\},$$

（a）基礎項の例，基礎項でない項の例をそれぞれ 3 つ挙げよ．ここで $+, \cdot, =$ は演算子として記述すること．

（b）項と使用する記号は同じだが，項とはならない記号列を 3 つ挙げよ．

（c）原子論理式の例を 3 つ挙げよ．

（d）$\neg, \wedge, \vee, \rightarrow$ をそれぞれ一つ以上含んで量化子を含まない，閉論理式の例と自由変数を含む論理式の例をそれぞれ 3 つ挙げよ．ただし括弧をできる限り省略すること．

（e）上の問題の回答の全称閉包を求めよ．

（f）$\neg, \wedge, \vee, \rightarrow, \forall, \exists$ をそれぞれ一つ以上含む，閉論理式の例と自由変数を含む論理式の例をそれぞれ 3 つ挙げよ．ただし括弧をできる限り省略すること．

（g）上の問題の回答の存在閉包を求めよ．

95. $(s(x) = s(y) \rightarrow x = y)[s(x)/z][s(y)/y]$ を求めよ．

96. $((\square_1 + \square_1) + x = x + \square_2 \wedge \square_3)[0, s(0), 1 = 1, \neg 0 = 1]$ を求めよ．

97. 使用している記号は同じだが，論理式ではない記号列の例を 3 つ挙げよ．