2012 年 6 月版

易しくない 論理学

科学書刊株式会社:電子版の原稿

「橋梁&都市 PROJECT: 2012」(ISSN 1344 - 7084)

島　田　　静　雄

この冊子は、雑誌「橋梁と都市 PROJECT」に連載することを予定して作成した MS-Word 版の原稿から、PDF 形式に変換したもののコピーです。2010 年度から、出版関係は電子出版を模索した努力をする時代に入りました。科学書刊株式会社雑誌は、この動きに対応するため、ハードコピーとしての「橋梁&都市 PROJECT」の発行を休刊とし、電子化にどのように対応するかの研究を始めました。実を言うと、この傾向は 10 年前から予測されていました。筆者は、この先取りとして、三種類の発表形式を試してきました

一つ目は、雑誌の記事としての形式です。「橋梁と都市 PROJECT」のスタイルは B5 版二段組みです。こちらの方は、しばらく休刊になりました。この原稿は MS-Word で A4 版一段組みで作成してきましたが、そのままで体裁のよいレポート形式になるように注意して編集してあります。この形式のままにしたのが、この PDF 版です。

二つ目、この PDF 版をインターネットで公開することです。すこしページ数が多くなりますが、ユーザは、これをダウンロードして印刷して見ることができます。プリンタをお持ちでなければ、原稿ファイルを USB にして持ち込めば、簡易製本までサービスしてくれる街中の印刷屋さんが見つかりようになりました。PDF 版の WEB サイトは、差し当たり下記にしてあります。

http://www.nakanihon.co.jp/gijyutsu/Shimada/shimadatop.html

三つ目は、パソコンの画面でランダムに項目がアクセスするようにリンクを張った WEB 版です。この利用方法を考えて、筆者の原稿は、約 600 字程度のパラグラフ単位分けてあって、インターネットでのアクセス速度が速くなるように、一つのパラグラフがパソコンの一画面に入るようにしてあります。目次と索引とを参照すれば、かなり便利な検索が使えます。WEB サイトは上の PDF 版と同じ個所です。この冊子は、表紙、目次などの頭の部分が 9 ページ、本文は 53 ページあります。電子出版を考えると、ページ番号で項目位置を探すことが実用的ではありません。したがって、目次と索引は、章・節・項のパラグラフ番号番号で検索するように使って下さい。

(この冊子は、全 62 ページあります)

目　　次

用 語 索 引

前書き

この報文の原稿は、学術レポートを作成するときの教材として作成した「**実用文書のまとめ方**」
http://www.nakanihon.co.jp/gijyutsu/Shimada/bunsyo/top.html
の副読本として1993年に私的な教材として編集しておいたものです。実用文書を作ることは技術です。この技術には三つの要素があります。第一は、正しい文章が書けること、第二が文書の書式、第三が全体としての体裁です。第一の要素には、学問としての論理学(logic)や言語学(linguistics)の素養が大切です。これらの学問は、従来、文科系の専門と考えられていました。大学の教育現場では、レポートや論文の作成を指導しておかなければなりません。これらは、文学的で自己満足的な作品とは異なり、話しかける相手が正確に理解できることに目的がありますので、実用文書とくくります。実践的な作文指導の項目は、英語などの一般教養や、論理学の学問的素養を踏まえます。論理学については、理科系・文科系に関わりなく、実用を意識した参考書や系統的な教育カリキュラムがあるのが理想です。しかしその目的に沿うような、まとまった著作物が見当たりませんでした。そこで、1993年の時点で幾つかの書籍を参考にしてパンフレット「論理学アブストラクト」にまとめ、受講者に配布してきました。2000年以降、インターネットの利用が便利になりましたので、筆者が私的にまとめておいた教材をWEB版として順次発表しています。2010年からは、出版関係は電子出版を模索して努力をする時代に入りました。この動きは以前から予測されていたことです。電子出版の現状は、新聞や雑誌のような、一過性の読み捨て式の使い方が意識されています。従来のハードコピーの形でも参照して利用することも、平行して準備しました。この報文の原稿は、A4版のMS-Wordとして体裁よくまとめ、そのままPDF版に落として、WEB版の利用サイトからダウンロードできるようにしてあります。

0. はじめに

0.1 論理学の素養が必要になる理由

この報文の原稿は、学術レポートを作成するときの教材として作成した「**実用文書のまとめ方**」

http://www.nakanihon.co.jp/gijyutsu/Shimada/bunsyo/top.html

の副読本として1991年に私的な教材として編集しておいたものです。実用文書を作ることは技術です。これには三つの要素があります。第一は、正しい文章が書けること、第二が文書の書式、第三が全体としての体裁です。第一の要素には、学問としての**論理学**(logic)の素養が大切です。この学問は、従来、文科系の専門と考えられていました。小学校時代に始まる作文教育では、「思ったことを書きなさい」のような感情移入の文学的な作文で済ませてきました。社会人の教養として必要になる 正しい話し方や作文の素養を埋めておく教育過程が抜けています。欧米の大学教育では、教養課程に **technical writing** を必修としています。これは、多くの言語環境から集まる学生に対して、正しい理解が得られるための話し方と書き方の約束が必要だからです。日本では、世界から見れば狭い閉鎖的な単一言語の環境でしたので、文学的、またはレトリック的な物言いを良い方に評価する傾向を産み、誤解されない話し方教育よりも、敬語教育的な見方がされてきました。理科系の専門では、論理学の教養をそれほど注目してきませんでした。しかし、初等幾何学は、歴史が古く、文章を使う論理学的な方法を利用していました。このこともあって、初等幾何学は、計算を主題とした代数学とは別の扱いがされてきました。コンピュータのプログラミング言語では、論理式が使われます。これらの使い方を理解するためには、理科系・文科系の区別をしないで、古典論理学の基本的な知識を埋めておくことが重要な素養です。

0.2 集合論との関連が問題を複雑にしている

論理学の研究対象は、形を持った物（具象）ではなく、**概念**（考えや判断）です。**思惟**(thinking)とも言います。具体的に扱う対象は、言語で表現された文です。これは物（名詞）ではないので**名辞**の用語が当てられています。文の中身は、何かの判断や断定を表し、これを**命題**と言います。何かの命題を、対立する二つの判断（真・偽）に分ける考え方から出発するのが二値論理学です。近代以降、これに代数学的な方法も応用されるようになりました。代数学は、論理の展開が厳密にできる利点があります。

一方、数学として扱う**集合論**(set theory)は、**逆**に（これも論理学の用語)、代数学的な考え方を言葉で説明するときに、論理学の言葉遣いを利用します。全く同じではなく、微妙に言い替えの別用語や記号体系があるのが混乱の元です。集合論は、一つ二つと数えられる普通**名詞**で表すことができるような、物をモデル化して扱います。これを名辞に代えて**要素**と言います。要素を複数集めたものを**集合**(set)として扱う新しい数学は、**カントール**（Georg Ferdinand Ludwig Philipp Cantor, 1845- 1918）に始まるとされています。集合論は、高校の数学で紹介されるようになりました。この説明の中に論理学の用語が使われます。一つの物でも幾つかの性質を持つとする考え方を含んでいますので、多値論理学の考え方があります。基本的な考え方は、二つの性質に限定した場合で利用します。ここに二値論理学と共通に利用する用語や記号があります。高校までの教育では、論理学そのものを文科系の科目として扱う場面がありません。

0.3 作文指導には論理学の知識を応用する

論理学を具体的に応用する場面は、主に、文章作成のときの、単語の選び方と文の繋がりを合理的に構成するときです。単語を修正する校正と言うよりも、文としての並びを推敲する場面と考えるとよいでしょう。単語と文の繋がりを司るのは、主に接続詞です。その基本的で代表的な用語は「および、かつ、～ならば」の三つです。淡々と客観的な過去や現在の事象を記述する場合は、この三つで済みます。説明、証明、予測などの文では、作者の思惟を丁寧に追加するため、「～ので（順接)、しかし（逆接)」のような文単位を繋ぐ接続詞も使います。後の二つは、省いても意味が誤解されることはありませんし、それが推奨されてもいます。一般社会で利用する作文は、相手が居て、その人に説明して納得してもらい、さらに、何かの作業をして貰うことを目的とします。その人には、日本語の分からない外国人、また、コンピュータも擬人化して含めます。このとき、こちらが考えている日本語の意味を、正しく英語またはプログラミング言語などに翻訳できるように、元の日本語文を注意深く作文しなければなりません。このとき、論理学の素養があることが重要です。最初の段落で説明したように、「実用文書のまとめ方」は一種のマニュアル的な使い方を目的とした編集です。この「易しくない論理学」は、学問としての論理学の、おさらい用の教科書として使うことが目的です。

0.4　古典的な論理学は二値論理学である

論理学は、正しい**思考**（考え方）の形式・法則を研究する学問です。一般論として、学問を扱うときの態度には、「物事には何かの美しい法則があるのだ」という思い込み、前提、想定、仮説、信念、または信仰に近い考え方があります。乱雑に見える部分を除き（**捨象**する）、純粋または本質的な部分を取り出す（**抽象**する）操作をします。用語としての抽象と捨象とは、英語用語ではどちらも abstract です。悪口を言えば、美味しそうなところのつまみ食いです。学者が使う「想定外」の弁明は、都合の悪そうな考え方を捨象したことの言い訳に使っていますので評判が悪いのです。古典的な論理学は、**アリストテレス**（Aristotles、B.C.384-B.C.322）にまで遡る**二値論理学**を基礎に置きます。二値論理学は、欧米の文化的な背景にしっかりと組み込まれていますが、日本の文化には三値論理学的な考え方が多く見られます。多値論理学は、1920 年ころから研究されるようになった新しい体系です。そのため、欧米の論理学を日本に紹介した明治時代の学者の著作では扱われていませんでした。二値論理学には欠点もあるのですが、複雑さを捨象して（真・偽）の二つに分類する**前提**（仮定：premise）で組み立てる方法ですので明快です。多くの重要な考え方も得られています。このこともあって、この教材では、主に、二値論理学について解説を扱い、その上で、三値論理学の説明を補います。

0.5　用語を理解することから始める

論理学の言葉自体、また、論理学に使われている漢字熟語の大部分は、明治初期、**西周**（にし あまね：1829-1897）らが欧米の文献を日本に紹介するときに造語した和製漢語です。和語には**抽象**的な意味を持つ用語が少ないので、漢字の造語能力を生かした熟語が多く工夫されました。用語としての抽象、捨象、具象、具体も理解し難い用語です。西周らは、漢学の素養がありましたので、かなりの数の和製漢語は、漢字本家の中国でも受け入れられる用語になりました。これらの用語は特殊です。それまでの和語には無かった**概念**（考え方）を使いますので、論理学自体も難解な、つまり、易しくない学問の印象（イメージ）があります。論理学は、相手に正しく理解してもらう話し方と、判り易い文章を作文するときとに必要となる実用的な技術を、理論的に支えます。**形式論理学**(formal logic)は、思考の中身を切り離し、記号化して、形式・法則を研究します。記号化を徹底したものが、**記号論理学**(symbolic logic)です。論理的操作を代数学的に行うようにしたものです。記号体系には種々の提案があります。代数学は、コンピュータのプログラミング技術にも応用されています。コンピュータを擬人化し、この人にプログラミング言語を使って話しかけ、理解してもらって仕事をしてもらいます。この分野の一つを**人工知能**(AI; artificial intelligence)と言うようにもなりました。プログラミング言語では、記号論理学の発展として**ブール代数**（George Boole；1815-1864）が使われ、科学技術の分野でも論理学の素養が必要になってきました。古典的な論理学の対象は、日常生活に使う言葉とはやや距離のある文の、**概念**、**判断**、**推論**を扱います。これに対して、社会で使われている文の論理を扱うことを**非形式論理学**と言うことがあります。人間社会では、相手に言葉で説明し、説得し、理解してもらうために、間違った言い方も見られます。その一つが**修辞学（レトリック；**rhetoric**）**です。こちらは、論理学の対立語に位置づけますが、正しい論理に対して**虚偽**の扱いとします。詐欺は、意図して悪い方に応用することですが、だます意思が無くても論理的に嘘になる論旨が虚偽です。これは、正しい論理学の**反面教師**(毛沢東の造語と言う)的な意義がありますので、虚偽の知識も重要です。

0.6　英語の用語が増えたこと

古典論理学の歴史が古いこともあって、欧米の論理学用語は、ラテン語起源のものが多く見られます。コンピュータ技術はアメリカ主導型で開発され、進歩してきましたので、論理学の用語にも英語が多く使われるようになりました。例えば、連言、選言、否定、真、偽の用語に代えて、英字のまま And, Or, Not, True, False を使うことがそうです。論理学を勉強するとき、以前から使われてきた漢字の熟語と、英語の用語とが、混在するようになりましたので、初心者はその意味を理解することが最初の難関です。それに加えて、特殊な論理記号も見られます。これらの用語と記号の意味は、本文中で解説しますし、索引からその場所を探すこともできます。しかし、主要な用語や記号などは、最初に説明しておく方が理解に便利でしょう。この報文の原稿は、教材を目的として 1991 年に作成したものを元に、インターネット時代を考えて 2012 年に改訂しました。以前の教材では、用語の定義などを付録とし、本文ページの後ろの方にまとめました。科学技術関係のマニュアルでは、用語や記号の定義を本文ページ構成の最初に置くことも多くなりましたので、この報文では第 1 章として独立させることにしました。

0.7　論理学の前に言語学があること

日本語は、英語の環境からみれば、曖昧さが多いとされています。これは日本語が不完全な言語であるのではなく、言葉の使い方が不適切であることの方に罪があります。世界から見れば、日本は、方言違いを別にすれば、一つの言語だけが使われていますので、他の言語環境の人との間で誤解を生じないような表現方法に無頓着なところがあります。狭い村単位の地域ならば、言葉を省いても、また、そこだけで通用する言葉(隠語など)を使っても不便を感じません。多くの地域から人が集まる大都市では、結果的に誤解を防ぐ標準化が育ちます。国家的にこれを進めることが、日本ならば国語教育です。これは、母語を日本語とする日本人を対象に考えられてきました。ところが、海外からの外国人留学生や、帰国子女に改めて日本語を教えようとなると、合理的な教育技術が育っていないことが問題になってきました。英語では、個人単位の家庭教師であっても、程々の教育技術を理解しています。これは、論理学の問題よりも前の、言語学の課題です。その実践的な手段が翻訳技法です。最も単純な英文和訳の方法は逐語訳です。英単語単位で日本語の訳語を当てます。対応する言葉がないときには造語が必要です。それに続けて、単語並びの文に助詞（がのにを）を加え、日本語の語順に変えます（例えばSVO→SOV)。この技法は、漢文訓読法でレ点や返り点を付ける方法と同質です。さらに、日本語の動詞は活用形を選びます。日常言語とはズレがありますが、意味は正確に理解できます。なぜ正確になるのかの理由を支えている骨格が、実は論理学にあります。英文和訳は、擬人化したコンピュータを介する自動翻訳（または機械翻訳）が実用的なレベルに達しています。しかし、逆の和文英訳は、みじめな成果しか得られていません。その理由は、元の日本語が論理学的に見て不完全な骨格になっていることの罪が大きいからです。したがって、作文をする本人自身が、論理学を踏まえて努力してもらうことが必要です。

0.8　英語と日本語との異同を理解しておくこと

明治以降、欧米の学問を日本で学ぶためには、欧米語で書かれた文書を日本語の文書表現に直す翻訳が必要です。このとき、言語構造の違いによる理解の過程に混乱が生じます。表音文字のアルファベットで表記された文を、意味をそのまま伝えるように翻訳するとき、表意文字である漢字を利用することは理解に役立ちます。元の語をカタカナ語で使うことは、原語の知識があれば何とかなりますが、原則として評判は良くありません。欧米語では表意文字がありませんので、表記を短くするため、省略表記、頭字語、または記号に代えますが、その定義や紹介を別に造ります。それらを声に出して読むことが普通に行われ、その場合には省略前のスペルで言います。例えば「a=b+c」の代数式は、「a is equal to b plus c」の言い方を記号化して表したものです。日本語では「a はｂ足すｃに等しい」と読むのですが、「b 足すｃは、a です」の逆の言い方が普通ですし、電卓を使う計算も「b ＋ c ＝ a」の順でキーを打ちます。これは、主語(S)・述語(V)・目的語(O)の語順の約束と関係しています。英語は、文字の並びと処理の流れが逆になることがあって、コンピュータに処理を指令するコンピュータ言語は、コンパイラが実計算の処理の流れに合わせるように翻訳します。論理学も集合論も、数式表現をすることで合理的な研究ができる利点があります。しかし、実社会で作文や話し方に応用するときは、文章表現です。

0.9　名詞と動詞の機能を理解しておくことが基本

文構成の最小単位は、主語と述語の対です。主語は名詞が普通です。英語の環境では、名詞の単複を神経質に区別し、単数扱いをする集合名詞もあります。これが動詞の活用形にも影響します。標準的な動詞は、英語の環境では be 動詞とそれ以外の一般動詞との二種類に大別します。これと対応する日本語の言い方が、文体としての「です・ます調」です。「です」が be 動詞に当たります。ただし形容詞の終止形に使うときに幾らか困ります。少し硬い文は「である調」一つです。こちらは be 動詞も含め、動詞一般の終止形に使い、形容詞は終止形をそのまま使います。「猫は動物です」「猫は動物である」は be 動詞の形と考えることができます。ここで、猫は一般名詞ですが、英語の環境では cat だけでなく、単複を区別し、冠詞を含めて使い分けます。動物は、英語の普通名詞 animal が当たります。意義的には集合名詞の性格も持ちます。したがって、語順を入れ換えた「動物は猫です」は誤りになるのです。「或る（数の）動物は猫です」は正しいのですが、名詞を入れ換えた「或る（数の）猫は動物です」の言い方は誤りです。集合論ではこれを明確に区別し、論理学がそれを理論的に支えます。文学的な作文は、意図して論理学的に誤った言い方も使いますし、それを容認することもしますが、実用文書では字面（じずら）の裏の意味を読者側が理解することを期待しないように作文をします。この読者側にコンピュータを含める時代になりました。このことを考えて、この報文の最後の章構成に、論理学の応用をまとめました。

1.　用語の定義と解説

1.1　一般的な論理学の用語

1.1.0　学問と技術とを区別して理解する

学問とは、平たく言えば、勉強(study)することです。その基本的な方法は、過去に得られた知識を理解しておくことです。そのためには、まず言葉を覚えます。特に海外の学問を理解するには、そこで使われている言語を覚え、そして専門用語の意味と定義とを自分の言語で理解することから始めます。双方の言語環境で同じ物があるときは、単純に言葉の対応辞書(dictionary)を作ることができます。例えば「犬：dog」とすれば済みます。しかし、外国語にあって日本語に無い言葉の場合には、言葉自体を作成することが必要になります。それがどういう物か事かの意味を説明する辞書が必要です。こちらは、英語では glossary が当たります。勉強は受動的な態度です。外から見れば、何もしません。

一方、**技術**は、視点が**未来**にあります。これから何かを創造する行動に移すため、意思決定が必要です。無難な態度は、過去にあった例を踏襲します。これを保守的と言いますが、悪い意味だけではありません。前例のない、また経験のない事柄に挑戦するときは、**研究**(research)の態度を取ります。進歩的であると尊敬されることもあるのですが、失敗の危険も考えておきます。失敗の場合を想定して、事前に、**安全**(safety)とそのための**救済**(security)の**対策**も考えます。この全体が**危機管理**(risk management)であって、大人の態度です。社会組織の中で、前例の無いことに取り組みたい事態が起こったとき、賛成または反対の考え方を、相手に納得が行くように言葉で説明する説得の技術が必要です。この技術を学問的に支えるときに、論理学の知識が役に立つでしょう。

論理学の用語の大部分は、通常の文章に出てくることのない学術用語です。この章の用語の並べ方は、主題ごとにグループ化しました。印刷物にまとめるときは、全体を章・節・項に分けた編集をし、項目の索引は本文のページ番号を使います。しかし、インターネットを介した電子出版の時代には向きません。本文をパソコンの狭いモニタに表示して閲覧します。全体を通した索引は、目次構成に使った章・節・項を示す三つの整数を小数点で繋ぐ番号を使うことにしました。

1.1.1　論理学（ロンリガク）：logic

広辞苑には、明治初年、西周の訳語とあります。**哲学**の哲も意義としては考えであって、これも西周の造語とされています。論理学の説明は、「**0.5　用語を理解することから始める**」の節を参照して下さい。主観的な判断である感覚や感情を表す文学的な文は、論理学の対象とはしません。本人が思っていること、また感覚的に感じていることを客観的または学問的に扱うことはできません。「他人は本人と全く同じように思いや感覚を再現できない」からです。

1.1.2　二値論理学：two-valued logic

論理学は、二つの概念を対立させて扱います。この二つを、(ホントとウソ)；(真、偽)；(正、誤)；(true, false)；(T,F)；(肯定、否定)；(はい、いいえ)；(yes, no)；さらに、数字の組(1,0) で代表して扱うことをします。「白、黒」などは、その中間に幾つもの段階の灰色が考えられますので、二つの対立概念（考え方）として扱うことができません。間違えやすいことは、(正、誤）の組と（肯定、否定）の組との区別です。例えば、肯定文が誤りで、否定文が正しいことがあるからです。命題の正誤の判断を記号に置き換えるとき、命題の文型の肯定・否定を(1,0) で扱い、論理的正誤にも(1,0) を使うと、混乱を生じます。なお、日本語では、論理的な正誤の判断は「はい、いいえ」で、肯定・否定を（である、でない）で使い分けます。二つの対立概念に分けるとき、そもそも、その二つの概念が互いに相手の補完になっていなければ論理が成り立ちません。この間違いの最近の深刻な例は、原子力発電所の**安全**と**危険**の判断を論議するときに見られます。「安全な状態である・危険な状態である」は、未だ現実に事故や災害が発生していないが、その恐れがあることを判断するときの思惟を言います。安全と言う判断は、危険があることを理解した上での**言い換え**になっていますから、実は、安全と危険とは同じ概念であって、対立概念ではないのです。第 1.3 節に列挙した虚偽の分類で言えば、「1.3.12　原因まちがえの虚偽」に入れます。似ている用語の対である**安定**と**不安定**とは、物理的に定義できる対立概念です。ただし、安定であっても、判断としては安全とは言えない場合もあります（この論議は「土木工学と安全」http://www.nakanihon.co.jp/gijyutsu/Shimada/SafetyEngg/Index.html　にまとめました)。

1.1.3 名辞（メイジ）：term

名辞とは、論理学が扱う基本材料であって、概念（考え）を言葉（文；文字並び：主語＋述語）で表したものです。文法上、名詞の説明と考えるのが分かりやすいのですが、種類や性質などを表す抽象名詞や、述語を形容詞とする場合も含みます。普通名詞を利用する名辞を一般名辞、または共通名辞と言います。形式論理学では、名辞を基本単位とし、名辞の関係に分解して扱います。

1.1.4 命題（メイダイ）：proposition

命題とは、我々の判断・断定・意見(mind)を述べた**文**です。判断は、いくつかの観念の組み合わせからなっています。命題は、例えば「この花は赤い」のように、客観的な**事実**(fact)を表した断定・意見の文です。そしてこの判断を受け入れるべきか、拒絶するかという選択の可能性がついてまわります。これを論理学においては、「命題とは真または偽である文である」と言います。アリストテレスは、命題を「真と偽を語ることが出来る文」であると定義しました。この真または偽という二つの値のどちらかを必ず取るという前提を、**二値原理**(principle of bivalence)と言います。いつもそうなるのではない場合を扱うことが**多値論理学**の議論です。

文は、主語・述語に大別する構造を持ちます。文の真偽だけに注目して命題の繋がりを扱う論理学を**命題論理学**と言います。幾つかの命題が接続詞（かつ、または、ならば）で繋いで複合してできた命題を**複合命題**と言い、そのときの構成要素となった命題を**要素命題**と言います。ある命題の否定形を**否定命題**と言います。二つの命題の接続が**選言命題**（ＰまたはＱ）の場合、Ｐ，Ｑのことを**選言肢**と言います。**連言命題**（ＰかつＱ）では**連言肢**と言います。二つの命題が「(もし)ＰならばＱ」のように結合された命題を**仮言命題**または**含意命題**（**条件命題、**英語では IF-THEN の構文）と言い、この場合のＰを**前件**、Ｑを**後件**と言います(仮定文も参照)。「すべての」で始まる命題を**全称命題**(universal proposition)、「或る（ある)」で始まる命題を**特称命題**、「である」で終わる命題を**肯定命題**、「ではない」で終わる命題が否定命題です。この分類は集合論で考えると明快です。論理学では、この４通りの組み合わせはそれぞれＡ，Ｅ，Ｉ，Ｏの記号で表します。これらをまとめて、**定言命題**と呼びます。

Ａ；　全称肯定命題、「すべての　ｐは、ｑである」
Ｅ；　全称否定命題、「すべての　ｐは、ｑではない」
Ｉ；　特称肯定命題、「或る　　ｐは、ｑである」
Ｏ；　特称否定命題、「或る　　ｐは、ｑではない」

Ａ，Ｅ，Ｉ，Ｏの主語ｐは，普通名詞です。固有名詞は単数扱いをしますので、これを主語とする命題のことを単称命題と言います。単称命題は、全称命題（ＡとＥ）の特殊な場合と考えます。ここで、「すべて」と「或る」の言い方は、集合論的であることに注意します。古典的な論理学では、集合論が未発達であったことを理解しておきます。

1.1.5 三値論理学（three-valued Logic)

三値論理学は、何かの事象を三つの概念に分け、それらを対立させて扱います。この**分け方**の方法を、筆者は**三分法**と言うことにしました。日本語の環境ではよく見られます。例えば、上中下、左中右、勝負ごとでは、勝ち・負けに、あいこまたは引き分けを加えます。じゃんけんは典型的な三分法です。何かの意思表示をするとき、賛成・反対に分けるのが二分法ですが、日本では保留の態度も普通です。左翼運動が盛んであったころは**日和見**（ひよりみ）の用語が見られました。これらは欧米流の判断法から見れば、曖昧な態度であると非難されます。しかし、解決方法に妥協の提案ができる意義がありますので よい意思表示法だと思います。しかし、意思決定の段階では、結果的に二値論理に帰着します。数学的な条件に応用するとき、数の（正・負）にゼロ(0)の場合を加えた三分法もそうです。三つの概念に分けるとき、二分法に比べて判断が曖昧になりますので、理論にこだわる人は嫌います。例えば、数を扱うとき、理論上の実数だけを考えるときは 0 を加える必要がありません。しかし、数を実用的に使うときは、実数であっても、或る桁数のところで表示を打ち切りますので、実質は整数化しています。数値計算は、実際には整数で計算をしています。数の並びを大小順で二つに分けるとき、境界の決め方も悩ましいところがあります。これは、**閾値**（しきいち：threshold）を決めると言う実用的な処理と関係します。数の場合には、切り上げ、切り捨て、四捨五入のどれを選ぶか、が一例です。一般的なプログラミング言語では、論理的な判断をするとき、二分法の(if～then～else)を使います。しかし数値計算用プログラミング言語 FORTRAN の古いバージョンでは、算術条件文と言うのがあって、if(A) で、A の値の（負・ゼロ・正）で判断をして三つの分岐先が選択できました。しかし、単純な二分岐条件文を応用することで解決できますので、この条件文は廃止されました。

1.1.6　推論：reasoning

与えられた前提から、未知の結論を導きだして行く過程のことです。命題から命題への推移により、未知の結論を発見していく過程です。**推理**と**推論**との使い方の区別は、はっきりしていません。推理は、発言者の主観的な思いや意見が入る意味を持ちます。推理小説は、特に犯罪の方法・動機などにまつわる謎を論理的に推理して解明する、作者の文章の進め方に読者が興味を持ちます。論理パズルも、同じように、論理の過程を楽しむ作品です。人工知能の研究分野では、推論を使います。

1.1.7　条件文と仮定文

論理学では、文法で言う**仮定文**(If sentence)を扱いません。これは文学的な言い回しだからです。「もし(If)」で始まる過去形の文は、現在はそうでないことを言う文学的な言い換えです。論理学的には**条件文**(conditional sentence)の用語を使い、「もし」を省き、動詞も現在形を使い「～ならば」で次の文に繋ぎます。現在の時点では「肯定・否定」または選択肢が決まっていなくて、この後（未来）でそれが決定される文です。コンピュータのプログラミング言語では(if ～then ～else)の書き方をしますので、英文法の仮定法の使い方と混同します。つまり、条件文と仮定文とは全く別の判断を言います。日本語では、通称で「れば・たら」として出てきます。この正しい使い方と意味の区別ができるでしょうか？　例として動詞の「在ったら・在れば」を考えれば判ります。「たら」は過去形ですので、「在ったら」と言うのは、実は無かったことの言い替えです。したがって、言い方としては「無かったので」を原因とした結果を述べた論旨です。実質は条件文でも仮定文でもありません。「在れば」は現在形であって、現在は無いことを意味し、未来では「在るか無いか」が決まっていません。日本語の文語動詞の活用形に、**未然形**と**已然形**があります。未然とは、「いまだしからず」と訓読みし、これから動作を起こす状態を意味します。已然形は「すでにそうなった」の過去を意味します。口語文法の用語では、これが仮定形の分類になりました。仮定を言う文は、過去の事象がそうでなかったことを言うからです。

ブール代数を応用するコンピュータ言語では、二値論理学を扱いますので、２分岐の条件文に(if ～then ～else)を使います。多値論理学な選択肢を実現させるときは、２分岐を連ねるときに ElseIf 文で繋ぐか、(select ～case)のような並列の選択肢を使います。日常の日本語では、「～する**場合**」「～する**とき**」が条件文です。なお、「場合」は大きい条件、「とき」は狭い条件と使い分けます。

1.1.8　演繹（エンエキ）：deduction

与えられた事実(facts) 、または前提 から出発して、必ずそうなる結論を引き出す方法です。幾何学の証明方法には、幾つかの知られた公式や法則を使って正しい結論が得られることを論証しますが、この方法が演繹です。演繹的方法の代表的なものに**三段論法**(syllogism) があるので、演繹＝三段論法と考える人も多くいます。或る結論が既に与えられていて、その結論を導きだす途中の過程や証拠を探しだす方法ですので、**後ろ向き推論**(backward reasoning)、**後ろ向き連鎖**(backward chaining) の用語があります。演繹法との対比で、**トップダウン探索／推論**(top-down search/reasoning) の用語もあります。

1.1.9　帰納（キノウ）：induction

演繹（エンエキ）と対にして使われる推論法です。個別の事柄から出発して、一般的な法則や結論を導く方法です。もし例外が見つかれば、この結論は正しくありません。**完全帰納法**（完全枚挙の帰納法）とは、実例をすべて挙げて一般的な法則を導く方法です。「ａ１　ならばＰである」、「ａ２　ならばＰである」などをすべて積み上げていって、「Ａ＝（ａ１　，ａ２　…）ならばＰである」のように発見的に法則を導きます。幾つかの例を学習して法則を導くことは、帰納的推論であるので learning from examples と言います。ＡからＰへと言う流れは「ＡならばＰである」または「ＡはＰである」と言う論理の向きでみると順方向ですので、**前向き推論**(forward reasoning) 、**前向き連鎖**(forward chaining)と言う用語がＡＩの方で使われます。また、**ボトムアップ推論**(bottom-up reasoning) や、**ボトムアップ探索**(bottom-up search) の用語があります。個別のものを下層に置き、積み上げて法則を導くという構造を念頭においた用語です。

1.1.10　証明：proof

或る物事または判断の真偽を定める根拠を示すことです。与えられた前提と結論とが、正しく論理法則によってつながることを確認する過程を言います。

1.2　論理演算に関係する用語

1.2.0　基本的な演算の約束

算術の基本的な演算は四則演算であって、用語としては加減乗除を使います。和語的に言えば、「足す・引く・掛ける・割る」です。この処理を記号で表すとき、日本語の環境では「＋、－、×、÷」を当てますが、英語表記では「＋、－、＊、／」と使います。演算の約束を代数式で書くとき、書き順と関係した演算の約束があります。二つの数を考えた加算と乗算は交換法則が成り立ち、減算と除算は成り立ちません。なお、0による除算はありませんが、エラーとはしない約束を決めることがあります。

二値論理学では、二つの命題間の擬似的な演算の用語として、**選言**、**連言**、**内含**があります。**否定**は、単独に使います。これらの詳しい説明は後の章で補いますが、交換法則が成り立つものと、そうでないものとがあります。集合論でも論理演算が使われるのですが、こちらでは全体と部分のような量的な考えを扱いますので、代数学的な加法と減法も使います。また、日本語ワープロでは特殊な演算子記号を使うことができますが、英字専用のテキストエディタでは表示が出来ない場合があります。

1.2.1　肯定（コウテイ）：affirmation

或る判断を与える主語と述語の関係をそのまま認める、または、それが妥当であると認めること。

1.2.2　否定（ヒテイ）：negation

或る判断を与える主語と述語の関係を拒否すること、または、打ち消すこと。「二重否定は肯定と同じ」という論理法則を「**二重否定律**」と言います。「敵の敵は味方」と言う論理は、二値論理学の法則です。三値論理学では必ずしも成立しない論理です。中立もあるからです。

1.2.3　選言（センゲン）：disjunction

論理的演算の一つであって、日常言語では「または(or)」に相当します。排他的な選言(exclusive or; XOR)と、非排他的な選言(通常の or)とがあります。Ａ，Ｂの二つを組みにする場合、非排他的な選言は、「ＡまたはＢ，またはその両方」と言います。英語では「A or B or both」を正確な言い方とし「A and/or B」の表記を使いません。算術和に似た性質がありますので、**論理和**(logical sum) とも言います。排他的選言は、「ＡまたはＢのどちらか」を選択するときを言います。ＡとＢとが同じであるときは、どちらも選択しません。これは、理解の難しい判断です。

1.2.4　連言（レンゲン）：conjunction

論理的演算の一つであって、日常言語では「かつ(and) 」に相当します。集合論で扱う連言は、算術積に似た性質がありますので、**論理積**(logical product) とも呼ばれます。これは、二つの集合の共通部分を取り出します。したがって、共通部分が無い場合は、代数学の積の演算と同じように0相当の結果になりますが、これを**空集合**と約束します。

1.2.5　内含（ナイガン）：implication, inclusion

含意、包含とも言います。「ＰならばＱである」と言う複合命題の「ならば」の関係を言います。この文を条件文と言い、英語の[if P then Q]に対応させます。Ｐが**前件**、Ｑが**後件**です。「ＰならばＱである」とき、ＱであるためにはＰでありさえすれば十分ですので、ＰはＱの**十分条件**(sufficient condition)と言い、ＱはＰの**必要条件**(necessary condition) であると言います。また、ＰとＱとを入れ替えてできる形を、もとの内含に対して、**逆**向きの内含と言います。集合論では、「ＰがＱに含まれる」の関係を**帰属(inclusion)**、「ＱがＰを含む」が**包摂(subsumption)**です。帰属はＰを部分的に、包摂は全部がＱに含まれます。記号文字の向きは、左右どちらも使いますので混乱しないようにします。

1.2.6　トートロジー：tautology

通常は**同語反復**を意味します。例えば「猫は猫である」のような表現になることを言います。長い論理式でも結果が常に真になるものはやはりトートロジーですが、この場合には**恒真式**（コウシンシキ）：valid formula とも呼ばれます。論理法則として知られているものには、恒真式が多くあります。

1.2.7　矛盾（ムジュン）：contradiction

一般的には、ある命題とその否定命題とを同時に主張することです。論理式で結果が常に偽になるときを矛盾と言います。例えば、「Ｐであり、かつＰでない」という文は矛盾です。これに対する「Ｐであるか、またはＰでない」と言うのが恒真式であって、こちらはトートロジーです。

1.2.8　対偶（タイグウ）：contraposition

「ＰならばＱである」に対して、否定形を使った［Ｑでないならば Ｐでない］を対偶と言います。これを対偶の法則、または**対偶律**(law of contraposition) と言います。

1.2.9　逆（ギャク）：converse

逆は、正確には、関係の逆、または逆関係と言います。「ＰならばＱであるに対して、ＰＱを入れ換えた「ＱならばＰである」を逆と言います。裏（ウラ）も参照のこと。

1.2.10　裏（ウラ）：reverse

裏とは、「ＰならばＱである」に対して、否定形に直して「Ｐでないならばでない」の関係のことを言います。逆と裏との定義を使い分けている研究者と、裏も広義に逆に含めて使っている研究者とがあります。

1.2.11　順（ジュン）

逆・裏・対偶の命題に対して、元の肯定命題を順ということがあります。

1.2.12　対当（タイトウ）：opposition

正しくは対当関係と言います。定言命題**Ａ，Ｅ，Ｉ，Ｏ**（命題の説明を参照のこと）の間に成り立つ形式的関係を言い、この組み合わせを、さらに４種に分類します。

ＡとＯ、ＥとＩの関係	矛盾対当
ＡとＩ、ＥとＯの関係	大小対当
ＡとＥの関係	反対対当
ＩとＰの関係	小反対対当

なお、**対立**という用語もあり、言葉としてはなじみやすいでしょう。英語は同じ opposition です。

1.2.13　逆説（ギャクセツ）：paradox

逆理とも言いますが、**パラドックス**(paradox)の用語の方がよく知られています。哲学史上の有名なパラドックスが、「アキレスは亀を追い越せない」です。論理学の方では矛盾を含む主張のことです。「私は私であって、かつ私でない」のような主張です。文学などではレトリック的な技巧として使うことがあります。また、逆説と似たものに、**二律背反**（**アンチノミー**：antinomy）があります。同一の命題から二つの互いに二つの相反する（矛盾する）命題が正当に引き出されることを言います。これに対するのが**排中律(law of excluded middle)** と言い、同一の命題は真であると同時に偽であることはできない、と言うものです。英語の用語からきた**トレードオフ**（trade-off）は、一方を追求すれば他方を犠牲にせざるを得ないという二律背反の状態・関係のことです。平重盛がつぶやいたという「忠ならんと欲すれば孝ならず、孝ならんと欲すれば忠ならず」も**二律背反**の心情を述べた例によく引用されます。シェークスピアのハムレットが自問する「To be, or not to be -- that is the question」も有名です。二者択一の四字熟語があります。現実社会では、意思決定が必要になるときの難しさを言います。この課題は、現在から未来を向いた視点が必要なときに起こります。これは、二値論理学で解決することができない問題ですし、二値論理学の欠点とも言えるでしょう。この状態の実用的な解決は、**妥協**ですが、これを認めると三値論理学になります。

1.2.14　ベン図：Venn's diagrams

部分集合、結び、交わりなどの集合の関係を、直観的理解を助けるために、円の組みあわせて表した図を言います。イギリスの論理学者 John Venn(1834-1923) の名で呼ばれます。スイスの数学者オイラー(Leonhardt Euler, 1707-83)が用いたので、オイラーの図とも言うことがあります。もともとは集合論の説明に使うのですが、論理演算を視覚的に理解するときにも利用します。

1.2.15　ド・モルガンの法則

イギリスの数学者で論理学者 Augustus De Morgan(1806-1871) の名前を冠した論理式です。これは、否定を含む論理演算で、結果が同じになる言い換え表現の法則です。

1.3 虚偽に関する用語

1.3.0 虚偽(fallacy) の定義

普通の言葉で言えば、「誤り」「間違い」「正しくない」ことです。論理学的には、虚偽とは、「正しくない推理、論証」、「正しいとみせかけているが、実は正しくない論証」、「正しいモノとして通用している、にもかかわらず、間違っている論証」のことを言います。一方、嘘（ウソ）は、「本当でない」ことであり、「わざとした誤り」の意味で使いますが、無意識的な場合にも使われるようになってきていて、真(true)対する偽の意味にも使われています。

1.3.1 一般化：generalization

概括とも言います。限られた少数の例にみられるコトガラを、その例と同種類のコトガラについておしひろげて主張すること。**帰納的推論**と方法が似ているので、「**帰納による概括**」とも言います。例えば、「いまどきの学生は…」、のような表現や、「アメリカではすべて…」の類いの主張です。

1.3.2 威力に訴える論証

ラテン語に argumentum ad baculum の用語があります。自己の威力または暴力を示して、相手を屈伏させる論証。いわゆる「勝てば官軍」「横紙破り」式の論証。

1.3.3 演繹的虚偽：(deductive fallacy)

帰納的虚偽(inductive fallacy) 、**形式的虚偽**、**論証的虚偽**などの用語と対比しながら使いますが、いずれも演繹推理や帰納推理の適用の際に起こり得る虚偽の総称です。

1.3.4 エンチュメーム：enthymeme

省略三段論法の一種。ありそうなこと、単なる推測、をはっきり出さずに自明のごとくよそおって、断定的結論を引き出す修辞的な三段論法。

1.3.5 換位の虚偽：(fallacy of conversion)

換位または**換位法**とは、直接推理法の一つであって、命題の質（肯定・否定）と正しさ（真理値）をかえずに主語と述語とをいれかえることを言います。換位ができる場合と、できない場合、またできても制限をともなう場合があります。この適用を誤ったときに生ずるのが換位の虚偽です。例えば、「猫は動物である」に対する「動物は猫である」が誤りです。この場合は「或る動物は猫である」ならば正しい文になります。この論理は、集合論を考えると、正誤の判断を理解できます。

1.3.6 換質の虚偽：(fallacy of obverion)

換質または**換質法**とは、直接推理法の一つであって、命題の意味をかえずに、肯定命題を否定命題に、または否定命題を肯定命題に換えることを言います。否定をつくるとき、別の概念を使うと換質の虚偽になります。例えば、「６は偶数である」に対して「６は奇数ではない」は正しいのですが、「この花は白くない」の換質に「この花は黒い」というのは誤りになります。

1.3.7 感情移入：empathy

相手の身になって、「…に違いない」と思う。これが感情移入です。人間相手だけでなく、動物やモノも対象になります。文学や美術などではよく用いられますが、論証に使うのは誤りです。西洋のことわざに、「人の心は悪魔にも分からない」というのがあるそうです。別に、「**感情に訴える虚偽**」とも言います。

1.3.8 形而上学的（ケイシジョウガク：metaphysical）

形而上とは、難解な哲学用語です。広辞苑によれば、井上哲次郎(1856-1944)の訳語とされています。種々の言葉が当てられています。哲学的、抽象的、超経験的、なにか神秘的な、直接感覚にたよれない、などの意味で使われます。その言葉の意味が他人に分かりにくいので、「独断的」と言い換えることができます。宗教での、悟りの状態の理解が一例です。対立して形而下の用語もあります。こちらは具体的な(physical)形があって理解できるものを言います。

1.3.9 結合の虚偽と分割の虚偽：fallacy of composition, fallacy of division

前提において別々に考えられていたコトガラを、結論で一まとめにすることを結合、また、その逆に、一まとめで考えられていたことを別々に主張することを分割と言い、それぞれの場合に生じる虚偽のことです。一般概念と集合概念を混同するときに起きます。例えば、「家族」は集合概念で、「男」「女」は一般概念です。夫婦は家族の最小単位ですが、男と女との問題と一緒にして論述、またその逆の論述などに虚偽が起こることがあります。

1.3.10 結論まちがえ：heterozetesis

求める結論ではない他の結論を引っ張り出して、論証が終わったようなフリをする誤り。

1.3.11 権威を利用する論証

「感情に訴える虚偽」の一つであって、「尊崇の情に訴える論証（ラテン語 argumentum ad verecundiam）とも言われます。ゲーテが言った、偉い先生がこう言った、古い諺などをひけらかして引用をする論法です。科学論文では、引用に当たっては、必ず出典を参考文献として上げなければなりません。また、名前に敬称をつけることもしません。ただし男女の区別や、職業や地位を論文の中でどのように扱うかは微妙な問題です。敬語を使うのは、権威に対する、へつらいの論述です。

1.3.12 原因まちがえの虚偽：fallacy of the false cause

ある事物の原因でないものを、原因だ、主張する誤り。これには大体三つあります。(a) 時間の前後関係を、因果関係にすりかえるまちがい。(b) ある事態に伴うものを、事態の原因だと思うまちがい。(c) そのことと関係ない外的な条件を原因と考えるまちがい。

1.3.13 言葉争い：logomachy

或る言葉はモノを表す記号と考えます。言葉によって示されるモノを全然考えに入れずに、言葉や話の上だけで言い争う議論のこと。神や仏の存在論争がそうです。

1.3.14 四個名辞の虚偽：fallacy of four terms

四個概念の虚偽とも言います。三段論法で用いる概念は三つに限ります。これより多いときには三段論法にならず、多ければまちがいを生じます。この誤りは、同じ名辞を前後の命題で異なった意味に用いるとき、ミカケは三つでも四つの概念になるとき、などに生じます。例えば、「彼は警察の犬である、犬は臭覚がするどい、だから彼は臭覚がするどい」というとき、「犬」の概念が同じではありません。

1.3.15 自然主義的虚偽：the naturalistic fallacy

非倫理的な前提から、倫理的結論を引き出してくる手続き。自然主義の倫理学でしばしば用いられますが、例えば、極端な動物愛護の精神などがあります。

1.3.16 衆人に訴える論証

ラテン語の arugumentum ad populum 。いわゆる雄弁術です。

1.3.17 重要性不当強調の虚偽：fallacy of illicit importance

「この問題は自明のことだ、だから重要だ」と主張するような論法。

1.3.18 独我論：solipsism

経験主義または主観主義を論理的に進めると、「天上天下我ひとり」の考えに陥ります。天動説を主張するような、自己中心的な論理ですが、科学的・客観的な見通しをつけ加えることで、このあやまちから救われます。

1.3.19 ニセ命題：pseudo proposition

ニセ言明とも言います。例えば、「月が光っている」は意味のある命題ですが、「月が泣いている」は感情移入ですので、ニセ命題と言います。ただし文学的な表現ではあります。

1.3.20 背論理：paralogism

不注意より生じた まちがっている論理。

1.3.21　**発生論的虚偽**：genetic fallacy
　或るものの起源と機能とは別のものであることを忘れて「起源の劣等性から、すぐにそのものの機能の劣等性を断定する論法」。

1.3.22　**非整合性**：inconsistency
　理論の中において、互いに相反する（矛盾する）命題を含んでいること。

1.3.23　**的はずれの虚偽**：ラ ignoratio elenchi
　論点相違の虚偽とも言います。求める結論とは関係のない結論を証明する論証。

1.3.24　**無知に乗ずる論証**：ラ argumentum ad ignorantiam
　相手の知らないことにつけこむ論証。

1.3.25　**憐愍に訴える論証**：ラ arugumentum ad misericordiam
　いわゆる「泣き落とし」。

1.3.26　**論点無視の虚偽**：fallacy of ignoring the issue
　論点相違の虚偽とも言います。

1.3.27　**前件否定の誤りと後件肯定の誤り**
　「ＡならばＢである」の内含を表す条件命題において、この全体命題が「偽」になるのは、前件Ａが「真」で、後件Ｂが「偽」になるときに限ります。

　一方、後件Ｂが肯定、つまり「真」であるとき、前件Ａが真偽いずれの場合にも条件命題が論理代数的に「真」となります。これが後件肯定の誤りです。例えば、「二つの三角形が合同ならば、それらは相似である」という命題において、「それらは合同でない、よってそれらは相似でない」という推論は、前件否定の誤りです。また「それらは相似である、よって合同である」という推論は、後件肯定の誤りです。なお、「ＡならばＢである」の対偶は「ＢでないならばＡでない」であるので、「二つの三角形が相似でないならば、それらは合同ではない」は正しい推論です。

1.4　人工知能に関する英語の用語

1.4.1　artificial intelligence(人工知能；ＡＩ)

人間の頭脳の働きをコンピュータがどのようにして実現させるか、を研究対象とする学問。頭脳の働きとは、認識(recoginition)、思考(thinking)、判断(making decisions)、推論(inference or ressoning)、問題解決(solving problems)、学習(learning, storing and receiving knowledge)などを言います。

1.4.2　axiom(自明の理、公理)

1.4.3　backward chaining / reasoning (後ろ向き連鎖／推論)、top-down search/reasoning (トップダウン探索／推論)

目標(goal)あるいは予測(expectation)から出発して、それを支持する証拠を探してプロダクションルールや作用素を適用しながら後ろ向きに進める推論。

1.4.4　breadth-first search(幅優先探索、横型探索)

盲目的探索の一つ。深さ優先探索と対比して言います。

1.4.5　cognition, cognitive science(認知すること、認知科学)

1.4.6　conclusion (結論)

論理的な探索の最終点

1.4.7　condition (条件)

プロダクションルールの If 部を指します。

1.4.8　consultation (相談)

エキスパートシステムの利用方法の一つの形態

1.4.9　data-driven (データ駆動)

要求駆動(demand-driven)と対比する用語。ツリー型のデータ探索構造を使って行う、前向き推論の方法。

1.4.10　deduction, deductive reasoning (演繹、演繹法)

与えられた事実(facts)または前提(premise)から出発して結論に至る推論の進め方。これは後ろ向き推論です。

1.4.11　depth-first search (深さ優先探索)

盲目的探索(blind search)の一つ。探索木の、どの枝を探索するかの順番の決め方で、横並びの同じ行を先に探索する横型探索に対比して、深い方を探す方を優先する方法。

1.4.12　examples, learning from examples (例、例からの学習)

帰納推論の一種。システムの外から、学ぶべき概念の例あるいは反例を与え、このマトリックス構造からプロダクションルールを導いて推論に用います。

1.4.13　expert system(エキスパートシステム)

人工知能（ＡＩ）の主題を構成するコンピュータプログラムとデータなどの全体です。

1.4.14　expertise

エキスパート（専門家）の持っている知識を(heuristics と facts)の形で構築すること。

1.4.15　explanation, explanation-based learning (説明に基づく学習)

エキスパートシステムにおいては、或る結論に至る道筋を示して、ユーザが納得して理解する様なときに用います。

1.4.16 facts(事実)
一般的には、知られていること。事実と訳し、知識の一種です。

1.4.17 fire (発火)
或る仕様が満足されるなどの条件が満たされた時、決められた処理が動き出すことを言います。

1.4.18 forward chaining/reasoning, bottom-up search/reasoning (前向き連鎖／推論、ボトムアップ探索／推論
入手可能な法則・事実から出発して、求めたい結論や目標を導き出す推論の方法。

1.4.19 goal (目標、ゴール)
論理的探索の最終目標。

1.4.20 goal-driven reasoning →backward reasoning

1.4.21 heuristics (ヒューリスティックス)
必ずしも常に成立するとは限らないのですが、多くの場合、問題解決に有効な働きをする個人的・経験的知識を言います。例えば、碁・将棋の定石などがそうです。これと対立する知識は、数学や物理学的法則のように、常に成り立つ知識です。

1.4.22 heuristic method (発見的方法)
試行錯誤などのように、生々しい実践的な方法

1.4.23 IF-THEN rule ()
多くの人工知能システムで採用されている法則であって、条件部と結論部とから構成されます。

1.4.24 induction, inductive reasoning/thinking/inference (帰納法、帰納推論)
論理的な方法の一つであって、個別の事柄から出発して、一般的な結論を導く方法です。演繹的推論と対比して用います。

1.4.25 inference, reasoning (推論)
或る原因から結果を導く過程で、論理の筋道を立てること。日常的な事象には reasoning の方を使うようです。

1.4.26 inference engine (推論エンジン)
エキスパートシステムを、知識のデータベースを別にして、実際に推論を実行するプログラム部分のことをエンジンと呼ぶようになりました。

1.4.27 inheritance(継承)
階層構造(hierarchy)になっている事象で、下位にある事象の性質が、上位の事象から引き継いで利用できること。知識を階層構造的に表すときなどに用いられます。

1.4.28 instantiation (具体化)
算術変数に対する論理的な変数は、論理値を表現できるような論理項に変更して処理します。文で書いた構造を、記号の並びに換える形式論理学がその具体化です。

1.4.29 intelligence (知能)
思考と推論によって知識を取得し、かつ応用できる能力。

1.4.30 knowledge (知識)
学習、知見、発見、推論、経験、理解などで得られたこと。

1.4.31 knowledge acquisition (知識獲得)
エキスパートシステムを構築する際に、専門家の頭の中にある知識を、コンピュータが利用できる形に変換する技法。

1.4.32 knowledge base(知識ベース)
データベースの連想から生まれた言葉で､エキスパートシステムの知識部分に相当するrules, facts, heuristicsを、問題解決に適するように集積したもの。

1.4.33 knowledge representation (知識表現)
知識や情報を、コンピュータが処理できるような用語、記号、文法などに表現することです。

1.4.34 logic (論理)
演繹(deduction)によってものごとを説明していく(reasoning)方法であって、前提とそれを解析していく過程を含みます。

1.4.35 mathematical induction (数学的帰納法)
整数Ｎ以上の自然数を含む項に関する命題を証明する方法です。まずその命題がＮについて成立することを示し、次に、もしＮ以上の任意のｎに対してもその命題が成立するならば（ｎ＋１）に対してもその命題が成立することを示すことによって証明する方法。

1.4.36 meta-knowledge (メタ知識)
知識についての知識。英語の 接頭辞に使う meta- は、日本語に訳し難い、また納得し難い、抽象的な用語の代表的なものです。強いて訳すと、後ろの言葉を繰り返して、「何とかの何とか」とします。その言葉を含める、さらに広い概念を意味します。

1.4.37 metaphor (隠喩)
修辞学で使う言葉で、意味的には本来使うべきでないのに使う表現法です。例えば「おへそが茶を沸かす」などです。

1.4.38 metarule
ルールについてのルール。接頭辞の meta がついた言葉ですので訳し難い用語です。「ある規則を利用するときの規則」とでも解釈しますか。

1.4.39 model inference system (モデル推論システム；ＭＩＳ)
エキスパートシステムを構築しようとするとき、或るモデルをあらかじめ設定して推論する方法。

1.4.40 modas ponens (モーダスポネンス)
ラテン語が原義の推論規則の一つであって、二つの命題から一つの結論を導く規則。三段論法が一つの例である。これと対比する用語は、modas tollens(モーダストレンス)。こちらは、対偶の関係です。

1.4.41 natural language processing(NLP;自然言語処理)

1.4.42 predicate(述語)
主題(proposition)を説明する部分。

1.4.43 predicate calculus (述語計算)
ＡＩのプログラムにおいて、一般的に用いられる推論の方法であって、データ間の関係に注目する。

1.4.44 premise (前提)

1.4.45 production rules (プロダクションルール)
生成規則または書き換え規則とも言います。単純にはIF-THEN ruleと考えてよい。

1.4.46 propositional calculus (命題計算)
論理規則の集合から結論を導き出す推論の方法。

1.4.47 pruning (枝刈り)
探索木構造をたどって探索をするとき、効率を上げるために不要の枝部分を省くこと。

1.4.48 reasoning (推論) →inference

1.4.49 rule(規則)

エキスパートシステムにおいては、IF-THEN の組みあわせた文の集合。

1.4.50 rule of thumb ()

親指を立てて「これだ」という欧米人の習慣からきた用語。Heuristics のこと。

1.4.51 shell (シェル)

もとは、UNIX システムにおけるコマンド解釈のプログラムを言いました。転じて、普通の意味でのプログラミングの方式でなく、一種のコマンド処理に近い様に簡略化したソフトウエアツールを呼びます。例えば、expert system shell など。

1.4.52 theorem (定理)

幾つかの公理から帰納的に正しいと証明が導かれた結論。

1.4.53 tree search (木構造探索)

木構造に構成されているグラフをなぞって探索すること。

1.5 論理式に使う記号

1.5.1 言葉の記号化の背景

話している言葉（音）を図形化して目に見える形にするのが**文字**(character)です。音だけを記録する表音文字を、英語ではletterと言います。英語のアルファベット文字は、正確に言うときはalphabetic letterと使います。日本語の仮名は、Kana letterと言います。日本語では単純に文字と括るのですが、英語の character は、ラテン文字（またはアルファベット）・数字（digit）・記号(symbol)、そして空白や改行など行わせる字形も音も無い機能文字(function code)、の四種を合わせて言います。漢字は表意文字ですので、Kanji character と使います。漢字本家の中国では、基本的に漢字 1 字は１音節(syllable)と１意を表します。日本語の環境では、和語の言い方に 輸入した漢字を当てましたので、漢字の読みの種類に音訓二種を持ち、それぞれに二つ以上あることがあって、複雑になってしまいました。表音文字並びは、何かの意味を持たせる言葉を表すとき、文字数を多く必要とします。例えば、数字の 123 を"one hundred and twenty three"と書くのを spell out と言います。この例では文字数を 9 倍多く書きます。英語の文単位では文頭を数字で始めることをしませんが、少し硬い文書では文中でも spell out する例を見ます。日本語の文書でも、算用数字を使うか漢数字を使うかの区別に注意します。この教材では、例えば、名詞は漢数字を使って二値論理学と書き、算用数字の２値論理学とは使いません。算用数字を頭に使うと、その名詞を見出し語に使うとき、見出しの番号との区別ができません。

1.5.2 省略法

表音文字を使う英米語では、文字数の多い単語を文中で何度も引用するとき、省略(abbreviation)する書き方が多く用いられます。例えば、The United States of Americaは、昔はU.S.Aとし、省略した事を意味する点(.)を挿入しましたが、それも省く USA が認められるようになりました。読むときは元のスペルで発音するのが基本的な態度ですが、「ユーエスエー」と言って通用するようになると正式な名詞扱いになり、辞書にも載るようになります。専門用語の省略語には大文字小文字を使い分けることがありますので、辞書に載らない省略語にするときは、約束を始めに書いておくのを正式な作文技術とします。英語のellipsisは**省略記法**と言い、ピリオドなどを使った省略の書き方を言います。「...」「***」などと書くのがそうです。科学技術を研究する場合、英米語では、図形記号を別に工夫しないと文書に記録する方法に不便です。これは省略法の工夫と似ています。既に単語があったものに記号を工夫して読みを当てます。数学、論理学、集合論の記号に見られます。一方、漢字はもともと一字で意味を表しますし、扁（へん）や旁（つくり）を図形の構成要素とした造語法です。この造語法を真似て、日本で作られた漢字を和製漢字または国字と言います。例えば、吋・呎は国字ですが、英語のインチ・フィートを表し、その読みで使います。漢字の画数が多いものを単純化した表記にしたものを**略字**と言いますが、英語の省略法とは違います。中国本土では、読み（音）を同じにした記号化で作られた漢字があって、**簡体字**と言います。元の字形を**繁体字**と言います。日本で開発された仮名（平仮名、片仮名）は漢字の意味の方を捨象した略字と言えるでしょう。

1.5.3 図記号は文字扱いをしない

図記号(pictogram)は、記号文字(symbol とは別です。従来から専門ごとの省略表示に種々の工夫がされて使われていました。東京オリンピック(1964)や大阪万博(1970)のとき案内図表示に種々のアイディアが応用されたことで、一般の人の関心が寄せられるようになりました。図記号には、名前を付けることもしますが、読みは定義しません。図記号の目的は、種々の言語環境からくる人が、文字を読めなくても意味を理解できることを意図したからです。パソコンの画面で使う小寸法の**アイコン**(icon)も、図記号です。こちらは、コンピュータに文字並びを知らせて、仕事をして貰い、ユーザの入力手間を省く目的で開発されました。しかし、アイコンの種類が増えてくると、そのアイコンが何をするかが判らなくなるようにもなりました。パソコンの単位画面であるフォームは、文字表示のメニューバーから文字並びを選択させる画面デザインが基本です。アイコンを表示するツールバーは必須のデザインではありません。しかし、メニューバーを省き、ツールバーだけでパソコンを制御するようにしたものがあります。マウスが使えなくなると、そのプログラムを利用する手段が全く無くなることが起こります。約束を決めた文字並びを入力すると、それが表す記号文字を利用できる方法が定義されるようになりました。その応用の一つが、数式編集用のプログラミング言語の **TEX** です。また HTML(Hyper Text Markup Language)でも、キーボードに無い特殊な文字や記号を表示させる文字並びの約束が定義されています。この目的は、通信回線を使ってキーボードにない文字列を送受信するときに、送受信側で約束した同じ文字記号を再現させるときに使います。**HTML** で使う記号文字（タグ）の約束を見て下さい。

1.5.4　キーボードにない記号の扱い方

印刷物にして論理的な説明をしたいとき、やや特殊な記号や文字は、活字を別に作りました。数学記号は比較的利用の頻度が高いのですが、活字を組むときに特別な約束が多いので、科学技術関係の書籍と文科系書籍の出版社などと、得意分野の専門区別があります。個人単位で文書原稿を作成するとき、欧米語ではタイプライタが標準的な道具ですが、日本語では手書き原稿が普通です。ワードプロセッサは、英文タイプライタのキーボードを利用します。このとき、そこで使える文字キーの組み合わせで特殊文字や記号の情報を入力します。初期の英文タイプライタでは、記号文字の種類は僅か 10 個程度です。電報の送受信に使ったテレタイプライタ(TTY; Teletypewriter)は、英小文字を使いませんでしたので、そのコード系は６ビットコードで間に合いました。初期のコンピュータは、入出力装置に TTY を使いました。数学式で扱う大小関係を表す記号（＜、＞）がありませんでしたので、プログラミング言語の FORTRAN では英字も大文字だけを使い「.LT.」「.GT.」のような表記法を使いました。コンピュータ用のキーボードは、IBM 社の提案になる 101-key keyboard の仕様が標準となって、英小文字も使えるようになり、かなりの数の記号文字が使えるようになりました。しかし、数式記号にはギリシャ文字なども使いますので、キーに文字や記号を割り当て、コードを決めることには限界があります。これは、通信回線を使ってデータを送受信し、同じ記号文字を再現するときの障害になります。漢字を使う日本語の文書では、漢字の種類が多いので、同じ問題を、ローマ字入力→仮名変換→漢字変換の方法で解決しました。論理学の記号と集合論の記号とは異なった図形仕様ですので、その比較一覧を表 1.1 に示します。

表 1.１　種々の記号

論理学用語	論理学の記号　(*1)	集合論の記号	計算機言語 (*2)	英文　(*3)	和文　(*3)
否定　(*1)	~~P~~, ￢Ｐ,	記号に上線を引く	NOT	not	…でない
選言、論理和	Ｐ∨Ｑ　(*4)	Ｐ∪Ｑ　(*5)	OR	or, and/or	または
排他的選言、排反、非両立	Ｐ⊻Ｑ　(*6)	Ｐ≠Ｑ	XOR	exclusive or	どちらか
連言、論理積	Ｐ∧Ｑ,　Ｐ＆Ｑ	Ｐ∩Ｑ	AND	and	かつ
内含、含意	Ｐ→Ｑ、	Ｐ⊃Ｑ Ｑ⊂Ｐ	IMP	if～then, imply	ならば
可逆的内含、対等、同値	Ｐ≡Ｑ	Ｐ⇔Ｑ,	EQV	equivalence	等しい
全称記号		∀ｘ, Πｘ		all	すべての…
存在記号		∃ｘ, Σｘ		some	あ(或)る…
帰属		∋、∈		inclusion	(部分を)含む
包摂		⊇、⊆		subsumption	(全部を)含む

注
(*1)この教材の説明では、論理変数の記号(取り消し線を使う~~P~~)の方を表記に用います。
　　式全体を否定にする場合は、式を括弧 () で括って、その前に￢の記号を書きます。
(*2)BASIC言語で用いる論理演算子の表し方を例示します。
(*3)日常言語のなかで使われる表現と、論理学での定義とでは少し異なることがあります。
(*4)全角英字のＶと字形が似ていますので注意して区別して下さい。
(*5)全角英字のＵと字形が似ていますので注意して区別して下さい。
(*6) 選言記号∨にアンダーラインを付けた表記です。

1.6　参考文献

[著者名のアイウエオ順です]

（1）　内井惣七、「いかにして推理するか、いかにして証明するか」、
ミネルヴァ書房、1981／1992、ISBN4－623－01340－5

（2）　内井惣七、「うそとパラドックス」、
講談社現代新書 881、講談社、1987／1991、ISBN4－06－148881－3

（3）　内井惣七、「パズルとパラドックス」、
講談社現代新書 970、講談社、1989／1991、ISBN4－06－148970－4

（4）　内井惣七、「真理・証明・計算　－　論理と機械　－　」、
ミネルヴァ書房、1989／1992、ISBN4－623-0188Ⅳ－9

（5）　大出　晃、高野守正、「論理学」、
慶応通信、1977／1990　　ISBN4－7664－0199－9

（6）　沢田允茂、「現代論理学入門」、
岩波新書 C14、岩波書店、1962／1982

（7）　沢田允茂、「考え方の論理」、
講談社学術文庫 45、講談社、1976／1992、ISBN4－06－158045－0

（8）　思想の科学研究会編、「増補改訂－哲学・論理用語辞典」、
三一書房、1959、ISBN4－380-75202x

（9）　ジャン・ショーヴィノー、芹沢正三訳、「記号論理学」、
文庫クセジュ 257、白水社、1959／1990、ISBN4－560－05257－3

（１０）千葉茂美、東　千尋、若山玄芳、「論理学入門」、
学陽書房、1974／1992、ISBN4－313-3500Ⅰ－2

（１１）中村秀吉、「パラドックス」、
中公新書 297、中央公論社、1972／1992、ISBN4－12－100297－0

（１２）野崎昭弘、「詭弁論理学」、
中公新書 448、中央公論社、1976／1992、ISBN4－12－100448－5

（１３）野崎昭弘、「逆説論理学」、
中公新書 593、中央公論社、1980／1989、ISBN4－12－100593－7

（１４）平凡社、「哲学事典」、
平凡社、1971、ISBN4－582-10001－5

（１５）山崎正一＋市川浩編、「現代哲学事典」、
講談社現代新書 225、講談社、1970、ISBN4－06－115625－x

（１６）山下正男、「論理的に考えること」、
岩波ジュニア新書 99、岩波書店、1985、ISBN4－00－500099－1

（１７）吉田夏彦、「論理と哲学の世界」、
新潮選書、新潮社、1977／1991、ISBN4－10－600191－8

2.　論理演算

2.1　文を記号化する方法

2.1.1　主語述語の揃った文を関数と考える

「この花は赤い」という文を考えます。(この花) を記号 x に置き換え、(…は赤い) の述語の部分を、記号 F で表し、「この花は赤い」を、Fx と書くとします。または、数学と同じように $f(x)$ のような関数形で表すと考えることもできます。このように扱うとき、**関数論理系**の用語を使います。この Fx の表し方を**命題関数**と言います。なぜ関数と言うかの理由は、「真・偽」二種類のうちの一つを出力する関数の性質を持たせるからです。 x は、数学では**変数**ですが、論理関数の場合は**変項**と言います。代数の演算は、関数の値を別の変数に代入することをしますので、$P = Fx$ と書く代入操作をして、P を**論理変数**、または単に変数と使うことをします。Fx は、一つの命題を記号化する書き方です。読み方は、「x は F である」です。日本語の言い方の順とは逆で、記号文字の表し方が英語流になっています。意味としては、定義文または宣言文であって、述部の動詞が先行します。ここでの主語は、文法用語と異なる定義であって、**主辞**の用語が使われます。主辞が複数のときもあります。これを意識したのが集合論です。主辞は、名詞を当てるのが普通ですが、論理学では名詞の代わりに**名辞**の用語を使います。また、述語は動詞です。「である」は英語の be 動詞に当たります。日本語では名詞で受けるか形容詞の終止形の場合があり、これらも名辞の扱いをします。「x は y の z である」のような文、さらには「x は y と z とで決まる」のような文は、変項が増えた複合命題です。

2.1.2　定言命題とその種類

定言とは、「かつ」「または」「ならば」などの接続詞や条件文形式を含まない、言い切りの文単位を言います。数学において、$y = f(x)$ を考えるとき、x の取り得る範囲に制限がつく場合があります。例えば、

$$f\ (x),\ (a < x \leqq b)$$

命題関数の場合、変項 x に制限がつくものを**束縛変項**と言います。束縛とは、変項 x に個数を考えることを言い、制限のない場合を**自由変項**と言います。英語は、名詞の単数・複数を神経質に区別します。命題関数では、x が複数個である場合の言い方の区別を、さらに二通りに分けます。全部と部分です。英語では「all, some」です。これを日本語に訳して言うときは、「すべて、或る」を当てます。

$\forall x\, Fx$;　　全称記号 $\forall$ を使う場合（束縛変項）
$\exists x\, Fx$;　　特称記号 $\exists$ を使う場合（束縛変項）
Fx ;　　なにも使わない場合　（自由変項、単に論理変数Pのようにも書きます）

表 2.1　定言命題－記号とその読み方

符号	定義の用語	記号での表し方	文章としての読み方	備　考
	単純な命題	Ax	・ x は A である。(単腹の区別をしないとき)	
A	全称肯定命題 (全称命題)	$\forall x\, Ax$ $(\neg \exists x \neg Ax)$	・すべての x は A である。皆……である。 ・「A でないものは存在しない」と読む、 ここでの「¬」は、否定の演算子記号です。	全部肯定 皆　(*2)
E	全称否定命題	$\neg \forall x\, Ax$	・すべてのものは A でない。 ・A であるものは存在しない。 ・皆…A でない。	全部否定 皆不
I	特称肯定命題 (存在命題)	$\exists x\, Ax$	・或るもの（こと）は A である。 ・A であるものが存在する。　(*1) ・皆…でない、ことはない。	部分肯定 不皆不
O	特称否定命題	$\exists x \neg Ax$	・A でないものが存在する。 ・或るものは A でない。 ・皆…である、ことはない。	部分否定 不皆

注(*1)：日本語では、存在を表す言葉に、「居る(いる)」と「在る(ある)」とを使い分けます。
「居る」は、人、動物などの生きている対象に用います。「在る」は、植物などの静物、無生物を対象とします。英語には、この言い換えの区別がありません。
(*2)：漢語では否定を表す「不」の使い方が合理的にできて、４つの命題を区別できます。

2.2 演算の表し方

2.2.1 論理変数の考え方

記号論理学は、代数学と同じように、文字や記号を使って文の性質や関係を扱います。記号論理学で使う用語と記号文字の書き方については、現在のところ 標準化は曖昧です。日本語のワードプロセッサで表記できる制限を考えると、第1.4.4項、表１.1のような種類があります。コンピュータが無かった時代は、文字並びに書いて、計算（演算）手順を説明しました。コンピュータ処理は、その文を解読（compile）して計算します。二値論理学に応用する代数学を**ブール代数**と言います。命題を形式化して、Ｐ，Ｑなどの文字を当てた論理変数で表します。その変数の取ることができる値を論理値と言い、（真・偽）の二つしかない、とします。この二つの論理値は、コンピュータの処理に向くようにするときは、２進数の（1.0）に置き換えます。そうすると、コンピュータメモリのビット並びを、演算の媒体に使うことができます。プログラミング技術の立場から言うと、コンピュータの演算では８ビットの集合を１バイトとし、幾つかのバイト数の並びで文字や数を表します。このとき、ビット並びの個別の位置ごとに変数を割り当てることもします。整数として利用する複数のバイト並びを論理値に代用するときは、ビット並びがすべて０であるときは偽、それ以外は真とする約束を使います。整数（－１）を表すビット並びは、すべてのビットをon(１のこと)にすると約束しています。ビット並びを、数を扱う場合と論理値を扱う場合とを混用して使うこともありますので、プログラミングでは注意が必要です。

2.2.2 演算の基本は二つの変数を使う

数を扱う算術の算法は、加減乗除の四則が基本です。この処理を表すとき、二つの変数を仮に文字Ａ，Ｂで代表させ、算術演算子(operator)の記号（＋、－、×、÷）を挟んで、例えば「Ａ＋Ｂ」のように書きます。このとき、書き順違いの「Ａ＋Ｂ」と「Ｂ＋Ａ」とは同じ結果が得られます。これは**交換法則**または**可換律**(commutative law)が成り立つと言います。この性質があるとき、筆者は「演算子が**対称**である」の言い方を使います。引き算と割り算とは交換法則が成り立ちません。これらは、記号式の書き方の約束ですので、一意に結果が決まります。しかし、これを文章で表現するときは、英語と日本語のような言語違いと共に、書き順の違いで計算処理が変ることがあります。割り算「Ａ÷Ｂ」の場合の英語の文章表現では、「Ａ is divided **by** Ｂ getting Ｃ」と「Ｂ divides **into** Ａ getting Ｃ」の言い方があって、変数Ａ、Ｂの書き順が逆です。英語風の構造を持ったコンピュータ言語の COBOL は、前置詞違いの二つの表現法を許しています。日本語で言えば、「Ａ割るＢはＣである」の言い方に対して、「ＢをＡに割り付けるとＣ個に分かれる」の言い方に当たります。後者の方は、殆ど使いません。

2.2.3 関数にまとめる考え方もある

二つの変数を使う演算の表記法に、関数(function)で表す略記の方法があります。例えば、足し算を、関数 Add(A,B)と書き、結果をＣに代入する処理を「C=Add(A,B)」と書きます。コンピュータ言語では、イコール記号「＝」を使う文形を、**代入文**(assignment expression)と言います。**式**(equation)と言うときは、変数と演算子記号を並べた擬似的な文表現(expression)を言います。式で表すとき、Add(A,B)と Add(B,A)は同じ結果が得られます。しかし、例えば、引き算「Ａ－Ｂ」を Sub(A,B)と表記を約束したとき、変数の並び順を換えた Sub(B,A)は、別の結果になります。したがって、関数を提案するときは、名前の約束と同時に、演算順序も定義しなければなりません。論理演算の場合にも、「文章表現・演算子を使う表現・関数を使う考え方」があります。関数は、論理演算の種類を説明するときには便利な考え方です。関数を使う方法では、演算に使う変数の個数を 0,1,2,3…のように撰ぶことができます。個数が 0 の関数は馴染みがありませんが、プログラミングでは、例えば円周率πの値を参照したいとき、定数を関数並みに引用するときに応用します。

2.2.4 サブルーチンに構成する考え方もある

整数を考えた割り算の計算結果は、**商**と**余り**（剰余）の二つが得られます。関数を使う方法では結果を一つしか返せませんので、余りの計算値を返す演算子（Visual Basic では**Mod**）を定義する、または関数を別に決めて使う方法の他、にサブルーチンを約束し、結果を返す引き数を余分に用意することがあります。論理数は（真・偽）二つしか値をとりませんし、一回の演算で複数の結果を返すことは矛盾ですので、サブルーチンに構成する考え方をしません。

2.3　変数を一つ使う演算

2.3.1　代数学ではごく普通に使う関数形式

数学で扱う三角関数「sin, cos, tan」は、入力変数一つを使い、一意の結果を一つ返す典型的な**一価関数**です。この逆関数もあります。しかし、入力する変数値の範囲に制限があります。結果も複数考えることができますので、**多価関数**です。したがって、一つの結果だけに制限する使い方の約束を決めます。一価関数と似た処理に、電卓（卓上計算機）には**符号変換**のキー「±」があることを再認識して下さい。演算処理では「－１」を乗じ、文字表記では、－符号を頭につけます。実は＋の場合にはこの記号を省略する約束ですので、－符号がついた数の符号変換は、－符号を消すか、＋記号に換えます。ここでの－符号は、「－１」を乗じる使い方をしています。文字としての「－」表記は、二つの顔を持ちます。引き算の演算子で使う場合と、符号変換の演算子です。電卓での操作を考えると分かります。負の数の入力は、－記号を最初に使います。符号変換は、演算の最後に±キーを押します。演算子の組み合わさった代数式の演算順序では、乗除の演算子で繋がった個所を先に計算しておいて、＋または－の計算を後にします。この約束が演算子の**優先順位**です。実際に数値計算をするときは、乗除の演算子の左右部分を先に計算し、それを個別に保存しておいて、最後に全体を加算します。引き算は、－符号の付いた数の足し算をする計算です。例えば、「２＋３×４」と表記した式の計算結果は 14 です。電卓のキーをこの表記順で押すと、結果は 20 です。表記順と計算手順とが変るのは、かなり深刻な問題であって、プログラミング言語の設計とその利用では注意が必要です。

2.3.2　二値の論理学は NOT を一価関数で考える

数の符号変換と似た形式論理学の演算に**否定**があります。ただし、この演算ができるのは二値論理学に限ります。多値論理学では、否定文を書くことはできますが、結果が一意に決まりません。例えば、ジャンケンで「グーではない」と否定しても、パーかチョキかは決まりません。二値論理学では、否定の演算は、「真・偽」を入れ換えます。ブール代数に応用するときは、数の「1,0」の対を「真・偽」の代わりに使いますので、偽の表現を 0 ではなく、－１と当てる間違いをすることがあります。数の大小関係を扱う集合論では、数の「負・0・正」の判定を使うこともします。これは三値論理学です。このときには「-1,0,1」で状態を表すこともしますので、それと混同しない注意が必要です。三値論理学の場合であっても、命題の「真・偽」の判断をします。このときに使う演算子を**関係演算子**と言います。「＜、＞、＝」の他に「≦、≧、≠」を必要とします。これらの記号は、日本語の環境では全角の記号文字を使うことができます。半角文字を使うことを標準としている英字の環境では、二つの記号文字を並べて「<=, >=, <>」と書きます。関係演算子を使う式の全体は論理式の扱いをしますので、演算結果は二値論理の定数の「真または偽」を返します。このときに、数字の「1,0」を考えると、間違った判断をすることがあります。それを避けるために、プログラミング言語では、論理定数の文字記号「True, False」を表記に使う約束を決めています。

2.3.3　論理演算子としての NOT

二値の論理演算は、変数の値としての「真・偽」を反転する方法を用意します。このときに、前章の 4.1.1 項、表 1.1 に紹介した否定の演算子と、その使い方を決めるのですが、これが悩ましい問題です。その理由は、通常のワードプロセッサでは表現できない特殊な記号文字または編集用記号を使うからです。高校数学の教科書では、集合論の否定を表す演算表記は、変数または変数を使った式の文字並びの上に横線を引きます。ワードプロセッサは、下線(underline)を引く処理はありますが、上横線を引く方法がありません。文字高さの中央に取り消し線を引くことができますので、この教材は、変数記号に便宜的にこの方法も採用しています。マイナス符号を使う方法に倣って、否定の演算子を決めて、論理変数の前に書く方法も利用します。日本語の全角文字では「￢」が定義されています。英字のまま Not を演算子に使うこともします。否定の演算は、電卓の符号変換と似たところがあります。二度続けて演算すると元に戻るからです。この演算の応用として良く知られている法則が、**二重否定律**です。コンピュータ言語では、この演算に使う演算子記号の約束とその使い方については、混乱があります。論理変数の型を明示的に定義できるもの（Visual Basic の Boolean）と、そうでない場合（C 言語）があります。Ｃ言語では、「！」（感嘆符；exclamation mark）を論理否定の演算子にしています。

2.3.4　一価関数として扱う否定演算の規則

代数学では、一価関数であることの標準的な表記法は f(x)です。変数 y に代入する表記では、x を**媒介変数**(parameter)、y を**従属変数**(subordinate variable)と呼び、解釈としては x を y に変換するとします。関数 f(x)の表記を演算子に換える例が、数の符号変換に使うマイナス記号「－」です。論理式は、否定の演算子です。演算子記号は、変数 x の前に書きます。一価関数のもう一つの解釈は、電気電子工学で利用され、x を**入力**(input)、y を**出力**(output)とする変換装置です。論理数Ｐの否定は、~~Ｐ~~, ￢Ｐまたは NotＰのように表記します。一価関数の表し方は、Ｑ＝Ｆ(Ｐ)の形を考えます。論理数の取り得る値は「真・偽」の二つしかありませんので、ＰからＱへの変換は４通りしかありません。これを表 2.2 にまとめます。なお、この表では「真・偽」を「1,0」で表しています。

表 2.2　変数一個の論理演算則の表

番　号	否定演算の表記法	Ｐの値	Ｑの値	定　　義	文章での言い方
1	~~Ｐ~~	1	1	トートロジー	ＰはＰである
2	￢Ｐ	1	0	Ｐの否定、NOT	Ｐでない
3	￢(Ｐ)	0	1	Ｐの否定、NOT	Ｐでない
4	NotＰ	0	0	矛盾	ＰはＰでない

2.3.5　補数を求める演算

二つの正の数値Ａ,Ｂを扱うとき、例えば(Ａ＋Ｂ)＝10 のとき、一方を他方の「10 の**補数**(complement)」と言います。２進数の場合は(1,0)が互いに補数の関係になります。高度のプログラミングをするとき、コンピュータメモリのビット並びを、個別に独立した二進数の扱いをして、それらのビット位置に何かの情報を与える使い方をすることがあります。例えば、個人情報で男女の区別をするとき、文字で区別するときは 2 バイト（16 ビット）のメモリ領域を取りますが、１ビットの on/off で済ませることができて、全体のメモリ寸法の節約にもなります。バイト並びのデータの、どのビット位置で、何を目的に使うかを設計するプログラミングは、整数型のデータを使って、その全体を論理数の集合扱いをします。このプログラミングは、コンピュータのハードウエアと関連を持ちますので、プログラミングの初等教育では、普通、ビット処理を説明しません。この整数型のデータは、或るときは全体としての整数を表し、或るときは論理型のデータと見なして、ビット位置ごとに AND や OR の計算をさせます。Ｃ言語は、プロのプログラマ向けの仕様を持たせていますので、特殊な論理演算子を使うことができます。論理値は(1,0)で表しますので、NOT の処理を、引き算の「１－P」で代用するような補数計算を考えるのですが、この計算はビット並びのデータには使うことができません。つまり全体を独立したビット並びとして論理演算をさせる、特別な補数計算をさせる演算子が必要です。

2.4　変数を二つ使う演算

2.4.1　二値論理学での演算は１６種類

算術演算の基本は、二つの変数を使う加減乗除、つまり４種類の算法です。二つの変数 A, B を使う算法を記述するとき、２変数の間に演算子を挟む方法、例えば「A＋B」とします。関数形式、例えばAdd(A,B)は、少し複雑な条件を考えるときの書き方です。どちらにしても、計算結果を判断するとき、もう一つ別の変数 C を考えて、C＝…のような代入文を使います。二値論理学では、数式に置き換えて代数学的に変数を使う算法は、次のような特徴があります。(1)変数の取り得る値が二種類「真または偽」しかないこと；(2)演算の結果も二種類に限ること；(3)変数の書き順の前後関係を考える必要があること、です。この組み合わせは 16 通りですので、演算の種類は 16 通りあります。算術演算に使う演算子は四種類ですが、実はマイナス記号「－」は二つの使い方があります（2.3.1 項参照)。論理演算の演算子は、連言(And)、選言(Or)、内含(Imp)の三つと、否定(Not)とを組み合わせれば 16 通りの演算を処理できます。ただし、内含は、割り算と同じように、変数の書き順を逆に考えることをしません。演算の種類を**表 2.3** に示します。この表で使っている演算子の記号は、論理学の方の「∧、∨、⇒」を使いました。論理演算を視覚的に理解するときは、**図 2.1　ベン図**(Venn)が便利です。

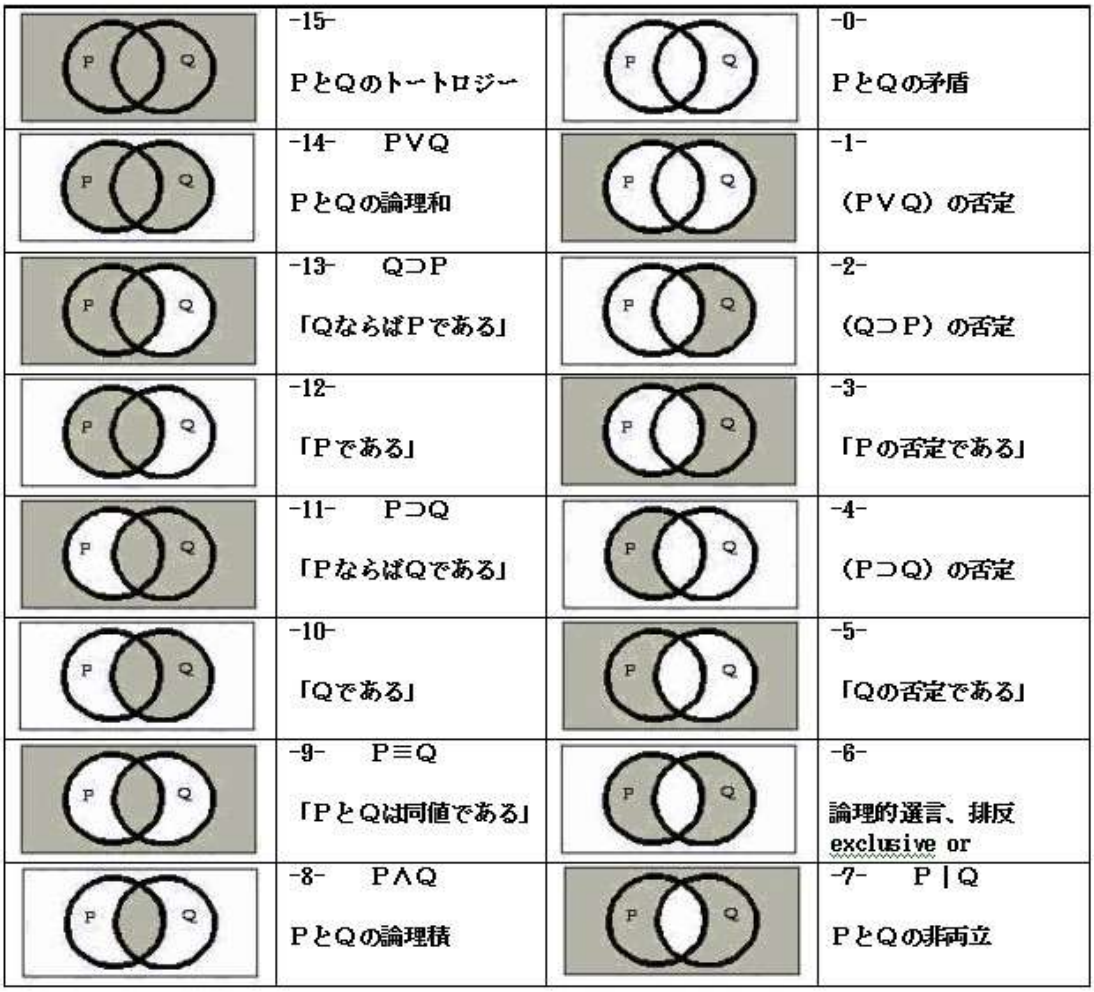
表 2.3　変数を二つ使う論理演算則の表

番号 (*1)	記　号	P，Q の組み合わせ				定　　義	文章で言うとき	備考 (*2)
	P	1	1	0	0			
	Q	1	0	1	0			
15		1	1	1	1	ＰとＱのトートロジー	（Ｐ，Ｑ）→（Ｐ，Ｑ）	対称
14	Ｐ∨Ｑ	1	1	1	0	ＰとＱの選言、論理和、OR	ＰまたはＱである	対称
13	Ｑ⇒Ｐ	1	1	0	1	ＰによるＱの逆向きの内含	ＱならばＰである	
12		1	1	0	0	Ｐである	［常にＰである］	
11	Ｐ⇒Ｑ	1	0	1	1	ＰによるＱの内含	ＰならばＱである	
10		1	0	1	0	Ｑである	［常にＱである］	
9	Ｐ≡Ｑ	1	0	0	1	可逆的な内含、対等、EQV		対称
8	Ｐ∧Ｑ	1	0	0	0	ＰとＱの連言、論理積、AND		対称
7	Ｐ｜Ｑ	0	1	1	1	ＰとＱの非両立		対称
6	Ｐ∨Ｑ	0	1	1	0	ＰとＱの排反、XOR	ＰまたはＱのどちらか	対称
5		0	1	0	1	Ｑの否定である		
4		0	1	0	0	（Ｐ⇒Ｑ）の否定		
3		0	0	1	1	Ｐの否定である		
2		0	0	1	0	（Ｑ⇒Ｐ）の否定		
1		0	0	0	1	（Ｐ∨Ｑ）の否定	ＰでもＱでもない	対称
0		0	0	0	0	ＰとＱの矛盾		対称

(*1)：この番号数字は、「Ｐ，Ｑ の組み合わせ」の欄の４つの二進数並びを十進数にしたものです。
(*2)：対称とは、交換法則が成り立つ演算の意味で使いました。

2.4.2　演算子の種類を減らすこと

二値論理学の演算の種類は 16 通りありますが、そのすべてに演算子記号を割り当てるのではなく、基本とする演算子を組み合わせて演算を表すことができます。日本語の環境では、否定・連言・選言・内含の４種類を基本とします。記号で表せば、（¬、∧、∨、⇒）です。これを組み合わせてすべての演算を定義できます。これらが論理法則または論理公式になります。これらは、変数記号を三つ以上使いますので、次の 2.5 節でまとめて説明します。英語の環境であるコンピュータ言語では、記号に換えて文字並びを使います。「Not, And, Or, If-then」と書きます。これは、キーボードから使える記号の種類が少ないからです。対等の演算子の代わりに「＝」も使います。コンピュータ言語では、排他的選言（exclusive or；表 2.3 では∨）と、対等(equivalence、表 2.3 では≡）を加えるものがあります。演算子記号は、コンピュータ言語ごとに仕様が異なります。



[備考：ここに示した番号は、表 2.3 左欄の数字番号に合わせてあります]

図 2.1　ベン図

2.4.3　演算子記号の呼び方

ベン図は、文章論理の形式的な演算の説明用と、数学寄りの集合論の演算説明用にも利用できる分かり易い図です。しかし、この演算の説明に使っている演算記号は、表 2.4 のように使い分けます。論理学で使う内含の記号⇒は、集合論では左右反対向きの記号⊃と⊂を使い分けることもします。図 2.1 のベン図では、便宜的に記号⊃を使っています。論理式を使うとき、演算子には特殊な字形の活字（フォント）を別に準備しなければなりません。文書の閲覧にインターネットを利用する電子書籍が普及してきましたので、HTML 文書の原稿を作成するときは、この演算子記号の送受信に使う表記法と、記号の呼び方が定義されています。これを表 2.4 に含めました。

表 2.4　演算子記号の読み方

日本語の言い方	論理学			集合論		
	記号	英語の読み方	HTML の表記法	記号	英語の読み方	HTML の表記法
および、かつ、連言、論理積、合接、積集合	∧	And	&and;	∩	Cap	&cap;
または、選言、論理和、離接、和集合	∨	Or	&or;	∪	Cup	&cup;
ならば、内含、部分集合	⇒	Imply	&rArr;	⊃ ⊂	Superset Subset	⊃ ⊂

2.4.4　ベン図は集合論の図であること

図 2.1 のベン図は、二値論理学の説明には誤解を起こします。その理由は、領域を表す図形に、丸が二つと外枠の矩形を考えて三つあるからです。集合論の説明図として見ると、二つの円が離れている場合、一方が他方に含まれる場合を扱っていません。二つの円が交差する図は、領域が四つできますので、演算の結果が四通りあります。しかし、この演算結果すべてに論理的な意味付けができません。集合論の説明にベン図を使うときは、二円（P、Q）が交差している場合、二円が離れている場合、PがQの内部にある（またはQがPの内部にある）場合を考えます。これを文で表現すると、表 2.5 に示すように四通りあります。記号（⊃と⊂）及び（⊇と⊆）は、論理学では使いません。

表 2.5　集合論で「或るとすべて」を考えた干渉記号

番号	２円（P、Q）の位置関係	記号式	言葉による説明
1	P、Qが離れている	P∩Q＝0	共通部分がない（空集合）
2	P、Qが交差している	P⊃Q または Q⊂P	Pの部分集合がQに在る
		P⊂Q または Q⊃P	PはQの部分集合を持つ
3	Pの内側にQが入る	P⊇Q または Q⊆P	PはQをすべて含む
4	Qの内側にPが入る	P⊆Q または Q⊇P	Pの部分集合はQである

2.4.5　集合論に論理学を応用するときの混乱

集合論は、一つ二つと数えられる**要素**（通常は普通名詞）の集まり（集合名詞）を**変数**単位にします。集合を論理的に扱うときは、要素を「持つか・持たないか」または要素が集合の中に「存在するか・しないか」を「真・偽」の区別に使います。このとき要素の数え方を「全部・部分」で大まかに区別します。言い方としては、英語の数量形容詞「all, some」を訳して、「すべて・或る」を当てます。要素を全く持たない場合を表す用語に**空集合**が定義されています。集合要素を数量形容詞で修飾した文が論理学の命題扱いです。例えば、「すべての猫は動物である」は真であり、「或る猫は動物である」は偽と判別します。このとき、猫が何匹居るかの具体的な数を使わないで、「居る・居ない」の二つの状態だけを区別するときに、フール代数で考えます。しかし「ここには猫が三匹も居る」「猫が一匹も居ない」の文を真としたいとき、ブール代数の「1」で考えると混乱を起こします。これを避ける一つの実践的な方法は、具体的な数を捨象し、真の場合を扱う論理変数「P、Q」は肯定文としておいて、この否定形の論理変数を別の変数記号「~~P~~、~~Q~~」と書いて追加し、四変数間の論理演算の組み合わせを扱います。演算子の種類で、連言と選言とを扱うときを表 2.6 に示します。この演算則は交換法則が成立します。内含の演算子⇒の場合、交換法則が成立しません。演算則は、表 2.7 にまとめます。

2.4.6　演算子には対称・非対称の区別と優先順位がある

二つの変数を使う演算は、二変数の間に演算子を挟む書き方をします。この演算で、変数の書き順を左右入れ換えても結果が同じであれば、この演算子は**対称**であるとし、同じにならなければ**非対称**であるとします。算術の演算子では、引き算記号と割り算記号とが非対称の演算子です。論理演算では、内含が非対称な演算子です。英語の環境では、演算子を使わないで、if-then で書きます。一価関数ｙ＝ｆ(ｘ)は、ｘからｙへの向きの演算ですので、非対称の演算の約束をしています。したがって、これに演算子記号を約束すると、非対称の性質を持ちます。実際の演算処理を考えると、変数を表記してから、それに続けて演算子を書くのが理屈なのですが、代数式では演算子記号を先に書きます。算術計算では－記号、論理式では Not を表す記号です。この書き方の約束は、他の演算に先だって、最高の優先順位で実行させることを意味しています。論理式では、否定の演算子の優先順位を最も高く約束します。否定を表す上横線の書き方は、ワープロでの表記では使えません。もしそれを使う場合の演算は、その線で繋いだ全体を先に計算しておいてから否定の演算をする約束です。否定の演算子、例えば¬を使うときは、否定する式全体を括弧で括っておいて、その括弧の先頭に演算子記号を付けます。もうひとつ、イコール記号「＝」の使い方があります。数学式では左右を入れ換えても意義としては同じです。しかしプログラミングでは、＝を含む式は**代入文**とされ、右辺の演算の結果を左辺の変数に代入しますので、演算の流れは逆向きであり非対称です。論理演算では、等値であることを検査する演算子の使い方をするため、＝に換えて≡を使いますが、こちらは対称な演算子です。ただし比較を行わせる演算と代入は、優先順位を最も低くしますので、これを使う場合には、比較計算をする演算の範囲を、あらかじめ括弧で括っておく書き方をして、計算順位の間違えを防ぎます。

表 2.6　否定文と組み合わせた「論理積・論理和」の演算則

番号 (*1)	記号 (*2)	論理値の組み合わせ				文章での表し方	備　　考
	P					「Pである」ときを真(1)	論理値の組み合わせの第２列は、肯定、否定を、第３欄を横に見て、「1,0」を代入するように演算する。
	Q	1	1	0	0	「Qである」ときを真(1)	
	~~P~~	1	0	1	0	~~P~~＝「Pでない」ときを真(1)	
	~~Q~~					~~Q~~＝「Qでない」ときを真(1)	
14	P∨Q	1	1	1	0	PまたはQである	
13	P∨~~Q~~	1	1	0	1	PまたはQでない	（Q⇒P）と等しい
11	~~P~~∨Q	1	0	1	1	Pでない、またはQである	（P⇒Q）と等しい
7	~~P~~∨~~Q~~	0	1	1	1	Pでない、またはQでない	（P∧Q）の否定…(*3)
12	P∨P	1	1	0	0	PまたはPは、Pである	（P∨P）≡P…(*4)
15	P∨~~P~~	1	1	1	1	PまたはPでない	排中律…(*4)
8	P∧Q	1	0	0	0	Pであり、かつQである	
4	P∧~~Q~~	0	1	0	0	Pであり、かつQでない	（P⇒Q）の否定と等しい
2	~~P~~∧Q	0	0	1	0	Pでなく、かつQである	（Q⇒P）の否定と等しい
1	~~P~~∧~~Q~~	0	0	0	1	Pでなく、かつQでない	（P∨Q）の否定…(*3)
12	P∧P	1	1	0	0	PかつPは、Pである	（P∧P）≡P…(*4)
0	P∧~~P~~	0	0	0	0	Pであり、かつPでない	…(*5)
7	P｜Q	0	1	1	1	PかつQである、ことはない	
11	P｜~~Q~~	1	0	1	1	PかつQでない、ことはない	（P⇒Q）と等しい
13	~~P~~｜Q	1	1	0	1	PかつQである、ことはない	（Q⇒P）と等しい
14	~~P~~｜~~Q~~	1	1	1	0	~~P~~かつ~~Q~~である、ことはない	（P∨Q）と等しい
9	P≡Q	1	0	0	1	Pと、Qと、が等しい	
6	P≡~~Q~~	0	1	1	0	Pと、Qの否定と、が等しい	（P⩒Q）と等しい
6	~~P~~≡Q	0	1	1	0	~~P~~と、Qと、が等しい	（P⩒Q）と等しい
9	~~P~~≡~~Q~~	1	0	0	1	~~P~~と~~Q~~とが、等しい	（P≡Q）と等しい
15	P≡P	1	1	1	1	PはPと等しい	同一律と言う
6	P⩒Q	0	1	1	0	Pであるか、またはQである	
9	P⩒~~Q~~	1	0	0	1	Pであるか、またはQでない	（P≡Q）と等しい
9	~~P~~⩒Q	1	0	0	1	Pでないか、またはQである	（P≡Q）と等しい
6	~~P~~⩒~~Q~~	0	1	1	0	Pでないか、またはQでない	（P⩒Q）と等しい
0	P⩒P	0	0	0	0	矛盾である	

[備考]

(*1) この番号は、表 2.2 の最左列に示した番号との対応を示します。

(*2) 記号⩒を、排他的選言（XOR）に用いました。

(*3) ド・モルガンの法則。

(*4) トートロジー。

(*5) トートロジーの否定を、矛盾律と言います。

・論理法則として知られているものの幾つかが、表 2.6 の中に含まれています（備考参照）。

・ 論理法則は、大別して二種類あります。一つは、論理式そのものがトートロジー（恒眞式）であるもの。もう一つは、ある論理式と、別の表現の論理式とが同じ論理値になる場合で、「式A≡式B」のように書き表すことができるときです。

・ 記号「≡」は、数学記号の＝」と考えてもよいし、また、演算子と考えることができます。後者の場合、演算結果がトートロジーになりますので、この論理法則は、すべて恒眞式です。

・ 「二重否定は肯定を表す」、と言う法則を「二重否定律」と言います。否定の記号に上横棒をつける方法は、ワープロ表記に向きません。否定記号に¬を使う場合、この法則は、「¬¬P≡P」と表すことができます。

・ 排他的選言(XOR) は、コンピュータ言語の論理演算子には良く用いられます。XOR の演算「P⩒P」は、あるレジスタのビット並びをすべてクリアするときに応用します。

表 2.7　　否定文と組み合わせた「ＰならばＱである」論理演算則

番号 (*1)	記号 (*2)	論理値の組み合わせ				文章での表し方	備　　考　(*3)	
	Ｐ					「Ｐである」ときを真(1)	論理値の組み合わせの第２列は、肯定、否定を、第３欄を横に見て、「1,0」を代入するように演算する。	
	Ｑ	1	1	0	0	「Ｑである」ときを真(1)		
	~~Ｐ~~	1	0	1	0	~~Ｐ~~＝「Ｐでない」ときを真(1)		
	~~Ｑ~~					~~Ｑ~~＝「Ｑでない」ときを真(1)		
11	Ｐ⇒Ｑ	1	0	1	1	ＰならばＱである	（Ｐ｜Ｑ）と等しい	A
7	Ｐ⇒~~Ｑ~~	0	1	1	1	ＰならばＱでない	（Ｐ∨Ｑ）と等しい	B
14	~~Ｐ~~⇒Ｑ	1	1	1	0	ＰでないならばＱである		C
13	~~Ｐ~~⇒~~Ｑ~~	1	1	0	1	ＰでないならばＱでない		D
13	Ｑ⇒Ｐ	1	1	0	1	ＱならばＰである	（Ｐ｜Ｑ）と等しい	a
7	Ｑ⇒~~Ｐ~~	0	1	1	1	ＱならばＰでない	（Ｐ∨Ｑ）と等しい	b
14	~~Ｑ~~⇒Ｐ	1	1	1	0	ＱでないならばＰである		c
11	~~Ｑ~~⇒~~Ｐ~~	1	0	1	1	ＱでないならばＰでない		d
	Ｐ⇒Ｐ	1	1	1	1	または　~~Ｐ~~⇒~~Ｐ~~	トートロジー	
	Ｐ⇒~~Ｐ~~	0	0	0	0	または　~~Ｐ~~⇒Ｐ	矛盾	

［備考］:

・ 内含記号⇒で繋いだ論理演算は、交換法則が成り立ちません（非対称)。代数式に書く場合、向きを反対にした逆向きの内含記号も使いません。一方、集合論では、図形Ｐが図形Ｑの全部または部分を含むことを図に描いて説明できますが、そのときの記号表示では、表 2.4 で示したように、コとこととを文表現に直しても誤解されません。

(*1) この番号は、表 2.2 の最左列に示した番号と同じである対応を示すものです。
同じ番号同士は、同じ演算結果になることを示しています。

(*2) 上の４つの演算式で、左右の英字変数記号を入れ換えたものが、その下の４つの演算式です。
これを最右欄の英字記号の大文字・小文字の対応「A,B,C,D」、「a,b,c,d」で示しました。
下の４つの式は、上４つの記号⇒を逆向き使う表記法と同じです。しかし、式を文に直して言うときは、読み順が逆になるので、⇒だけを使うようにします。

(*3) 右端の英字「大文字・小文字」の対応は、図 2.3 の中で、論理式の種別を示すときに使います。

2.4.7　関数の考え方が応用される論理回路

論理演算子は、二つの論理変数から一つの論理値を決定する計算に使いますので、数学的には二価関数、例えばｚ＝ｆ(ｘ,ｙ)と考えることができます。さらに、この演算を電気・電子工学の論理回路の設計に応用するときは、固有の演算回路の図記号を利用します。これを図 2.2 に示します。演算の呼び方は、表 2.3 と対応を付けると、下のようになります。

回路の名称	表 2.2 の番号	用語
AND	8	論理積
OR	14	論理和
NAND	7	論理積の否定
NOR	1	論理和の否定
EXOR	6	排他的論理和
ENOR	9	対等

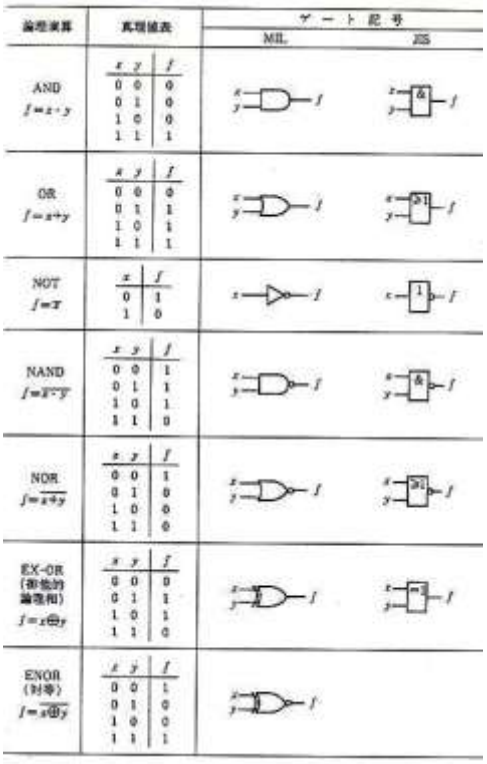


図 2.2　論理回路の図記号

2.4.8　対偶・対当・順・逆・裏の説明

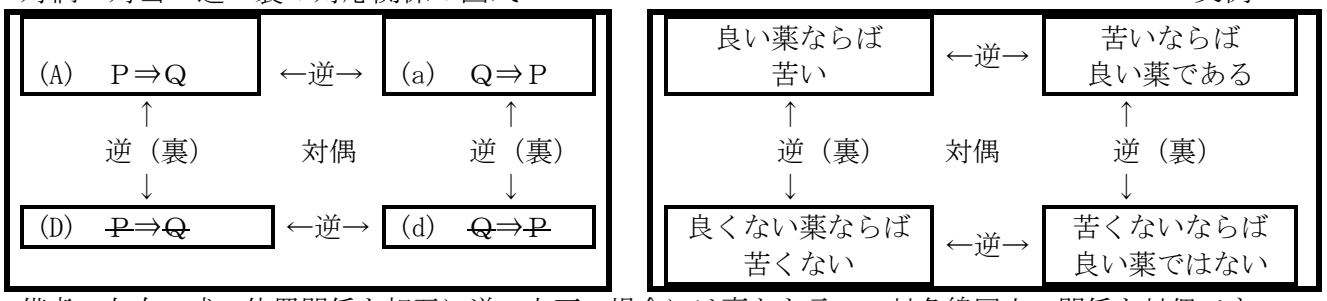
論理学では、似たような文であっても、その表現方法の違いを厳格に区別して、「正しい・誤り」の判断をします。文を変形する一つの方法は、否定表現です。二つの文を演算子で繋ぐ擬似的な演算では、左右を入れ換える変形があります。表 2.3 に説明した演算子で、対称と注記したものは、左右を入れ換えても結果は変りません。二つの文を対称な演算子で繋ぐ例が表 2.6 です。その中に、注(*3)式で表現した**ド・モルガンの法則**があります。式違いで二ヶ所にあることに注意して下さい。

一方、内含の演算は非対称です。否定文にすることと合わせて、８通りの言い換えの種類があります。演算式の表し方ではなく、文例を使った例を図 2.3 に対応関係として示します。その元にする例文を「良薬、口に苦し」としましょう。これは、薬と言う名詞を主語として、「良い・良くない」の二値の性質に分ける命題と、「苦い・苦くない」の二値に分ける命題との組み合わせ（論理演算）を考えます。正確な命題の文は、主語・述語を明確に言う「Ｐ：この薬は良い」「Ｑ：この薬は苦い」の言い方を単位とします。下に示す図 2.3 では、これらの文を繰り返すとくどくなりますので、省略した言い方で書いてあります。Ｐ、Ｑから複合命題に演算させるとき、連言（かつ）と選言（または）で繋ぐ演算では文の並び順を換えても論旨は変りません。内含で繋ぐ複合文は、文の並び順を換えると、論旨も変ります。そこで、最初の基本とする言い方を「良い薬は苦い」とし、これを「**順**」とします。

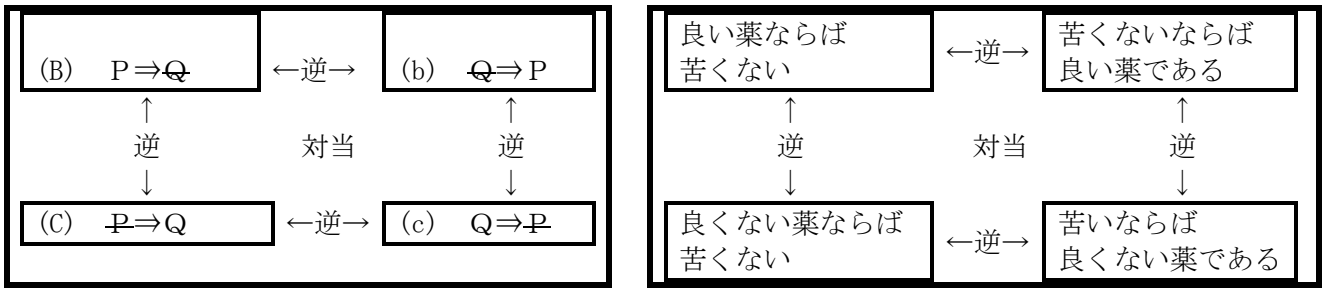
「ＰならばＱである」（文記号 A）と、「ＱならばＰである」（文記号 d）との関係は、互いに他の**逆**と言います。論理表を見れば解るように、逆は必ずしも同じにはなりません。「ＰならばＱである」の論旨は、Ｐが真であるときのみ、文の真偽が正しい意味を持ちます。Ｐが偽であるときのことは何も言ってないのですが、真偽値は、常に眞（１）に設定されます。これを普通の文章表現では「ウソからでたマコト」と言うことがあって、実際の文章に使うときは**虚偽**の扱いとします（第 1.3.27 項「前件否定の誤りと後件肯定の誤り」参照）。

否定形を組み合わせた二つの文関係の性質は、「**逆**」、または「**裏**」の用語で区別する組み合わせがあります。文表現が変っても、本質的な論旨が変らない関係になる文の対を**対偶**と言います。対偶が成り立つことを規則にしたものを**対偶律**と言います。反対になる文の対を**対当**と言います。この議論は、二つの文単位の関係ではなく、実は文単位で勘定すると、八通りの言い換えの文から四つの文をとりだして、その組み合わせを扱っています。変数を三つ以上扱う論理式は次節の第 2.5 節で、改めて取り上げます。

対偶・対当・逆・裏の対応関係の図式　　文例



備考：左右の式の位置関係を相互に逆、上下の場合には裏とも言い、対角線同士の関係を対偶です。



備考：左右及び上下の式の位置関係を相互に逆、対角線同士の関係が対当です。

図 2.3　否定文と組み合わせた内含の対応関係：「良薬、口に苦し」の変形

2.4.9　トートロジーとは演算結果があたりまえのことを言う

表 2.7 の下欄の２行は、内含の演算の特殊な場合を示したものです。「ＰならばＰである」は、論理的には「あたりまえ」のことを述べた命題ですので、論理値は常に「真」を当てます。論理学の用語ではトートロジーと言います。論理式で表したものでも結果が常に真になるものを恒真式と言い、トートロジーの言い換えです。その幾つかは論理公式に加えることあります。ＰならばＰである（Ｐ⇒Ｐ）は、その最も単純な場合です。

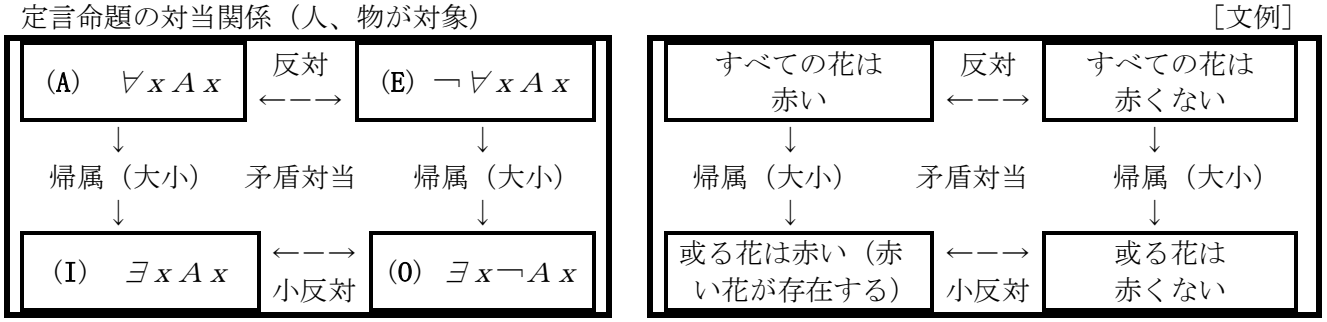
文章で表すとき、「そんなことはあり得ない」と言う結論になるとき、論理学では矛盾とします。論理式で、結果が常に偽になる場合です。しかし、文章論理を展開していくと、真とも偽とも判定ができない場合が起こることがあります。これは、そもそも、その文が真偽の二値に分けられないか、三値論理学の変数であるときに起こります。

2.4.10　集合を扱うときの論理文の構造

そもそも、代数式は、言葉（文）で説明することを記号の並びに置き換えたものです。逆に、式を文に直して読み上げることもします。論理学で扱う文単位は名詞を主語としますので、その名詞に数の形容詞を付ける言い方を必要とします。具体的に数を使うのではなく、大小、多少の区別ができるとして、「すべて」と「部分（或る）」二つの対立概念に分けます。この数量形容詞は、抽象的な性質をもった言葉です。これに加えて、「肯定・否定」の二つ対立概念を組み合わせると４つの文形が得られます。例文を加えて、下の４つの言い方を示します（第 1.1.4 項,命題を参照）。

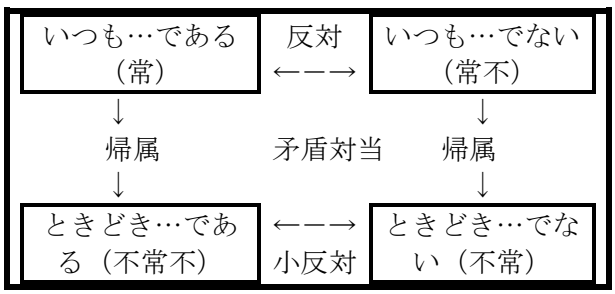
Ａ；　全称肯定命題、「すべての花は、赤い」
Ｅ；　全称否定命題、「すべての花は、赤くない」
Ｉ；　特称肯定命題、「或る花は、　　赤い」
Ｏ；　特称否定命題、「或る花は、　　赤くない」

この４つの文を記号式に表し、さらに相対的な意味関係を図 2.3 に倣って図 2.4 のグラフで示します。



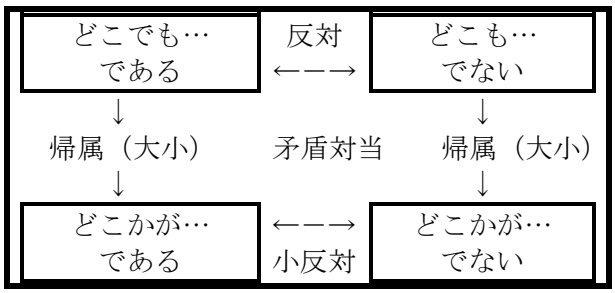
［備考］：　縦方向の文並びは、**帰属**または**包摂**の関係。（内含は集合論の用語では使いません。）
左右方向は互いに否定の関係ですが、**反対**と**小反対**の用語で区別をします。
対角線同士の関係は、矛盾です。これを**矛盾対当**と言います。

図 2.4　反対・小反対・矛盾・内含の対当関係及び実例



［解説］
文章を書くとき、４Ｗ１Ｈを明らかにすることが必要です。「いつ（時間）」、「どこ（場所）」、「だれ（人）」、「なに（物）」に加えて、Ｈにあたる「どのように」を使うときの対当関係を例にして、以降の幾つかの図式を示します。対角線関係は矛盾対当です。同時に使うと矛盾ですので注意します。

図 2.5　時間の長短を数形容詞で扱うときの言い方



［解説］
　この図式は、場所の情報を含む文で、広さの概念を使い分けるときの言い方の図式です

図 2.6　場所の広さを大小関係の数形容詞で扱うときの言い方

大小の数学的対当関係　　［文例］

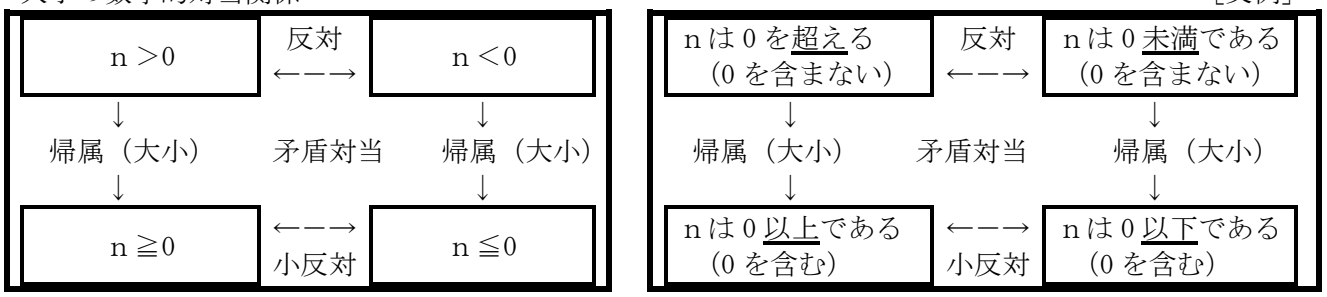


図 2.7　数の大小を数式で言う時と言葉で言うとき

言葉で言うとき　　［記号で表した場合］

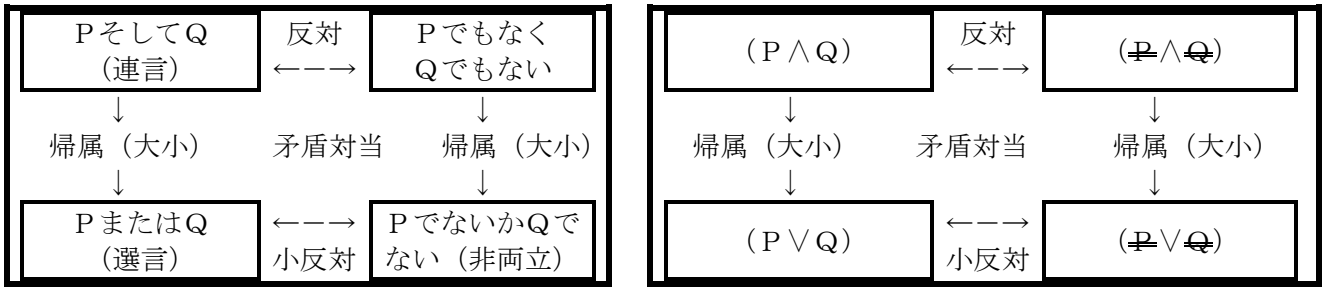


図 2.8　複合命題の対当関係の図式

［解説］

・ 複合命題、特に、連言と選言との対当関係を示す図式です。表 2.6 の演算式相互の関係を示します。この図式の関係は、文章において否定文と組み合わせて接続詞「かつ」と「または」を使い分けるときに、論理的に厳密に吟味するときに必要な知識です。この図式には、論理則の幾つかが含まれています。例えば、矛盾律は、対角線どうしの要素について、下の式が恒真式になることです。

［（P∧Q）∨（¬P∨¬Q）］

2.4.11　規則や法律を論理的に作文する文の構造

人間社会では、多くの約束事を決めています。これが法律や規則です。それを文に表すとき、正しい理解が得られ、また、矛盾のないようにまとめるには論理学の素養を踏まえます。これらの文では、特別な用語や言い回しがあります。また、その文を理解するときの概念も使われます。まず、**必然**と**可能**の概念の説明をし、それを使い分けるときの対当関係を図式に示すことから始めます。

論理学での必然とは、「必ずそうなる」ことを言います。数学での式の結果や、論理式での結果のような場合が論理的な必然です。しかし、分野によって解釈が少し異なるものがあります。道徳的な必然は、**当為**と言われ、当然なすべきものを指します。現実的な必然は、**偶然**に対するもので、「太陽は東から昇る」などを言います。「必ずそうなる」の反対が「必ずそうならない」です。「必ずしもそうならない」とは、全体に対する部分の意味を持ちますので、帰属の関係です。

可能とは、論理的に矛盾がないことを言います。必然の場合には結果の予測がただ一通りであるのに対して、可能では複数の結果が予想されるものの中で、一つを指します。「可能である」とは「そうなることもある」のことです。この反対は小反対であって、「そうならないこともある」と言い分けます。

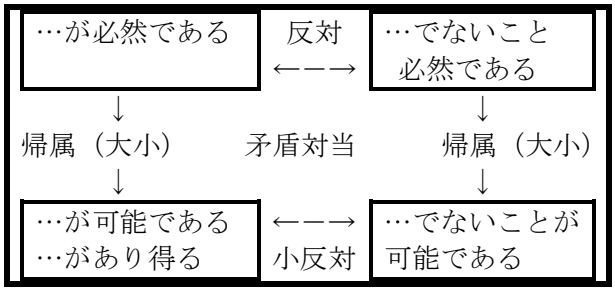


図 2.9　可能と必然の対当関係の図式

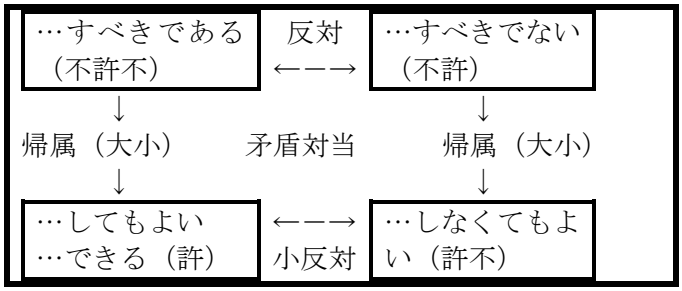
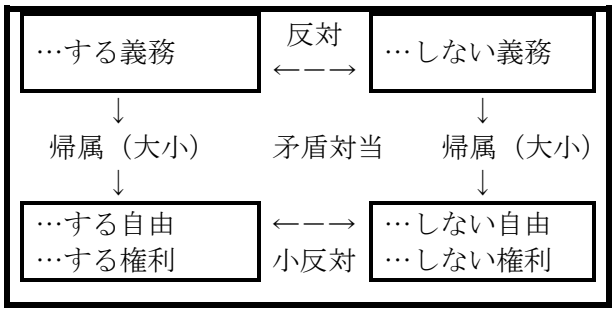


図 2.10　命令と許容の対当関係の図式

必然と可能とは、「…になる」の言い方の区別です。「…をする」の言い方を区別することが、**命令**と**許容**です。「…しなさい」が命令、「…してもよい」が許容です。「…しなさい」の言い換えは、「…せよ」「…すること」「すべきである」などの言い方になります。英語では、動詞の命令形のほか、shall, must の助動詞を使います。「…しなさい」の否定が**禁止**です。日本語では「…してはならない」の否定形を取ります。「…してもよい」の言い換えは、「…することができる」「…することが許される」などとなります。英語では should, it is allowed, may, can などを使い分けます。日本語の命令文と許容文では、主語を明示してなくても二人称です。英語では、主語に日本語の目的格を使うこともあります。

2.4.12　人の社会的行動を規制する用語



［解説］
この文の組み合わせは、前項の命令と許容の対当関係とほぼ同義です。命令と許容は、相手に対して働き掛ける言葉遣いですが、義務と権利とは、自分自身に科する行動規制です。自分と相手との間でこの対当関係を認めることが契約です。社会的な意味での不文律は、前の 2.4.10 項の分類では**礼儀**と、この項の**行儀**とが対応しています。

図 2.11　義務と権利の対当関係の図式

2.5　変数を三つ以上使う演算

2.5.1　算術演算の場合の計算順序

算術計算の手順（ステップと言います）を示すために、数と演算子記号の文字並び（＋－×÷）とを交互に書くことを考えて下さい。数が二つで、その間に演算子記号が入る場合の実際計算は、簡単に理解できますし、計算手順を間違えることはありません。例えば、「２÷３」と「３÷２」とは違う結果が得られます。この理解は、割り算記号の意味に、左から右向きに文字と記号並びを判断して演算する約束があるためであって、それが自然です。この約束は、電卓を使うとき、または算盤（そろばん）を使うときの作業手順と同じです。しかし、数を三つ以上使い、演算子記号が二つ以上表れるときは、この単純な約束がいつも使えるとは限りません。例えば、「5＋3÷2×7」の文字並びを挙げましょう。電卓を使う場面で、この文字並びの順にキーを押して、最後にイコール記号「＝」を押せば、答えは 28 と得られます。しかし、上の文字並びを数学的に解釈すると、15.5 が解です。この違いは何から来るかと言うと、演算子に**優先順位**があって、掛け算と割り算は、足し算を引き算よりも先に計算を済ます約束があるためです。これは面倒な約束です。数学に達者である人であっても、予測した手順が実際の計算手順にならない事態も起こります。数式を書くときは、括弧を適当に使って、その中の計算を先に済ますようにします。複数の括弧の対が、入れ子状に構成されるときは、最も内側の括弧から計算を済ませて行きます。その括弧の中だけの答えが得られれば、その括弧を削除します。これを繰り返して行って、括弧が無くなったところで、単純な加減算が残り、結果が一意に決まります。上の文字並びならば、次のように括弧を補います；「5＋(3÷2×7)」。電卓は、総計用メモリ(GT: grand total)を一つ使うことができますので、そこに個別に得られた計算結果を加算して行きます。しかし、式が複雑になると、GT 一つでは足りなくなりますので、メモ用紙に途中結果を書き出しておいて、後から参照します。参照を間違えないようにする方法も技術ですので、ここに上手下手の差が出ます。

2.5.2　コンパイラは文字処理のプログラムである

パソコンを使って数値計算をさせるときは、何かのプログラミング言語を使って文字で書いた代数式を解読させて、実行形式のプログラムに翻訳（コンパイル）して計算させます。つまり、**コンパイラ**は文字処理のプログラムです。パソコンで行わせる数値計算の機械的な手順は、電卓の動作手順と全く同じです。電卓では手作業でレジスタに数値をセットするのに対して、パソコンでは数値が格納されたメモリからデータを作業用レジスタに読み込む機械語の命令があります。途中の計算結果を一時的に保存する作業用メモリも複数使うことができます。計算結果も、或る特定のメモリに転送させます。これを**代入**と言います。具体的な数値をあらかじめメモリに読み込ませておくことと、結果を眼に見えるようにモニタに表示させるか、印刷で得るために、入出力の命令が別に必要です。この操作をすること全体がユーザインタフェースです。変数文字と演算子記号の並びで書かれたプログラム文を解読して機械語の実行手順に構成するとき、コンパイラは演算子の演算順序を勘案し、擬似的に括弧を挿入したような文字並びに換えます。このアルゴリズム（算法）の組み立ては、プログラミング言語の提案をするときに工夫が必要な個所の一つです。そうであっても、数値計算をさせるプログラミングは、元の式に括弧を明示的に使うようにして、計算手順が文字並びと違って前後しないように書くことが望ましいのです。

2.5.3　演算子は計算手順の向きも決められている

代数学の知識が一般化していますので、気にしなくなりましたが、二つの変数と一つの演算子を使う文字並び、例えば「Ａ＋Ｂ」と書けば、左から右への順で計算します。電卓にデータ入力をする順と考えることができます。このとき、Ａを演算子の**左辺値**、Ｂを**右辺値**と言います。掛け算と足し算では、左辺値と右辺値とを入れ換えても同じ結果です。筆者は、これを対称な演算子と言うことにしました。数学の用語では、交換法則が成り立つ演算子と言います。引き算と割り算とでは、Ａ,Ｂを入れ換えると別の結果になります。これを明示的に説明するとき、「演算を左辺値から右辺値へ向う右方向で評価する」と言います。ところが、プログラミング言語でイコール記号を使う式、例えば「Ａ＝Ｂ」の意味は、右辺値を左辺値に**代入**する左向きの処理です。数学式を扱うとき、例えば「Ｃ＝Ａ＋Ｂ」を「Ａ＋Ｂ＝Ｃ」と書き換えても、意義的にはイコール記号の左右が等しいことを示します。しかし、プログラミング言語で演算の指示をする文では、この書き換えを許しません。紛らわしいことが、比較演算子の使い方のときに起こります。プログラミング言語の Visual Basic では、If で始まる条件文の中で使うイコール記号は、左辺値と右辺値とが等しいときに True を返す論理演算子です。Ｃ言語では、代入文との紛らわしさを避けるため、左右が等しいことを評価する比較演算子の方は、==としています。

2.5.4　括弧を使うときの実用的な規則

数値計算を実行するとき、最も自然に評価できる式の書き方は、左から右に文字並びを見て、その順に計算が実行できるときです。この節の始め、第 2.5.1 項で挙げた例題「5＋(3÷2×7)」の場合は、5 の足し算を保留しておいて後から足すか、電卓の場合には GT を一つの足し算用のバッファー（一時的な記憶領域）に使うことができます。式を評価するとき、左括弧が現れたら、一旦処理を中断して、そこまでの処理が再開できるように保留しておいてから、改めて括弧の内側から文字並びの解読を 0 から始めます。括弧で括ることは演算手順を入れ子状に構成することです。左括弧は、演算子と考えることができて、その優先順位は、代入演算子「＝」の一つ手前の最低位です。左括弧が複数連続すれば、その順で入れ子の構成が深くなります。右括弧は、最後に現れた右括弧の対となる演算子であって、優先順位は最高位で扱います。これが現れたところで、括弧の内側を計算し、その結果を持った仮の変数に置き換え、当該の括弧の対を消去します。同時に、入れ子構造では一段解上の保留状態に復元し、そこから処理を再開します。言葉で説明するとくどいのですが、実際の作業で混乱することはありません。括弧に括る演算単位は、二つの変数と一つの演算子を使うのが基本です。そして、明示的に括弧を使って計算式を書く場合にも、この形でまとめるようにします。例えば、上の算術式は、「5＋ ((3÷2) ×7)」または「5＋ (3÷(2×7))」と書くことができます。二通りの書き方ができる理由は、演算子×÷が優先順位で同順位だからです。ここまでの説明は、次項から始める、論理変数三つ以上を使う論理演算の組み立ての予備知識であるとして理解して下さい。

2.5.5　論理演算子の種類と優先順位

論理変数を三つ以上並べる論理式は、算術式の場合と同様に、論理変数と演算子とを交互に並べた文字列を、原則として、左から右の順に評価します。そこで、三つの論理変数を扱う場合を基本知識としてまとめておきます。論理変数記号をＰ,Ｑ,Ｒと表記し、その間に二つの論理演算子＃$_1$、＃$_2$を挟む式を考えます；

Ｐ ＃$_1$ Ｑ ＃$_2$ Ｒ　　式 2.1

演算子の種類とその記号は下の 6 種が主なものです。

¬、∧、∨、⊻、≡、⇒

演算子の名称で言えば「否定、論理積、論理和、排反、同値、内含」です。プログラミング言語の Visual Basic では、変数の型として明示的に Boolean 型が定義されていて、この演算子を下のように、英字並びで利用します。理由は、コンピュータの英字用キーボードでは、特殊な記号文字が使えないからです。

Not, And, Or, Xor, Eqv, Imp

二値の記号論理学では、この順を優先順位としています。また、同順位の演算子はありません。他の言語では、論理型変数の定義そのものを持っていないことも多く、整数型の変数を代用しています。論理型で扱うデータは、条件文(If 文)の引き数として、関係演算や比較演算が使われ、その演算結果を論理型として得て、分岐の判定に間接的に利用しています。これらの論理演算子の中、最初の三つが基本であって、残りは、この三つを組み合わせて使うことで実用的な演算処理ができます(前の第 2.4 節参照)。しかし、複雑な論理式を処理したいときは、6 種類を使えることが基本です。

2.5.6　論理変数三個を使う論理演算

括弧を使わない前項の式 2.1 を評価するとき、原則は、最初の演算子＃$_1$で繋がった２変数Ｐ,Ｑ間の演算を先に済ませ、その結果を仮の変数に代入しておいて、二番目の演算子＃$_2$と次の変数Ｒとの演算をさせます。これは、括弧を使う下の演算式を考えることです；

（Ｐ ＃$_1$ Ｑ）＃$_2$ Ｒ　　式 2.2

しかし、演算子の優先順位が＃$_1$をよりも＃$_2$の方が高ければ、この計算手順は間違いです。コンパイラは、この場合、括弧の使い方を下の式 2.3 のように評価します；

Ｐ ＃$_1$（Ｑ ＃$_2$ Ｒ）　　式 2.3

演算子は６種類ありますが、否定の演算は個別の論理変数ごとに（真・偽）を扱うことにします。三つの式、(2.1)～(2.3)のグループごとに 25 通りの種類がありますので、演算の種別は 75 通りもあります。論理演算を提案するときは、できるだけ二変数を扱う演算を明示的に括弧で括る式を使います。括弧の使い方に関係しないときは、二つの演算子が同じで対称な場合だけです。内含の演算子⇒は非対称ですので、括弧を左側で使うときと、右側で使うときとで演算の結果は異なります。表 2.8 は、括弧の使い方によって演算結果が変わること見るために作成しました。括弧を使わない論理式をプログラミング言語のコンパイラが演算順序を組み立てるときは、演算子の優先順位を考えます。そのときに、どこに括弧が入るかを下段の方にまとめました。

これを明示的に括弧を使うときの結果を、表 2.8 の左欄に示します。括弧を右側の２変数の計算をまとめるようにした場合、その演算結果を対応させて右欄に示します。左右が同じ結果の場合と異なる場合があるのは、演算子に優先順位があるときです。この左右の演算結果を参考にすると、括弧を使わない場合の演算の結果が理解できます。これを表 2.8 の下段に示しました（表の行数を詰めるため、左右の欄を使っています)。

表 2.8　変数三個の論理演算則の一覧表

番号	記　号	P,Q,R の真偽値	備考	番号	記　号	P,Q,R の真偽値	備考
	P	1 1 1 1 0 0 0 0			P	1 1 1 1 0 0 0 0	
	Q	1 1 0 0 1 1 0 0			Q	1 1 0 0 1 1 0 0	
	R	1 0 1 0 1 0 1 0			R	1 0 1 0 1 0 1 0	
	論理式	論理式の関数値			論理式	論理式の関数値	
左側二変数を括弧で括る場合（式 2.2）				右側二変数を括弧で括る場合（式 2.3）			
32L	（P∨Q）∧R	1 0 1 0 1 0 0 0		32R	P∨（Q∧R）	1 1 1 1 1 0 0 0	(*C)
42L	（P⊻Q）∧R	0 0 1 0 1 0 0 0		42R	P⊻（Q∧R）	0 1 1 1 1 0 0 0	
52L	（P≡Q）∧R	1 0 0 0 0 0 1 0		52R	P≡（Q∧R）	1 0 0 0 0 1 1 1	
62L	（P⇒Q）∧R	1 0 0 0 1 0 1 0		62R	P⇒（Q∧R）	1 0 0 0 1 1 1 1	(*G)
23L	（P∧Q）∨R	1 1 1 0 1 0 1 0		23R	P∧（Q∨R）	1 1 1 0 0 0 0 0	(*B)
43L	（P⊻Q）∨R	1 0 1 1 1 1 1 0		43R	P⊻（Q∨R）	0 0 0 1 1 1 1 0	
53L	（P≡Q）∨R	1 1 1 0 1 0 1 1		53R	P≡（Q∨R）	1 1 1 0 0 0 0 1	
63L	（P⇒Q）∨R	1 1 1 0 1 1 1 1	(*5) (*H)	63R	P⇒（Q∨R）	1 1 1 0 1 1 1 1	=63L
24L	（P∧Q）⊻R	0 1 1 0 1 0 1 0		24R	P∧（Q⊻R）	0 1 1 0 0 0 0 0	
34L	（P∨Q）⊻R	0 1 0 1 0 1 1 0		34R	P∨（Q⊻R）	1 1 1 1 0 1 1 0	
54L	（P≡Q）⊻R	0 1 1 0 1 0 0 1		54R	P≡（Q⊻R）	0 1 1 0 1 0 0 1	
64L	（P⇒Q）⊻R	0 1 1 0 0 1 0 1		64R	P⇒（Q⊻R）	0 1 1 0 1 1 1 1	
25L	（P∧Q）≡R	1 0 0 1 0 1 0 1		25R	P∧（Q≡R）	1 0 0 1 0 0 0 0	
35L	（P∨Q）≡R	1 0 1 0 1 0 0 1	(*C)	35R	P∨（Q≡R）	1 1 1 1 1 0 0 1	(*E)
45L	（P⊻Q）≡R	0 1 1 0 1 0 0 1		45R	P⊻（Q≡R）	1 0 0 1 0 1 1 0	
65L	（P⇒Q）≡R	1 0 0 1 1 0 1 0		65R	P⇒（Q≡R）	1 0 0 1 1 1 1 1	(*I)
26L	（P∧Q）⇒R	1 0 1 1 1 1 1 1	(*6)	26R	P∧（Q⇒R）	1 0 1 1 0 0 0 0	
36L	（P∨Q）⇒R	1 0 1 0 1 0 1 1		36R	P∨（Q⇒R）	1 1 1 1 1 0 1 1	(*F)
46L	（P⊻Q）⇒R	1 1 1 0 1 0 1 1		46R	P⊻（Q⇒R）	0 1 0 0 1 0 1 1	
56L	（P≡Q）⇒R	1 0 1 1 1 1 1 0		56R	P≡（Q⇒R）	1 0 1 1 0 1 0 0	
66L	（P⇒Q）⇒R	1 0 1 1 1 0 1 0		66R	P⇒（Q⇒R）	1 0 1 1 1 1 1 1	=26L, (*J)
括弧を使わない場合、優先順位を考えてコンピュータが解釈する演算							
32	P∨Q∧R	P∨（Q∧R）	=32R	25	P∧Q≡R	（P∧Q）≡R	=25L
42	P⊻Q∧R	P⊻（Q∧R）	=42R	35	P∨Q≡R	（P∨Q）≡R	=35L
52	P≡Q∧R	P≡（Q∧R）	=52R	45	P⊻Q≡R	（P⊻Q）≡R	=45L
62	P⇒Q∧R	P⇒（Q∧R）	=62R	65	P⇒Q≡R	P⇒（Q≡R）	=65R
23	P∧Q∨R	（P∧Q）⊻R	=23L	26R	P∧Q⇒R	（P∧Q）⇒R	=26L
43	P⊻Q∨R	P⊻（Q∨R）	=43R	36R	P∨Q⇒R	（P∨Q）⇒R	=36L
53	P≡Q∨R	P≡（Q∨R）	=53R	46R	P⊻Q⇒R	（P⊻Q）⇒R	=46L
63	P⇒Q∨R	P⇒（Q∨R）	=63R	56R	P≡Q⇒R	（P≡Q）⇒R	=56L
24	P∧Q⊻R	（P∧Q）⊻R	=24L	22	P∧Q∧R	1 0 0 0 0 0 0 0	(*1) (*A)
34	P∨Q⊻R	（P∨Q）⊻R	=34L	33	P∨Q∨R	1 1 1 1 1 1 1 0	(*2) (*D)
54	P≡Q⊻R	P≡（Q⊻R）	=54R	44	P⊻Q⊻R	1 0 0 1 1 1 1 0	(*3)
64	P⇒Q⊻R	P⇒（Q⊻R）	=64R	55	P≡Q≡R	1 0 1 1 0 1 0 0	(*4)
				66	P⇒Q⇒R	（P⇒Q）⇒R	=66L

表の見方：

＊　この表は括弧の使い方違いを対応させるように、大きく縦の２欄で構成してあります。
＊　各欄左端の分類番号は、二桁の整数にＬまたはＲを付け、括弧を左側か右側を区別します。
＊　二桁の数字は、演算子の優先順位です。32L の意味は、演算子を∨、∧の順であることを示します。
＊　括弧を使わない書き方をするときは、コンパイラは、優先順位を考えて内部的に括弧を補います。
＊　その場合の分類番号は英字のＬ，Ｒを付けてありません。これを表 2.8 の下段に示し、それが対応する分類番号を備考欄に示しました（表の行数を詰めるため、左右の欄を使っています)。
＊

2.5.7　変数三個の演算に見られる論理法則

二変数の対を組み合わせた括弧単位の演算で表した表現を、分配率と言う論理法則にしたものがあります（表 2.9）。前の第 2.4.6 項では、否定形とを組み合わせると、２つの論理変数の演算で、演算子違いでも同じ結果が得られる場合を解説しました。したがって、表 2.9 は、否定形を組み合わせると、さらに多くの表し方が得られます。表 2.8 と表 2.9 とには、三つの変数をＰ，Ｑ，Ｒで扱い、否定の演算子演算させた場合を含めてありません。否定形を含めると、三つの変数の選択種類は 2×2×2＝8 通りありますので、３変数の演算式の種類は 48×8＝384 通りもあります。これらを全部網羅的に示すことはしません。その解決には、プログラミング言語で、論理変数を使った代数式を利用する方が便利です。この説明は次の第 2.6 節の証明方法で取り上げます。このときに記号論理学が威力を発揮します。

表 2.9　変数三個の論理法則

(*1)	結合律：	［(Ｐ∧Ｑ) ∧Ｒ］	＝	［Ｐ∧ (Ｑ∧Ｒ)］	＝	［Ｐ∧Ｑ∧Ｒ］
(*2)	結合律：	［(Ｐ∨Ｑ) ∨Ｒ］	＝	［Ｐ∨ (Ｑ∨Ｒ)］	＝	［Ｐ∨Ｑ∨Ｒ］
(*3)	結合律：	［(Ｐ⊻Ｑ) ⊻Ｒ］	＝	［Ｐ⊻ (Ｑ⊻Ｒ)］	＝	［Ｐ⊻Ｑ⊻Ｒ］
(*4)	結合律：	［(Ｐ≡Ｑ) ≡Ｒ］	＝	［Ｐ≡ (Ｑ≡Ｒ)］	＝	［Ｐ≡Ｑ≡Ｒ］
(*5)	結合律：	［(Ｐ⇒Ｑ) ∨Ｒ］	＝	［Ｐ⇒ (Ｑ∨Ｒ)］	＝	［Ｐ⇒Ｑ∨Ｒ］
(*6)	同値	［(Ｐ∧Ｑ) ⇒Ｒ］	＝	［Ｐ⇒ (Ｑ⇒Ｒ)］		
(*A)	分配律：	［Ｐ∧ (Ｑ∧Ｒ)］	＝	［(Ｐ∧Ｑ) ∧ (Ｐ∧Ｒ)］		
(*B)	分配律：	［Ｐ∧ (Ｑ∨Ｒ)］	＝	［(Ｐ∧Ｑ) ∨ (Ｐ∧Ｒ)］		
(*C)	分配律：	［Ｐ∨ (Ｑ∧Ｒ)］	＝	［(Ｐ∨Ｑ) ∧ (Ｐ∨Ｒ)］		
(*D)	分配律：	［Ｐ∨ (Ｑ∨Ｒ)］	＝	［(Ｐ∨Ｑ) ∨ (Ｐ∨Ｒ)］		
(*E)	分配律：	［Ｐ∨ (Ｑ≡Ｒ)］	＝	［(Ｐ∨Ｑ) ≡ (Ｐ∨Ｒ)］		
(*F)	分配律：	［Ｐ∨ (Ｑ⇒Ｒ)］	＝	［(Ｐ∨Ｑ) ⇒ (Ｐ∨Ｒ)］		
(*G)	分配律：	［Ｐ⇒ (Ｑ∧Ｒ)］	＝	［(Ｐ⇒Ｑ) ∧ (Ｐ⇒Ｒ)］		
(*H)	分配律：	［Ｐ⇒ (Ｑ∨Ｒ)］	＝	［(Ｐ⇒Ｑ) ∨ (Ｐ⇒Ｒ)］		
(*I)	分配率：	［Ｐ⇒ (Ｑ≡Ｒ)］	＝	［(Ｐ⇒Ｑ) ≡ (Ｐ⇒Ｒ)］		
(*J)	分配律：	［Ｐ⇒ (Ｑ⇒Ｒ)］	＝	［(Ｐ⇒Ｑ) ⇒ (Ｐ⇒Ｒ)］		
		［Ｐ⇒ (Ｑ⇒Ｒ)］	⇒	［(Ｐ⇒Ｑ) ⇒ (Ｐ⇒Ｒ)］		
		［Ｐ⇒ (Ｑ⇒Ｒ)］	≡	［(Ｐ⇒Ｑ) ⇒ (Ｐ⇒Ｒ)］		
	推移律：	［(Ｐ⇒Ｑ) ∧ (Ｑ⇒Ｒ)］⇒ (Ｐ⇒Ｒ)				……純粋仮言三段論法　（＊注）
		［［(Ｐ⇒Ｒ) ∧ (Ｑ⇒Ｒ)］∧ (Ｐ⊻Ｑ)］⇒Ｒ				……簡単構成的ディレンマ
		［［(Ｐ⇒Ｑ) ∧ (Ｐ⇒Ｒ)］∧ (~~Ｐ~~⊻~~Ｑ~~)］⇒~~Ｐ~~				……簡単破壊的ディレンマ

（＊注）三段論法は、第 2.5.8 項で説明します。

2.5.8　等しいことを確認する演算

二つの変数値が同じであることを調べる方法が幾つかあります。数値計算を考えるとき、「引き算をして結果が 0 になる」こと、また、変数が 0 でなければ、「割り算をして結果が１になること」を調べるのがそうです。この判定には、条件文「if-then-else」の中で、等しいことを調べる比較演算子を使います。コンピュータ言語の Visual Basic では、比較する２数が等しいことを調べる比較演算子にイコール記号＝を使います。これは代入文でも使うイコール記号と紛らわしいので、Ｃ言語では==を使います。記号論理学では、同値の演算子≡(Eqv)を使い、左右の変数を比較する演算をし、同じであれば真を返します。この演算が常に真であるとき、この式の表現が**恒真式**（トートロジー）であって、その幾つかは論理法則になっています。コンピュータのビット並びのデータで、異同を判定するときは、対応するビット間で排他的選言(exclusive Or; XOR、記号は or の記号に下線を引く⊻を使いました)の演算をさせて、ビット並びがすべて 0 になれば等しいと判定します。

2.5.9　二つの論理式の比較

二組の論理式が異なった表現であっても、演算結果が同じになるものがあります。文章表現では、二つ以上の命題の組み合わせは複合命題です。文章違いでも、実は同じ意味になる場合があって、こちらは、論理公式にすることがあります。この代表的なものが、前の 2.4.6 項 表 2.6 で紹介したド・モルガンの法則です。記号論理学では、演算子の記号を少なくして、例えば「≡、⇒、⊻」を「¬、∧、∨」の３種類を使って書き換えることをします。この紹介は、表 2.6 の備考欄に示しました。この変換式は、比較的頻繁に利用しますので、実用公式または論理法則にします。これらを次項で説明します。

2.5.10　幾つかの演算の公式

(1) 恒真式（トートロジー）

（Ｐ∨~~Ｐ~~），　（Ｐ≡Ｐ），　（Ｐ⇒Ｐ）

などの形になるものを恒真式と言い、代数計算では（$a \div a$）＝１　に相当します。この形が成分に現れたら、消去できます。

(2) 矛盾式

（Ｐ∧~~Ｐ~~）

この形は、矛盾式です。この形が最終的にできる論理式は矛盾式であり、論理式として実用的な意味はありません。

(3)　命題と記号の重複消去

二重否定律：　¬(¬Ｐ)≡Ｐ

同じ命題の連言：　Ｐ∧Ｐ≡Ｐ

同じ命題の選言：　Ｐ∨Ｐ≡Ｐ

(4)　分配律

代数計算の分配律の例は、「$a(b+c)=ab+ac$」です。これと似たものが、論理積∧と論理和∨を使う式でも成り立ちます（表 2.8、表 2.9 を参照)。

(5)　双対原理

双対原理とは、「或る連続連言式、あるいは連続選言式において、

①：式の中のすべての連言と選言とを交換し、

②：式のなかのすべての命題の肯定と否定とを交換するならば、

③：このようにして得られた論理式は、」最初の論理式の否定と対等である」

と言うものです。

連続連言式とは、例えば；　Ｐ∧Ｑ∧Ｒ…

連続選言式とは、例えば；　Ｐ∨Ｑ∨Ｒ…

双対原理は、上の場合、次のようになります；

¬(Ｐ∧Ｑ∧Ｒ)≡~~Ｐ~~∨~~Ｑ~~∨~~Ｒ~~

¬(Ｐ∨Ｑ∨Ｒ)≡　~~Ｐ~~∧~~Ｑ~~∧~~Ｒ~~

二つの命題に関する双対原理が、ド・モルガンの法則です（表 2-6 も参照)。

¬(Ｐ∧Ｑ)≡~~Ｐ~~∨~~Ｑ~~

¬(Ｐ∨Ｑ)≡~~Ｐ~~∧~~Ｑ~~

連続連言の連続選言式は、例えば次のような式です。これを、選言的標準形と言います；

(Ｐ∧Ｑ)∨(Ｒ∧Ｓ)∨(…∧…)∨…

連続選言の連続連言式は、例えば次のような式です。これを、連言的標準形と言います；

(Ｐ∨Ｑ)∧(Ｒ∨Ｓ)∧(…∨…)∧…

(6)　内含の式（Ｐ⇒Ｑ）を選言的標準形で表す-その 1

一つの公式は「(Ｐ⇒Ｑ)≡(~~Ｐ~~∨Ｑ)」があります。これを証明する一つの方法を示します。左辺の演算組み合わせは、表 2.3 に示すように(1, 0, 1, 1)の 4 通りです。これを(1, 0, 0, 0)、(0, 0, 1, 0)、(0, 0, 0, 1)の三つの演算結果の論理和で合成すると考えて、下の式を立てます。

(Ｐ⇒Ｑ)≡(Ｐ∧Ｑ)∨(~~Ｐ~~∧Ｑ)∨(~~Ｐ~~∧~~Ｑ~~)

右辺に分配律などを適用して変換していくと；

(Ｐ⇒Ｑ)≡(Ｐ∧Ｑ)∨[~~Ｐ~~∧(Ｑ∨~~Ｑ~~)]

≡(Ｐ∧Ｑ)∨~~Ｐ~~

≡(Ｐ∨~~Ｐ~~)∧ (Ｑ∨~~Ｐ~~)

≡(Ｑ∨~~Ｐ~~)

∴(Ｐ⇒Ｑ)≡(~~Ｐ~~∨Ｑ)　　……証明終り

この、式の変換の進め方が演繹です。最後に∴（ゆえに）で締めくくったことが証明です。表 2-3 、表 2.6、表 2.7 の、式番号 11 の結果を確認して下さい。

(7)　内含の式（P⇒Q）を選言的標準形で表す-その２

もう一つ別の証明の導き方があります。まず、（P⇒Q）の否定を選言的標準形で表します。この場合には、一つしか項がないので

¬(P⇒Q)≡(P∧~~Q~~)

このまま否定を取った表し方もあります。すなわち；

(P⇒Q)≡¬(P∧~~Q~~)

しかし、双対原理を使えば、標準の表し方が得られます；

∴(P⇒Q)≡(~~P~~∨Q)

(8)　その他

表 2.10　二命題の多項式で表されるその他の論理法則

記号式　(*1)	文章での表し方	備　考
P⇒(P∨Q)	トートロジー	(PQ)について対称
(P∧Q) ⇒P	トートロジー	(PQ)について対称
P⇒[Q⇒(P∧Q)]	トートロジー	(PQ)について対称
P⇒(Q⇒P)	トートロジー	
[(P⇒Q)∧(P⇒~~Q~~)] ⇒~~P~~	Pならば、Qでかつ~~Q~~でない。ゆえに~~P~~でない	背理法
[P∧(P∨Q))≡P	Qが何であっても、Pが残る。Qが吸収される。	吸収律
[P∨(P∧Q)]≡P	Qが何であっても、Pが残る。Qが吸収される。	吸収律
[(P∧Q)∨(P∧~~Q~~)]≡P	Qが何であっても、Pが残る。Qが吸収される。	吸収律
[(P∨Q)∧(P∨~~Q~~)]≡P	Qが何であっても、Pが残る。Qが吸収される。	吸収律
[(P⇒Q)∧P] ⇒Q	PならばQである。Pである。ゆえにQである。	肯定式(modus ponens)
[(P⇒Q)∧~~Q~~] ⇒~~P~~	PならばQである。Qでない。ゆえにPでない。	否定式(modus tollens)

2.5.11　三段論法

三段論法は、二つの前提から一つの結論を推論する方法です。基本的な形を論理式にすると、

「前提１」かつ「前提２」ならば「結論」

と表すことができます。この形は、すでに表 2.9 で、純粋仮言三段論法、などがそうです。表 2-10 の肯定式と否定式もそうです。特に純粋仮言三段論法の一般的な形は、次の４つの型があります。

	大前提　と　小前提　⇒　結論
Ⅰ	（Q⇒R）∧（P⇒Q）⇒（P⇒R）
Ⅱ	（R⇒Q）∧（P⇒Q）⇒（P⇒R）
Ⅲ	（Q⇒R）∧（Q⇒P）⇒（P⇒R）
Ⅳ	（R⇒Q）∧（Q⇒P）⇒（P⇒R）

上の４型で、P、Q、Rが否定命題の場合を含めるとそれぞれ 8 通りの組み合わせがあります。また、括弧 () 全体を否定する組み合わせが 8 通りあります。つまり、この種の論理式は、全部で 4×8×8＝256 通りもあって、必ずしもすべてが恒真式ではありません。

定言三段論法は、上の（P⇒Q）の単位の代りに主語を p、述語を q とする形の定言命題、（p，q）で置き換えたものです。逆の命題（q，p）の場合もあります。そこで、記号として、Sを小名辞（小概念）、Mを中名辞（中概念）または媒名辞（媒概念）Pを大名辞（大概念）に置き換えたものが、三段論法の型です；

格	大前提	小前提	結論
Ⅰ	（M，P）∧（S，M）⇒（S，P）		
Ⅱ	（P，M）∧（S，M）⇒（S，P）		
Ⅲ	（M，P）∧（M，S）⇒（S，P）		
Ⅳ	（P，M）∧（M，S）⇒（S，P）		

一つの定言命題には（A，E，I，O）の４通りの選択があります。{大前提、小前提、結論} の並びに、この選択の組み合わせを当てはめます。例えば（AEO）のような組み合わせを数え上げると、全部でやはり 256 通りの種類があります。しかし、妥当な組み合わせは、次の 24 個しかありません。

Ⅰ	AAA	AAI	EAE	EAO	AII	EIO
Ⅱ	EAE	EAO	AEE	AEO	EIO	AOO
Ⅲ	AAI	IAI	AII	EAO	OAO	EIO
Ⅳ	AAI	AEE	AEO	IAI	EAO	EIO

3.　演繹と証明の実践的方法

3.1　言葉の説明

3.1.1　証明法を理解する

何かの文の並びは、一つの文単位ごとに、その文末を「である・ではない」の形の、肯定文または否定文の「言い切り」文で繋ぎます。現代日本語は、その文体に「です・ます調」と「である調」を使い分けています。文学的な作文ではなく、実務で使う文は、読む側が正しく理解できるような説得の技術を必要とします。感情を含む言い方や文学的技巧を避け、淡々と客観的な事実を説明する文は、接続詞として「かつ・または」で繋ぎます。前提となる理由と、それから得られる結論とを必要とするときに「ならば」で繋ぎます。その結論が正しいことを説明することを証明と言います。証明に使う方法は、二つあります。下の項で、具体的な例として、数学で使っている証明法を説明します。

3.1.2　数学で使っている演繹と推論

証明法の第一は、式を順々に並べていって、最後に、「故に…である」で締めます。結論が一つしかないときの普通の方法です。式を合理的な順に並べて行く過程が演繹です。特殊な記号「∴」を「ゆえに」と読んで、それを頭に付けた式が結論であって、式全体を並べる方法が証明です。演繹は、説明の途中に、直接必要としない余分な式を使うことを、できるだけ省きます。順に説明する途中の式を省くと、証明の過程が分からなくなることがあります。これを避ける省略方法は、既知の公式、または参考文献の引用です。推論の用語を使うときは、説明の途中または結論に、直接必要としない余分な式などを含む意味があります。これは、式を書く人の選択で左右されます。数学の式を扱うとき、式の形を変えていく過程のうち、展開は演繹に当たり、一般的に理解するときは、括弧を含む式で、括弧を外した表現に直すことと考えることができます。この逆順の意味の用語は思い付きませんが、例えば、括弧に括った式に直すことです。括弧にまとめるときの式の形は、式を扱う人の判断も反映します。その過程は、推論の用語の方が適しています。それによって、式の意味が分かり易くなったり、別の性質を見つけたりします。その代表的なものが、二次式の因数分解です。しかし、必ず実用的な解ができるのではなく、できない場合（複素数の解など）もあります。結論が一つ（等根）ではなく、解が二通り得られるとき、その一つを選ぶには理由が必要です。

3.1.3　帰納法

証明に使う第二の方法は、結論を最初に言っておいて、その正しさを説明するときに「なぜならば」を表す記号「∵」を書いて、それに続けて式を並べます。結論だけを利用し、その結論が得られた過程の説明を省くと不親切になるときは、後から説明または証明を追加する形をとります。その順番を区別したいとき、後ろ向き推論と言います。演繹と言うときは、説明用の式が先行し、それから結論を言う素直な順ですので、前向き推論です。何かの表式をあらかじめ仮定しておいて、その仮定を正しいものとして演繹し、正しい結果が得られたならば、元の仮定が正しいとする証明方法を、帰納法と言います。その過程を帰納的推論と言います。帰納は、演繹と対にする用語です。

3.1.4　背理法

証明とは、正しい結果が得られることを説明する手続きです。数学では、式の左右を、イコール記号「＝」で表すことができれば、よい手続きを踏まえた証明であると判断します。論理式では、これが等値記号「≡」で繋ぐことができて、この式が恒真式（トートロジー）であることを示すことが証明です。イコール記号「＝」と等値記号「≡」とは、記号の左右が同じであることを示す記号ですが、演算子として使うことがあります。この使い分けで混乱することがあります。演繹の方法で得られた結論は明快ですので、公式として使うことがあります。有名な物理学の公式には、アインシュタインのエネルギーの数学公式「Ｅ＝ｍｃ^2」があります。歴史的に有名な証明問題に四色問題があります。この結論は、常識的に認められていますが、その合理的な証明方法が見つからなかったのでした。多くの事例を調べていって、一つでも例外が見つかれば、正しくないとします。また、一つも例外がないことを説明できれば正しさ証明になります。この証明方法が背理法です。これに使う方法に、論理学では対偶・対当・逆・裏の用語を使う関係式が応用されます。また、例外が一つも無い場合の説明には「すべて」の用語を使う論理式を使うことがありますが、これを考えるときに集合論の知識が必要です。

3.2　真偽値を使う演繹の計算方法

3.2.1　真偽値表を作成して演算を進める方法

命題を、Ｐ、Ｑ、Ｒのような記号で与えると抽象的ですので、命題の取り得る真偽値(１,0)の組み合わせから、直接に演算結果を計算する方法を示します。この方法は、すでに２章で表 2-3 、表 2-4 、表 2-7 で使っています。計算例題として、下の論理式の計算を取り上げます。

Ｆ＝（Ｐ∨Ｑ）∧（Ｐ⇒Ｑ）⇒Ｑ　　…式(3.1)

表 3.1　真偽値表にまとめた論理計算

手順番号	演　算	記　号　式	真偽値表	説　　明
①	入力条件	Ｐ	１１００	Ｐの入力値組み合わせ
②	入力条件	Ｑ	１０１０	Ｑの入力値組み合わせ
③	①∨②	（Ｐ∨Ｑ）	１１１０	括弧内論理和の計算
④	①⇒②	（Ｐ⇒Ｑ）	１０１１	括弧内内含の計算
⑤	③∧④	（Ｐ∨Ｑ）∧（Ｐ⇒Ｑ）	１０１０	括弧同士の論理積
⑥	⑤⇒②	Ｆ＝	１１１１	Ｑとの内含の計算
		結論として、Ｆは恒真式であることが分かりました		

3.2.2　公式を適用した演繹

与えられた論理式の中で、内含の演算があれば、これを否定、連言、選言で置き換え、選言的標準形または連言的標準形になるように式を変形します。そして、分配律や双対原理などの公式を使って、簡単にして行きます。連続選言の形で、矛盾の成分ができたなら、この部分は消去できます。トートロジーの成分ができれば、全体の式は恒真式と判定できます。連続連言の形で、トートロジーの成分ができたなら、この部分を消去できます。矛盾の成分ができたなら、全体の式は矛盾です。演算の例を下に示す。上の項で解説した式(3.1)は、内含の演算子⇒を使わないで、下のように変換することができます；

Ｆ≡［(Ｐ∨Ｑ）∧（~~Ｐ~~∨Ｑ)］⇒Ｑ　　…式(3.2)

3.2.3　推論の方法

幾つかの命題（複合命題も含む）Ａ、Ｂ、Ｃ…があって、これらの間の論理的な関係が論理式（複数の場合がある）で与えられているとします。これらの論理式を使って、元の命題の真偽の状態を決定する具体的な方法を説明します。このときの命題Ａ、Ｂ、Ｃ…は、肯定命題を標準の形と仮定するのが普通です。推論に使う基本的な関係は、連言・選言・内含の三つです。推論では、この関係が真になる条件を吟味していくのが普通です。しかし、逆に、偽になる条件を吟味して、その条件を除外していく方法も取られます。

(1) 連言（Ａ∧Ｂ）が真である条件は、ＡとＢとがともに真であるときに限ります。（Ａ∧Ｂ）が偽になる条件の方は、ＡとＢとの可能な選択が増えるので、推論では仮定として使います。

(2) 選言（Ａ∨Ｂ）では、ＡとＢとがともに偽であるときに限って選言が偽になる一意性があります。選言が真になる条件は、ＡかＢか、どちらか真であるか、と両方真であるかを二度吟味しなければなりません。

(3) 内含（Ａ⇒Ｂ）は、推論に利用するとき、最も間違えやすい規則です。内含は、Ａが真、Ｂが偽のときに限って偽になる。そのため、内含（Ａ⇒Ｂ）は連言（Ａ∧~~Ｂ~~）の否定と等しく、さらに、双対原理で選言（~~Ａ~~∨Ｂ）と等しくなります。Ａが偽であっても内含は真を返します。したがって、内含が真になる条件を正しく使うためには、Ａが真である条件が必要です。その時に限って、Ｂの真偽は、（Ａ⇒Ｂ）の真偽と同じ、と推論することができます。上の条件を正しく使う推論方法が、肯定式、否定式です（表 2-10 参照)。また、適用を間違える推論が、前件否定の虚偽と後件肯定の虚偽です。

3.2.4　演繹を言葉で説明する例題

命題Ｐ、Ｑ、Ｒ…にどのような真偽の組み合わせを仮定しても、論理式の結果が常に真になるときはトートロジー（恒真式）です。また、常に偽であるときは矛盾式です。ところが、一般の推論のときは、命題Ｐ、Ｑ、Ｒ…の真偽のパターンに、幾つかの可能性ができることがあります。例えば、命題Ｐは、真偽どちらとも言えないときです。これは、条件文が足りない場合です。また、条件文が多すぎると、例えば、命題Ｐが真の場合と偽の場合、両方の推論が必要になりますが、これがディレンマです。論理式は、ただ一通りの結論が得られるものが最善です。例題として、下の式を考えます。

Ａ≡（Ｐ∨Ｑ）∧（Ｐ⇒Ｑ）　を吟味する

① Ａが真になる条件－その１　（Ｐ∨Ｑ）が真になるための仮定は
② 　Ｐが真、Ｑが真と仮定
③ 　Ｐが真、Ｑが偽と仮定
④ 　Ｐが偽、Ｑが真と仮定
⑤ Ａが真になる条件－その２　（Ｐ⇒Ｑ）も真となる条件は
　②は成立、
　③は不成立、
　④が成立。よって；
⑥ 　②と④と二つの条件から、Ｐは真偽どちらでもよい。Ｑは真。

⑦ Ａが偽になる条件－その１　（Ｐ∨Ｑ）が偽になる場合
　Ｐ、Ｑどちらも偽
⑧ Ａが偽になる条件－その２　（Ｐ⇒Ｑ）が偽になる場合
　Ｐが真、Ｑが偽
⑨ ⑦と⑧から、Ｐは真偽どちらでもよく、Ｑは偽

結論として、論理式Ａの真偽は、Ｑの真偽に等しくなり、Ｐの真偽には無関係です。これは、Ｐについては吸収律です。そうすると

Ｆ≡［(Ｐ∨Ｑ）∧（Ｐ⇒Ｑ)］⇒Ｑ　の形を考えれば

Ｆ≡　Ｑ⇒Ｑ

となり、表 3.1 で計算した恒真式になります。

3.3　プログラミングを利用する演繹

3.3.1　論理変数を明示的に使うことはあまりない

プログラミング言語 Visual Basic を利用して論理解析に使う例は、数値の大小関係を判定するときが普通です。プログラミング言語に論理型の変数が定義できるとは言っても、実践的なプログラムでは殆ど表（おもて）に現れることはなくて、内部的に使われています。例えば；

```
If A=4 And B <9 Or C >= 3 Then ...
```

のような構文は、下のような論理式の集合と解釈されます。

```
If {((A=4) And (B<9)) or (C >= 3)} Then ...
```

さらに、これらの論理式を内部的に個別の論理変数に変えます。

```
If ((L1 And L2) Or L3) Then ...          括弧内を先に計算して処理を進めて、
If (L4 or L3) Then ...                    最後に
If (L5) Then ...
```

の形にして条件の True/False を得て制御に使います。この演算の( )の部分に論理式を判断して論理変数に変えているのですが、一般的には論理変数が介在していることに注意しません。この演算を組み立てるとき、実は、演算の順序、または優先順位の約束が必要です。括弧を使わない式の形から演算順序を組み立てるとき、コンパイラは、演算子の優先順位を考えて内部的に括弧を補った式に直します。

3.3.2　論理判断を伴う実行制御文の種類

論理変数の取り得る値は True/False の二つですので、これを判定してプログラムの流れを二つ選択（２分岐）します。三つ以上の選択肢がある場合も中身は２分岐を組み合わせるだけです。論理的判断を使う制御文は下のような種類があります。

```
If ...Then ...Else
For ...Next
Select Case
Do While ... Loop
While...Wend
```

3.3.3　ふだんから論理的な文章を書く素養が大切

制御文を巧みに使うことができるようになれば、プログラミングをかなり理解したことになります。そのために必要な教養が、この章で紹介した論理演算の知識です。とりわけ、ド・モルガンの法則は、文章表現では下のような言い換えの規則です。

「お金があって友達がいれば、映画に行く」≡「お金がないか友達もいなければ、映画に行かない」

「お金があるか友達がいれば、映画に行く≡「お金がないときも、友達もいなければ、映画に行かない」

上の文章表現で、(お金がある＝Ｐ)（友達がいる＝Ｑ）(映画に行く＝Ｒ）の記号に置き換え、助詞の「て」「も」を And の演算子∧、「か」を Or の演算子∨を使って論理式に直すと、ド・モルガンの法則が成り立ちます。

Ｒ＝(Ｐ∧Ｑ)の否定は、~~Ｒ~~＝(~~Ｐ~~∨~~Ｑ~~)

Ｒ＝(Ｐ∨Ｑ)の否定は、~~Ｒ~~＝(~~Ｐ~~∧~~Ｑ~~)

表 3.2　ド・モルガンの定理を応用して書き換えるプログラミングの例

元の表現方法	改良した方法

If (A>5) And (A<10) Then GoTo LabelB
　Call ProA
　GoTo LabelC
End If
GoTo LabelC
LabelB:
Call ProcB
LabelC:
……
	If (A<=5) Or (A>=10) Then
	　Call ProcB
	Else
	　Call ProcA
	End If
	……

4. 論理学の応用場面

4.1 自然言語処理の課題

4.1.1 研究の発生場面

自然言語処理（natural language processing：NLP）は、文書をコンピュータで扱う文字処理の発展として研究対象になった分野です。その文字処理の始まりは、プログラミング言語の設計と、その翻訳ソフト（コンパイラ：compiler）の開発です。歴史の古い COBOL は、その機能の説明に、English-like Structured Language とあります。日本語で言えば、英語風の構文を持ったプログラミング言語です。文書（テキスト）を書くための必須のソフトウエアは、テキストエディタ(text editor)です。その発展として、ビジネス文書の作成に利用する需要を反映して、マイコン(micro computer)を利用したワープロ（ワードプロセッサ：word processor）専用機の開発ブームが起こりました。欧米の一般企業は、機械式の英文タイプライタが必須の事務機械でした。それを利用する女性秘書は、タイピング間違いをしないように神経を使わなければなりませんでした。彼女たちが仕えるボスに対して、彼女たちは、英文用ワープロの購入を熱望し、購入しなければストライキも辞さない、と言う雰囲気まで生まれました。これは、大型計算機の開発の方を主な戦略としていた企業の予測には無かったのでした。マイクロプロセッサを応用したビジネス指向のワープロの開発は、結果的に事務処理を主題とするコンピュータの開発へと焦点が移り、パソコン(personal computer)の性能向上に伴って、パソコン利用の時代へと移ってきました。マイクロソフト社のソフトウエアは、秘書の利用を視野に入れて、office の名称で括ったソフトウエア製品を発売しているのは、この歴史を映しています。

4.1.2 日本語ワープロの開発の経緯

1970 年代までの日本の一般企業は、機械式の報文タイプライタが必須の事務機械でしたが、事務処理のコンピュータ化には最大の障害でした。日本語ワードプロの原点は、1978 年、漢字の印刷ができる東芝の JW-10 です。これが実用機に育った背景は、ハードウエアとしてドットマトリックスプリンタが開発されたことです。ソフトウエアの機能を支えたのは、仮名漢字変換です。それと関連して、自然言語処理が注目されるようになりました。この研究は、学問的に扱う態度と、応用を意識して技術を研究する態度との二つの面があります。人間が日常的に使っている言語は、長い歴史の流れの中で、自然発生的に正しい話し方と書きかの約束、つまり文法が決まってきました。しかし、実社会では、文法に忠実な言葉の使い方に束縛されない表現法が大勢を占めます。これを自然言語と言うようになりました。学問的な態度は、この自然言語の使い方にも、何かの法則性があるのだ、とする信念があります。学問としての言語学がそうです。実際の人間社会では、場面に応じた種々の変った書き方や話し方が使われ、多くの曖昧さを持ちます。この人同士の中間にコンピュータを介在させ、翻訳などに利用すると、必ずしも正しく情報が伝わらないことが起こります。その理由を、分析し研究するのが、自然言語処理の学問的態度です。この研究は、文法規則の例外が多いこともあって、焦点を定め難い面があります。これにデータベースの手法を応用し、それを発展させた知識ベースにまとめることで、人工知能の研究へと繋がるようになりました。コンピュータ側に立って言語学を見る分野をコンピュータ言語学（計算言語学：computational linguistics）と言います。

4.1.3 自動翻訳開発に起こる問題

英語は、単語の分かち書きをし、単語の並び順に規則があります。したがって、英単語を品詞単位で日本語の単語に当て、語順を調整すれば、かなりの程度で正しい意味が分かります。歴史的に見れば、漢文訓読法は、一種の中国語の翻訳法です。この場合には、漢字の単語をそのまま生かし、日本語の語順に合わせるように、レ点や返り点などを補います。英文から日本語への翻訳も、単語単位での翻訳に加えて、原理的には語順の入れ換えをします。不思議な言い方になることもありますが、意味は正しく理解できます。しかし、どの言語であっても表現の曖昧さと、単語の意味の多様さがありますので、とんでもない誤訳も起こります。人が文を読むときは、その人の属する社会や専門の知識を補って正しい意味を選択できます。しかし、コンピュータを介して自動翻訳をさせたいとするとき、膨大な知識ベースの助けを借りる必要があることが分かってきました。そうであっても、すべての文章の解釈を明快に解決することはできません。曖昧さ自体を学問的な方法で扱おうとすると、出口の無い迷路に入ってしまいます。自動翻訳を技術的に処理させることを考えるときは、妥協の道を探ります。その一つの方法は、元の文を自動翻訳に向くように作文、または添削しておくことです。

4.1.4　中間言語に英語を考える

中学・高校・大学と連なる基礎教育では、外国語の代表である英語を理解できることが重要な素養になってきました。その第一段階が英文和訳です。中学・高校では、基礎的な英語教育が行われています。そこで扱う英文は、比較的上品な、文法を正しく使った、実用的な意味を持たせた書き言葉です。文学作品を教材にすると、レトリック的な表現を含むようになります。情景描写などは、まだ客観的な文構造をしています。しかし、作者の感情を表した文、詩的な表現、さらには会話を引用した文は、意味の理解ができなくなることが多くなります。その典型的な例は、シェークスピアの劇場向けの作品です。物理的な単語の並びだけから意味を理解することが難しいので、その全体に解説が必要です。これには、幅広い知識が必要です。正しい言葉遣いの英会話の文が、英語教育の教材になる例は多くありません。このこともあって、日本での英語教育では、英会話、特に音声を媒介として英語を扱うことに混乱があります。英文を理解すること、つまり英文和訳の逆方向が、和文英訳です。このとき、正しい日本語の文を踏まえなければ、相手に理解できる良質の英文になりません。この双方向の翻訳に、コンピュータの助けを借りる自動翻訳が注目されるようになりました。世界には多くの言語の種類がありますので、個別に双方向の翻訳ソフトを計画するのではなく、或る中間言語を介して翻訳をさせれば、翻訳ソフトの数を抑えることができます。その中間言語に英語をます。ただし、そこで使う英語は、アメリカ英語・イギリス英語の区別を超越した、理想化した文構造を採用します。それに対応して、日本語の方も、正しい日本語を扱う必要があります。このとき、翻って、この報文が主題とした論理学に照らして、文章を扱うことの素養が重要になります。

4.1.5　話し言葉は音楽的な言い方をしている

紙に書かれた文書は、眼で見て理解していると思うでしょうが、実は、声に出して、それを耳で聞かなければ理解できません。小児用の書物は仮名だけを使いますが、分かち書きをしないと読むときに不自由します。漢字かな混じり文は、漢字がキーワードの役目をしますので、分かち書きをしなくても正しく読み分けができますが、いつもそうではありません。漢字は、音・訓二様の読みがありますので、複数の漢字並びは、熟語または品詞単位での切れ目に読み間違えが起こります。漢字の使い方に常用漢字の制限を適用して部分的に仮名に書き換えたり、別の漢字を当てたりすると、読みの混乱だけでなく意味の混乱が起こります。普通の文書を黙読している場合でも、それを頭の中では音に直しています。読みが分からないと、理解に必要な時間的な過程が中断します。我々が話し言葉を聞いている時、発話者の微妙な息遣い、強弱アクセント、高低アクセントで言い分けていることを助けにして、正確に理解しています。音楽の音譜は、それを明示的に文書化する方法です。音譜単独は音を切る記号ですが、音を連続させる記号にスラーとタイとを使い分けます。種々の長さの休止記号も使います。聞いて心地よく聞こえる話し方は、単語や文節の切れ目で、適度な間を持たせますので、音楽的です。

4.1.6　文書をコンピュータに発声させる

テキストデータをコンピュータに発声させることの研究は、英語の場合には古くから開発されてきました。アルファベットを使う欧米語は、音を表す文字を並べ、単語単位で分かち書きをします。言語ごとに微妙に発声が異なることに対応させた特別な文字、例えばドイツ語のウムラウト、もあります。この表音文字並びの文構造は、機械的な発声に向きます。日本語は厄介です。分かち書きをしないこと、漢字の読みに音・訓の別がそれぞれ複数あること、それ以外にも特別な読みもあること、単語の切れ目が分からないことでの読み間違えがあること、などです。日本語を正しく読ませるため、息継ぎや語の切れ目に句読点を使います。漢字熟語の知識も助けになります。これを辞書だけの情報ではなく、人工知能(AI)を利用する研究もあります。しかし、ここで発想を切り替えると、コンピュータが日本語テキストを正確に読み上げるためには、作文技術を工夫することも考えられます。句読点以外の区切り符号(delimiter)を使い分ける方法です。その一つは、息継ぎをする個所に半角のスペースを入れます。コンピュータに日本語テキストを発声させることは、眼の不自由な人が情報を利用するときの大きな助けになります。これを目的とするときは、人の方が論理的で分かり易い作文をすること、または作文された文をそのように直すことに、コンピュータを利用することを考えます。これは、文章校正のソフトと言うことができて、自動翻訳の前段階に使うプリプロセッサと考えることができます。ワードプロセッサには、英語ではスペルチェッカ、日本語では表記の揺れを検査するツールを使うことができるようになったことが、これに当たります。文章の構成が論理的に筋の通った内容にまとめるのは、あくまでも人の側の責任です。日本語のワードプロセッサの仮名漢字変換は、変換候補をコンピュータ側が提示し、選択を人間側で行います。つまり、コンピュータの知能は、字面だけの解析に留めます。

4.2　論理パズル

4.2.1　文を記号化する演習に役立つ

主語・述語が整った短い文単位があって、意味がまともであるか（真）、そうでないか（偽）の判断を保留しておきます。幾つかの文単位を「かつ、または、ならば～である」で繋いだ全体で、或る論旨を導くときの過程から、この文並びの結論が、真であるか偽であるかの判断をします。この結論が直ぐには分からないことがあります。それを問題とした遊びが論理パズルです。本来、文字並びを読んで、考えを巡らして結論を出す頭の体操ですが、形式論理学を応用して解く方法もあります。これは、文を式に置き換える方法を理解することに役立ちます。以下に示す例題は、下の参考文献からの引用です。

小野田博一、「新作論理パズル」、ブルーバックス B1061，1995，ISBN4-06-257061-0 C0240 0740E（0）

4.2.2　問題と解答（変数１個の例）

盗まれた手紙（上の参考文献の Q76 より）

これが盗まれた手紙なら、これは盗まれた手紙ではない。この前提から得られる結論は？

この問題は、「これは盗まれた手紙である」を一つの命題変数Ｐとし、(Ｐ⇒~~Ｐ~~)を言う文形です。日常の文形ではないので、論理パズルになります。これを、下の記号論理学の演算に直して演算します。

表 4.1　(Ｐ⇒~~Ｐ~~)の演算

手順番号	記号式と演算			演算結果説明
①	Ｐ	1	0	Ｐの条件
②	~~Ｐ~~	0	1	Ｐの否定
③	(Ｐ⇒~~Ｐ~~)	0	1	Ｐならば~~Ｐ~~である
④	~~Ｑ~~	0	1	Ｑの否定形
⑤	(③∧④)	0	0	(~~Ｐ~~∧~~Ｑ~~) と同値
⑥	~~Ｐ~~	0	0	Ｐの否定形
⑦	(③∧⑥)	0	0	Ｐの否定形と同値

表 4.1 において、Ｐの取り得る真偽値が(1,0)の二種ですので、これを２列の真偽値表に並べます。演算は各列ごとに縦に進めます。このときの演算則は、表 2.3 の論理演算則の表を利用します。演算結果は、⑤で２列とも偽(0)と判定します。これはＰの否定ですので、「これは盗まれた手紙ではない」と推論することが解です。

4.2.3　問題と解答（変数２個の例）

美少女であるための条件(前項文献の Q13 より)

「女の子は、聡明であれば美しい」と言う文を考えます。この結論は、以下のどれから導きだせるのでしょうか？

(a:) 女の子は、美しければ聡明である。

(b:) 女の子は、聡明でなければ美しくない。

(c:) 女の子は、美しくなければ聡明でない。

表 4.2　発言の論理式表現とその演算

	演算種別	記号式と演算	真偽値表				演算結果説明
論理変数を下のように決めます。 Ｐ：女の子は聡明である Ｑ：女の子は美しい 問題は（Ｐ⇒Ｑ）と等しくなる論理式はどれか？を求めることです。(a:)，(b:)，(c:)の演算結果から、トートロジーになる(c:),が正解です。この関係は**対偶**です。	命題変数	Ｐ	1	1	0	0	Ｐの条件
		Ｑ	1	0	1	0	Ｑの条件
		~~Ｐ~~	0	0	1	1	Ｐの否定
		~~Ｑ~~	0	1	0	1	Ｑの否定
	前提	(Ｐ⇒Ｑ)	1	0	1	1	
	(a:)	(Ｑ⇒Ｐ)	1	1	0	1	
		(Ｑ⇒Ｐ)⇒(Ｐ⇒Ｑ)	1	0	1	1	トートロジーではない
	(b:)	(~~Ｐ~~⇒~~Ｑ~~)	1	1	0	1	
		(~~Ｐ~~⇒~~Ｑ~~)⇒(Ｐ⇒Ｑ)	1	0	1	1	トートロジーではない
	(c:)	(~~Ｑ~~⇒~~Ｐ~~)	1	0	1	1	
		(~~Ｑ~~⇒~~Ｐ~~)⇒(Ｐ⇒Ｑ)	1	1	1	1	トートロジー

4.2.4　問題と解答（変数３個の例）

黒衣の三少女(前項文献の Q77 より)

(a:)女の子が３人います。３人の名は、彩乃・瑠奈・有紗です。それぞれ服装は異なり、１人は黒いドレス、１人は黒いブラウス、残る一人は黒いＴシャツを身につけています（順不同)。
この３人のうち、２人が下の発言をしました。これらのうち、一方の発言はホント（真）で、もう一方の発言はウソ（偽）でした。

(b:)黒いＴシャツの少女の発言：「黒いドレスを着ているのが綾乃なら黒いブラウスは瑠奈です。」そして黒いブラウスを着ているのは瑠奈ではありません。」

(c:)黒いブラウスの少女の発言：「黒いドレスを着ているのが綾乃なら私は瑠奈です。そして黒いドレスを着ているのは綾乃ではありません。」

さて、どの服の子がだれだったのでしょう？

解析までの手順を説明します。変数記号の約束と仮説の条件を立てます。

(d:)　「黒いドレスの女の子は綾乃である」とした命題変数をＰとします。

(e:)　「黒いブラウス女の子は瑠奈である」とした命題変数をＱとします。

(f:)　「黒いＴシャツ女の子は有紗である」とした命題変数変数をＲとします。

(g:)　Ｐ，Ｑ，Ｒの真偽値は未定です。それを二人の少女の発言から求め（推論し）ます。

(h:) 黒いＴシャツの少女の発言は、論理式で表すと、「(Ｐ⇒Ｑ) ∧ ~~Ｑ~~」の演算です。真偽の取り得る値の演算を表 4.3 で①～⑤の手順で示します。

(i:)一方、黒いブラウスの少女の発言は、語順は違いますが、論理式で表すと、(Ｐ⇒Ｑ)∧~~Ｐ~~　で演算します。これを同じく表 4.3 の下に並べました。なお、この結果が真ならＱが真である、と追加の主張しています。

表 4.3　発言の論理式表現とその演算

条件となる発言	手順番号	記号式と演算	真理値表				演算結果説明
黒いＴシャツの少女の発言 (Ｐ⇒Ｑ) ∧ ~~Ｑ~~	命題変数	Ｐ	1	1	0	0	Ｐの条件
		Ｑ	1	0	1	0	Ｑの条件
	①	(Ｐ⇒Ｑ)	1	0	1	1	ＰならばＱである
	②	~~Ｑ~~	0	1	0	1	Ｑの否定形
	③	(③∧④)	0	0	0	1	(~~Ｐ~~∧~~Ｑ~~) と同値
黒いブラウスの少女の発言 (Ｐ⇒Ｑ)∧~~Ｐ~~	④	~~Ｐ~~	0	0	1	1	Ｐの否定形
	⑤	(③∧⑥)	0	0	1	1	Ｐの否定と同値

(i:) 黒いＴシャツの少女の発言は、手順番号③に演算結果が得られています。その意味は、ＰでもＱでもないときに限って真、つまりＰ，Ｑが共に偽(0)のときに限ります。

(j:) 黒いブラウスの少女の発言では、演算結果が⑤です。Ｐが偽であれば成り立ち、Ｑには関係しません。そこで、Ｑも真であるとの主張が追加されていたのです。

(j:) ここから推論の組み立てに入ります。
黒いブラウスの少女の発言をホントとすると、Ｐが偽(0)、Ｑが真(1)です。この値を、黒いＴシャツの少女の発言に代入すると、結果は偽（ウソ）です。

(k:) Ｐが偽であることは、黒いドレスの少女は綾乃ではありません。これを、表 4.4 には△記号を入れました。Ｑは真です。これは瑠奈は黒いブラウスを付けていませんので、これも△です。そうすると、黒いドレスを着ている女の子は有紗と推論できます。これを、表 4.4 には○記号で示しました。結局、綾乃が黒いＴシャツを着ています。

表 4.4　発言を元にした推論の結果

	綾乃	瑠奈	有紗
Ｐ：黒いドレス	△	△	○
Ｑ：黒いブラウス		○	
Ｒ：黒いＴシャツ	○		

(l:) 上記までの説明には論理変数Ｒを含む演算を一度も使っていません。使う場面が無いためです。

4.2.5　問題と解答（変数３個の例）

アップルパイの論理(前項文献の Q23 より)
「論理的であるか詩人である者は、論理的であるかアップルパイ好きである」。
これは真実と言えるでしょうか？　真実と言う意味は、演算結果がトートロジーになることであって、元になる式が恒真式であることを意味します。

三つの論理変数をＰ，Ｑ，Ｒとしたとき、変数の取り得る真偽の組み合わせが８通りになりますので、真偽表で追いかける演算は８列を使います。

表 4.5　問題文を元にした推論の結果

	手順番号	記号式と演算	真偽値表								説明
条件	命題変数	Ｐ	1	1	1	1	0	0	0	0	論理的
		Ｑ	1	1	0	0	1	1	0	0	詩人
		Ｒ	1	0	1	0	1	0	1	0	アップルパイ好き
設問の演算	①	（Ｐ∨Ｑ）	1	1	0	0	0	0	0	0	
	②	（Ｐ∨Ｒ）	1	0	1	0	0	0	1	0	
	③	（Ｐ∨Ｑ）⇒（Ｐ∨Ｒ）	1	0	1	1	1	1	1	1	
	④	（Ｑ⇒Ｒ）	1	0	1	1	1	0	1	1	
	⑤	（（Ｐ∨Ｑ）⇒（Ｐ∨Ｒ））⇒（Ｑ⇒Ｒ）	1	1	1	1	1	0	1	1	

この真偽値表の演算結果は、第６列だけが偽(0)です。これは、連続連言式（~~Ｐ~~∧Ｑ∧~~Ｒ~~）の否定と同じですので、双対原理を使えば、連続選言式（Ｐ∨~~Ｑ~~∨Ｒ）と同値です。

問題の続き：上の結果を判断して、下の論理式の真偽を判定して下さい。なお、参考のため、問題の文を論理式で表したものを右端に書きました。

(a) 詩人はアップルパイ好き。　（Ｑ⇒Ｒ）
(b) アップル好きでないなら詩人ではない。　（~~Ｒ~~⇒~~Ｑ~~）
(c) 論理的でない詩人はアップルパイ好き。　（~~Ｐ~~∧Ｑ）⇒Ｒ
(d) 詩人でない論理的な人はアップルパイ好き。　（Ｐ∧~~Ｑ~~）⇒Ｒ
(e) アップルパイ好きでない論理的な人は詩人。　（Ｐ∧~~Ｒ~~）⇒Ｑ
(f) アップルパイ好きでない詩人は論理的。　（Ｑ∧~~Ｒ~~）⇒Ｐ
(g) 論理的でなく、かつアップルパイ好きでない人は、詩人。　（~~Ｐ~~∧~~Ｒ~~）⇒Ｑ
(h) 論理的でなく、かつアップルパイ好きでない人は、詩人ではない。　（~~Ｐ~~∧~~Ｒ~~）⇒~~Ｑ~~

８通りの設問の真偽は、「⑤ならば(a)である」「⑤ならば(b)である」…の演算を個別にして、トートロジーであれれば真であると判定します。個別の演算は、表 4.5 のように進めるのですが、表 4.6 は、途中経過の計算式を示すことを省いて、「式の真偽値表計算」に結果を示しました。設問の真偽値表の計算がトートロジーであれば、この設問はホント（真）です。

表 4.6　設問の真偽値の判定

設問	記号式と演算	式の真偽値表計算								設問の真偽値表の計算								判定
(a)	Ｑ⇒Ｒ	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	ウソ
(b)	~~Ｒ~~⇒~~Ｑ~~	1	0	1	1	1	0	1	1	1	0	1	1	1	1	1	1	ウソ
(c)	（~~Ｐ~~∧Ｑ）⇒Ｒ	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	トートロジーである
(d)	（Ｐ∧~~Ｑ~~）⇒Ｒ	1	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	ウソ
(e)	（Ｐ∧~~Ｒ~~）⇒Ｑ	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	ウソ
(f)	（Ｑ∧~~Ｒ~~）⇒Ｐ	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	トートロジーである
(g)	~~Ｐ~~∧~~Ｒ~~）⇒Ｑ	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	ウソ
(h)	（~~Ｐ~~∧~~Ｒ~~）⇒~~Ｑ~~	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	トートロジーである

4.2.6　問題と解答（変数４個の例）

海王星の魚（前項文献の Q34 より）
このほど、海王星の魚について、超高精度の天体望遠鏡を使った、以下のような調査結果が発表されました。
(1) 海王星には、肺のある魚・背びれのある魚・鱗（うろこ）のある魚・牙のある魚がいる。
(2) 肺か鱗がある魚は、背びれを持たない。
(3) 鱗がない魚には背びれがある。
(4) 肺か背びれがある魚は、鱗か牙のどちらか一方のみを持つ。
このとき、以下の命題の真偽を調べよ。
Ⅰ　背びれがある魚には牙がある。
Ⅱ　牙がある魚には肺はない。

前提となる命題は４つです。これを論理変数Ｐ，Ｑ，Ｒ，Ｓで表します。
Ｐ：肺がある
Ｑ：背びれがある
Ｒ：鱗がある
Ｓ：牙がある

命題Ⅰ，Ⅱの真偽を調べるには、文で言えば、「(2)，(3)，(4)が共に成り立つ（論理積でつなぐ）ならば、ⅠまたはⅡとの内含の結果で判定する」となります。例えばⅡを調べるときは、命題代記号式で書くと、下の式(4.1)ようになります。

$$(((P \vee R) \Rightarrow \overline{Q}) \wedge (\overline{R} \Rightarrow Q) \wedge ((P \vee Q) \Rightarrow (R \mid S))) \Rightarrow (S \Rightarrow \overline{P}) \qquad \cdots式(4.1)$$

ここで、記号「$\underline{\vee}$」は、排他的選言 XOR です。

Ｐ，Ｑ，Ｒ，Ｓの真偽の組み合わせは。2^4＝16 通りあります。これを真偽値表に構成するには、16 行必要です。幾らが込み入った計算表ですが、式(4.7)は、手計算でも手順を追いかけることができるように真偽値表の演算をまとめました。設問Ⅱの真偽の判断は、手順番号⑫と⑮から真と分かります。

表 4.7　命題の論理式表現とその演算

	手順番号	記号式と演算	Ｐ，Ｑ，Ｒの可能な組み合わせを考えた真理値表																演算結果説明
命題変数	変数記号	Ｐ	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	肺がある
		Ｑ	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	背びれがある
		Ｒ	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	鱗がある
		Ｓ	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	牙がある
前提の計算	①	（Ｐ∨Ｒ）	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	0	
	②	$\overline{Q}$	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	
	③	①⇒②	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	前提条件(2)
	④	$\overline{R}$	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
	⑤	（$\overline{R}$⇒Ｑ）	1	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0	前提条件(3)
	⑥	③∧⑤	0	0	0	0	1	1	0	0	0	0	1	1	1	1	0	0	
	⑦	（Ｐ∨Ｑ）	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	
	⑧	（Ｒ$\underline{\vee}$Ｓ）	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	
	⑨	⑦⇒⑧	0	1	1	0	0	1	1	0	0	1	1	0	1	1	1	1	前提条件(4)
	⑩	⑥∧⑨	0	0	0	0	0	1	0	0	0	0	1	0	1	1	0	0	前提のまとめ
仮説の検証	⑪	Ｑ⇒Ｓ	1	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1	仮説Ⅰ
	⑫	⑩⇒⑪	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	トートロジー
	⑬	$\overline{P}$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	
	⑭	（Ｓ⇒$\overline{P}$）	0	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	仮説Ⅱ
	⑮	⑩⇒⑫	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	トートロジー

4.2.7　パソコンを利用することの準備

命題変数の数が少なければ、論理思考で答えを求めることもできます。しかし、真理値表にまとめた結果を参照しなければ正しい推論はできません。さらに、前の第 4.2.6 項の命題変数は４個ですし、命題の演算式も長いので、手計算で真理値表を計算することは面倒です。折角、代数式の形で表すことが提案されていますので、この式を利用してパーソナルコンピュータ（パソコン）に真理値表の計算部分を肩代わりさせることを工夫します。これには、手持ちのパソコンに論理演算のできるプログラミング言語が組み込まれていることと、それを使いこなすプログラミング技術の勉強が必要です。1980 年代は、16 ビットのパソコンの全盛時代でした。BASIC 言語が使えることは、一つの標準でした。この第 4.2 節の始めに紹介した参考文献に載っている BASIC のプログラミングコードは、特に説明をしなくても、読者側で実行を試すことができました。1998 年、マイクロソフト社は、Windows の OS の下で動作する Visual Basic 6.0（VB6 と略記）を発売しました。このプログラミング言語は、パソコンでの標準ソフトの扱いではなくなったことと、それまでのプログラミングの習慣と大きく変ったことで、初心者(beginners)向きの言語と言うよりも、高級言語になってしまいました。VB6 で書いたプログラムを紹介しても、読者の側でそれを実行するためには、VB6 をパソコンにインストール（組み込み）し、余分に作業用のフォーム画面の設計と、入出力を制御するプログラミングを必要とするようになりました。この部分の解説には、かなり多くのページ数が必要ですので、別資料として公開することにしました。なお、Visual Basic は、VB6 以降、改訂が重ねられています。VB6 のソースコードを紹介はしますが、改訂版での利用には、幾らか書き換えが必要です。

4.2.8　ソースコードの準備

海王星の魚の問題（第 4.2.6 項）を解くプログラムは、式 4.1 を BASIC 言語で書き直し、制御文や入出力文を追加します。VB6 で実行させるときの、筆者のソースコードを下に例示します。

```
Public Sub CRT02()
    Dim PP, QQ, RR, SS, XX As Integer
    Dim P, Q, R, S, X As Boolean
    For PP = 1 To 0 Step -1
        P = CBool(PP)
        For QQ = 1 To 0 Step -1
            Q = CBool(QQ)
            For RR = 1 To 0 Step -1
                R = CBool(RR)
                For SS = 1 To 0 Step -1
                    S = CBool(SS)
                    X = (((P Or R) Imp (Not Q)) And ((Not R) Imp Q) And _
                         ((P Or Q) Imp (R Xor S))) Imp (S Imp (Not P))
                    Write6 CStr(X)
                Next
            Next
        Next
    Next
End Sub
```

1980 年代のパソコン用 BASIC 言語は、上のコードよりはずっと簡単ですし、実行も面倒な手続きを必要としませんでした。しかし、VB6 の高級化によって、変数の宣言などを厳密にしなければならなくなりました。また PRINT 文が無くなりましたので、それに代わるコマンド Write6 を筆者は自前で準備しました。これは、モニタ画面に擬似的なテキストエディタを作成するフォームモジュールをプログラミングしておいて、そこを文字の書き出しに使います。見てくれの良い真偽値表、例えば表 4.7 のように作成するプログラミングコードを工夫するのは大変です。表計算のできる EXCEL でプログラミングできれば、表作成には便利でしょうが、EXCEL ではデータ型に論理型がありませんし、論理演算子の Imp, Xor もありません。使い勝手のよい BASIC のインタプリタがユーティリティプログラムとして利用できる環境が欲しいところです。

4.3　日本語文の見直し

4.3.1　日本語の解読に必要な常識

論理学が応用される主要な場面は、正確に相手に意味を伝える実用文書の作成のときです。感覚的、感情的な表現を評価する文学的な作文や、自己の考え方を主張する意見陳述の作文は扱いません。言葉を正確に使うには、論理学的には主語・述語の対で構成した命題を単位とし、それらを接続詞などで繋いで、意味を補います。書かれた命題は、論理学的にはすべて（真）でなければなりません。明示的に真偽を表した文が省かれている場合があります。これは、今話題としている文の環境での常識、または事実として決まった知識です。文章を読む相手側にコンピュータを考え、それに理解させるには、これらの知識を補う知識ベースの利用が必要になります。この課題が人工知能ですが、この報文での扱いには含めません。幾つかの命題の並びで、何かの結論を主張する文は、前の第 4.3 節にあるような、パズル的になる作文をしません。しかし、説明に使われる文ではレトリックも組み込まれますので、尤もらしい論旨の進め方であっても、ウソになることがあります。そうなる主な原因は、用語の定義とその使い方を正確に決めておかないことにあります。これを教材にまとめたのが、この報文の最初に紹介した「実用文書のまとめ方」です。そこでは論理学の説明を省いてありますので、それを補う資料として表 4.8 を見て下さい。

4.3.2　日本語では名詞の定義が曖昧

日本語では、名詞とは、眼に見える具体的な物（もの）を区別する用語です。日本語では人の場合には者を使い、場面によっては名前を付けて区別します。しかし、漢字の「物」は、やや抽象的に広い概念を指します。例えば、人物・動物・植物のように、生き物にも使う熟語がそうです。事（こと）の方の名詞は、抽象的な概念を表します。和語にはこの種の用語が少ないので、漢語の助けを借ります。第１章で解説した用語の定義は、すべて抽象名詞を扱っていて、その説明をしました。普通名詞の説明は一つもありません。英語を含む外国語にあって日本に無い物は用語がありませんので、名詞を作らなければなりません。元の呼び方を利用する、例えばカタカナ用語を決めると同時に、それがどういう物かの説明が必要です。英語では、物と事とをひっくるめた、やや広い概念の名詞の一つに object があります。日本語で理解しようとするときは、物の方だけを考えますが、英語の環境では事も含むことまでの理解が難しいようです。英語の場合、物を表す名詞の分類として、固有名詞・普通名詞・集合名詞・物質名詞の区別をし、動詞の単複の言い方と関係します。日本語の命題で「猫は動物である」を英語で言う時には、猫も動物も集合名詞の概念持たせています。猫は動物の一種ですので、集合論的に言えば、猫は動物に属しています。これを記号式で表すと、「猫⊇動物」です。論理的は、名詞単独は命題ではなくて、「それは猫である」のように仮の主語を立て、述語に be 動詞相当の動詞「である」で繋いで、主語・述語の組みで命題とします。単純に「猫」と言うとは省略の言い方です。日本人にはそれでも理解できますが、文法的に言えば体言止めの表現であって、誤解される危険があります。猫は動物に含まれますので、記号式は「猫⇒動物」で表します。英語の cat は、冠詞の区別と（a, the, 使わない）、単複の組み合わせで、意味の限定ができます。定冠詞 the が付くと、既に話題として取り上げたと特定しますので、固有名詞扱いです。それ以外は、「猫という物」の定義で使うか、集合名詞と解釈します。論理記号⇒は、「猫ならば動物の一種である」と解釈するのですが、これは定義の意味で使うことになりますので、(if～then)の条件文ではなく、英語の be 動詞を使う定義文になります。

4.3.3　所有格を表す方法に注意する

日本語の「の」は便利な助詞です。階層構造を持つ名詞を「の」で繋ぐと、上位から下位の順に並びます。内含記号を使う「Ｐ⇒Ｑ⇒Ｒ」は、日本語では逆順の「ＲのＱのＰ」の表し方になります。上のパラグラフで使った「猫⇒動物」は、猫の定義文ですが、「動物の猫」と言うと、「動物」を修飾語に換えた使い方です。

4.3.4　論理用語に対応する日本語

表 4.8　日本語の名詞以外の品詞の使い方(例として見て下さい)

論理学の用語	日　本　語　で　の　表　し　方	備　　考
肯定 存在 部分肯定	…です、…である、…だ、(英語の be 動詞相当)、動詞の終止形一般 …い、…しい（形容詞の終止形)、…だ（形容動詞の終止形） …がある（無生物)、…がいる（生物)、存在する …だけ、…もいる、…もある、	「です・ます」など、言い回しの区別があります。
限量詞 代名詞	すべて（の)、なんでも、みな、というもの、つねに、いつも、或る、 任意の、だけ、少なくとも、ともに、…も、 もの、こと、その、おなじ、のような だれでも、だれも、どこも、どれも	数量の定義に使います。これらは集合論に関係します。
否定の限量詞	なにも…、まったく（全く）…、全然…、必ずしも…	全称否定命題
連言	および（及び)、かつ、さらに、したがって、そこで、そして、 そのうえ、それから、それだけでなく、それで、それなのに、 それゆえ、ただし、…と、なお、ならびに（並びに)、…の時、 ふたつとも、また、…し、…して、…（し）たり…で、…が、 ところで、だが、けれども、でも、ところが（逆態）しかし、だが、のに、 ながら、さりながら、がしかし（逆接）さて、つぎに、では、	連用中止形も連言とします。いわゆる「て」フォームも含みます。逆接、逆態も論理的には連言。「など」はやたらに使いません。
選言	あるいは、いずれか、か、かあるいは、または（又は)、 それとも、 もしくは（若しくは)（主格の「…も」は補助的に用いられる）	普通の日本語では、排反的な意味です。両立的とも言います。
内含 条件文の前件 条件文の後件	ならば、…（する）と、なら、ば、れば、たら、(の）場合（に)、 …のとき（同時性の場合には、「時」を使い、連言と同じ） ときだけ、の限り、…のみ、…だけは、だけである にかぎ（限）られる、にかぎ（限）る、にかぎ（限）らない …のとき、そのときにかぎ（限）って…、ときだけである だから、ゆえに	仮定法の「もし」を使いません。 双条件文
推論	ゆえに必ず、ゆえにおそらく、当然である、必然である、必ず、 かならず、みなす、そうだ、ようだ、らしい、う、よう 可能である、あり得る、 …ので、妥当である、だから、はず、ちがいない、しかない、 なければならない、ありえない、あき（明）らかである	必然的真 可能
理由	ために、ためには、だから、…（する）ためである、ので なぜなら、…からである、…からだ	
許容	よろしい、(し）てもよい、…てよい、可能である、できる れる、られる	
命令 禁止	（し）なければならない、すべきである、必要である （し）てはならない	
定義文	英語の be 動詞に相当する「です・である」式の文が、定義文。 ｛「単語」は、「区別」＋「概念」である｝　…または、 ｛「区別」＋「概念」が、「単語」である｝　…の形式がよい。 動詞を定義するとき以外に、「もの」や「こと」などの代名詞を使わないようにする。	定義文は双条件文にする。言いかえと混同しない
説明文	読む、書く、走る、歩くなど、動作動詞を使う普通の文は、説明文になる。	普通名詞は具体性が低い
比較文	比べる対象との類似点をあげるやり方（含意、対等）と、相違点をあげるやり方（逆、反対、矛盾）とがある。	…のような、…らしい；は修辞学の直喩
副詞、 形容動詞、 形容詞	少々、しばらく、すぐに、早く、十分に、大幅に、大きい、小さい、 非常に、間もなく、急激に、いろいろな、高い、古い、 大勢、至急、よく、たまに、……	意味に幅があり、二値論理に向きません。数量化して使います。
意見陳述	考えられる、思われる、好き、嫌い、心地よい、痛い、苦しい、眠い、 見える、聞こえる、甘い、辛い、臭う、	感覚に関する用語は主観的な判断です。

4.4　幾何モデリングの論理

4.4.1　立体図形の論理処理

コンピュータ支援設計(CAD: Computer Aided Design)では、これから製作する三次元の形状を平面的な設計図に描くことを省いて、立体的なモデル（ソリッドモデル：solid model）をコンピュータの中にデータとして作成しておいて、それをコンピュータ制御の工作機械に送って製作する方法（コンピュータ支援製作：CAM: Computer Aided Manufacturing）で作業を進めます。この設計段階では、コンピュータグラフィックスのツールを使って、立体形状の幾何モデルの素材を加工して行く過程をモニター上で観察しながらインタラクティブに（対話的な方法で）作業を進めます。その基本的な作業の一例を図 4.1 に示します。これは、材料として二つの六面体（仮にＡ，Ｂとします）を用意し、擬似的に和（Ａ＋Ｂ）、差（Ａ－Ｂ）、積（Ａ×Ｂ）の処理をして新しい立体形状に加工した結果を示したものです。面を使って切断するときは、その面を持った仮の立体を作って引き算の材料に当てます。幾何モデルがあれば、技術データの計算に広い応用ができます。例えば、改めて従来の設計図を描くこともできますが､図 4.1 のような透視図に作成し、モニターで観察することが好まれています。

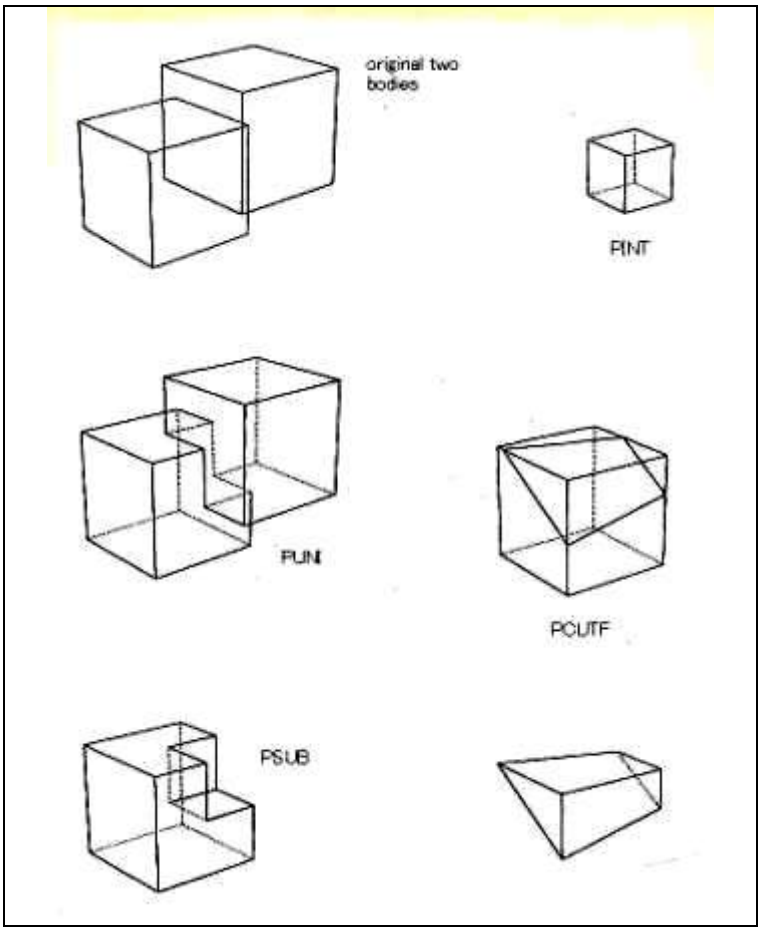


図 4.1　３Ｄ幾何モデリングで行わせる論理処理

4.4.2　幾何モデルの論理計算原理

立体的な幾何モデルは、集合論の見方をすると、二値の論理モデルです。モデルの内側は同じ材料で埋められていますので、実体のある、論理的に言えば真値(1)の集合です。モデルの外側は空集合です。論理的には偽(0)です。外形は、多角形の面で構成しますが、幾何学的には、中身が詰まっているとする**充実モデル**(solid model)と、多角形の紙状の面が連続して繋がっている、と考えた**張りぼてモデル**(surface model)とがあります。眼で見る限り、外見では区別ができません。見えているのは面の表であると約束します。紙細工(paper craft)と言う時は、紙の裏も見えるモデルです。紙の裏表の連続性が曖昧な、メビウスの輪のような構造も考えられるのですが、幾何モデリングでは扱いません。充実モデルの内部に空洞があるとき、空洞の内面を張りぼてモデルの扱いにします。空洞は、単独に外形として見れば、表と裏の面の約束が逆です。このことを集合論的に言うと、実体モデルの補集合です。擬似的に二つの充実モデルの干渉処理をプログラミングするときは、張りぼてモデルのデータ構造を使って面の交差を判定し､新しい辺を持った面に変形し､また､相手の内部に入る面を削除する処理をします。この幾何学的な計算処理は、判定しようとする座標点から任意の向きの半直線を引き、張りぼてモデルの面との交差をする点の個数を数えます。偶数個であれば外側、奇数個であれば内側です。空洞の場合は逆の関係です。干渉の計算を論理式で示すと、表 4.9 のように集合の論理演算で行われます。

表 4.9　多面体の干渉処理のコマンド(プログラム GEOMAP)

コマンド名	干渉処理の内容	概念式	論理式を使う表現
ＰＵＮＩ	二つの３Ｄモデルの和	Ａ＋Ｂ	Ａ or Ｂ
ＰＳＵＢ	モデルＡからＢと重なる部分を引く	Ａ－Ｂ	￢(~~Ａ~~ or Ｂ)
ＰＩＮＴ	モデルＡとＢとの共通部分を求める	Ａ×Ｂ	￢(~~Ａ~~ or ~~Ｂ~~)

4.4.3　平面図形の論理処理

コンピュータグラフィックスの作画技術は、平面図形の幾何学的処理の多様な応用で構成されています。これには、幾何モデルとして、切り紙モデルと地図モデルの二種類があります。切り紙の加工処理は、種々の形状の切り紙に切ることと、それを重ね合わせて連続した一つの形状にすることです。地図モデルは、切り紙の個別の形状の集合で、境界を考えた地図のような図形を作成することです。図 4.2 は、３Ｄモデルの干渉処理である和・差・積の演算に加え、重ね合わせて地図のように上描き技法を示したものです。

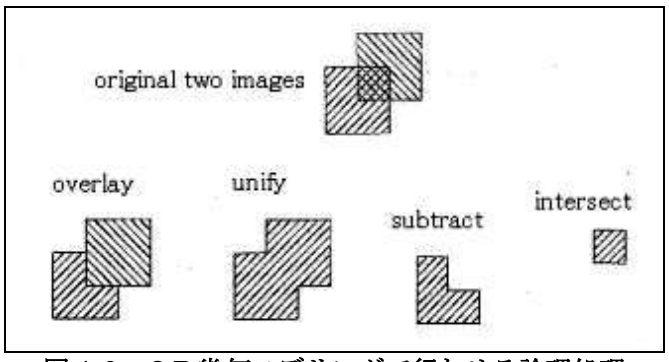


図 4.2　２Ｄ幾何モデリングで行わせる論理処理

二つの平面図形の論理処理は、第 2.4 節の図 2.1 ベン図が原理を説明しています。ただし、その中で、論理和と論理積の二つだけが実践的な意義を持ちます。図 4.2 は、二つの四変形の干渉結果を図示していますが、演算原理は表 4.2 の場合と同じ方法を使います。切り紙モデルは、単位としては一つの連続した紙を考え、材料の有る部分と無い部分とを真偽値の(1,0)で区別しますので、二値論理学の扱いをします。なお、穴の空いた切り紙も考えられ、この穴の領域は偽の扱いです。ところで、地図モデルは、切り紙モデル全体を複数の領域に区分けした、図形としては区分地図のようになります。この図形の特徴は、領域を区切る境界線にあります。幾何モデルとして扱うときは、領域単位の図形を多角形とします。この多角形の辺は、隣り合う領域の境界線である場合と、切り紙の外形線になる場合とがあって、外形線は空の領域との境界線です。空の領域には穴の場合があります。前項で説明した多面体モデルは、面の境界となる辺は、二次元の地図モデルでの境界線と同質の図形構成要素です。しかし、この辺は、必ず隣合う多角形の境界であって、空の領域との境界は有りません。多面体の展開図を作成するときは、この多面体を張りぼてモデルに考えて、どこかの辺に沿って切り出して展開する操作をします。この図から、幾つかの領域の境界を外して行くと、最後に一つの切り紙ができます。ここでの地図モデルは、論理学的には多値モデルです。四色問題は、四値モデルで地図の領域分けの問題でもあります。

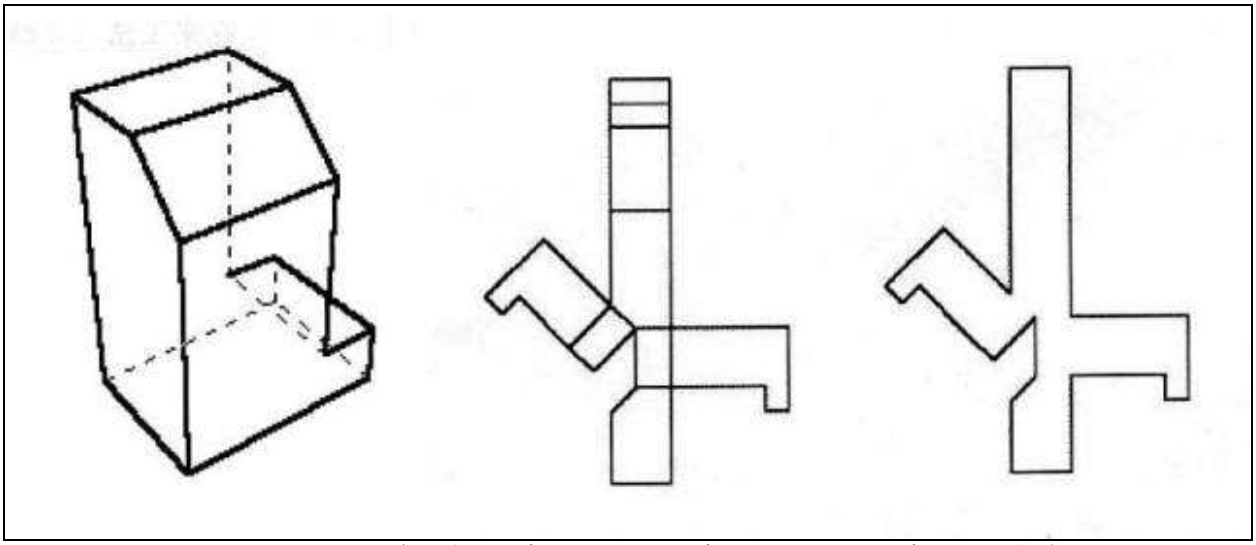


図 4.3　左から多面体モデル→地図モデル→切り紙モデルへの変換

99. 終わりに

論理学は、相手に正しく理解してもらう実践的な文章の話し方と書き方について、その規則を学問的に扱います。文学的な作文は、扱いません。その基本単位に、主語と述語とが揃った文(命題)を考えます。その文を細かく見るとき、名詞や動詞などの品詞の使い方を踏まえなければなりません。そうすると、言語学、さらには文法の知識が必要です。しかし、これらの品詞をどのように定義しておくかについては、触れていません。名詞、つまり、物や事を表す言葉は、或る一つの概念を持たせることが約束です。同じ名前を、場面によって異なった概念で使うのは、論理学では虚偽です（第 1.3.14 項参照)。これを避けるため、改まった文書では、そこで使っている用語の定義を説明する章を加えます。英語では glossary です。国語の辞書では、漢字の二字熟語を説明するとき、和語での言い換えをしています。一方、同じ概念であっても、別の言葉で言い換えることが増えてきました。日本語の環境では「和語に無い言葉に漢字の熟語を当てる；欧米語をカタカナ語にして使う」などに見られます。どれかに統一して利用するのが論理学的には正しいのですが、併用することもありますし、また、省略語に変更することや、勝手な造語を提案することさえあります。これは文学作品では大目に見ることもしますが、実用文書を作文するときには、修辞学的であって、論理学的には虚偽になります。

日本語は、英語から見れば曖昧な表現であるとの批判を受けます。しかし、日本語の環境に居る限り、曖昧さを意識することはありません。これは、言葉の背景に文化があるからです。言語学は欧米からの輸入学問ですので、欧米語の言葉の習慣に基づく、特に説明されない、常識的な規則を理解しておく必要があります。日本語に翻訳するときには何とか理解できることであっても、逆に日本語から欧米語に翻訳して発信する場面になると、相手側が正しく理解できなくて、曖昧である、との批判を受けることが起こります。この解決には、両方の言語に達者な人が作文の添削をするのが最善でしょうが、現状では日本人側が勉強することに多くの努力が必要です。言語習慣の違いの大きいものは、名詞の扱いがその一つです。英語を例にすると、名詞は単数・複数を厳格に区別し、冠詞の使い分け（a, the, なし）が意味を限定し、さらに動詞の活用とも関係を持ちます。集合論は、集合名詞の常識を踏まえています。

一方、動詞について言えば、日常頻繁に使う日本語の動詞、例えば「とる、みる、ひく」、英語では「take, see, get,など」は、耳で聞いても誤解されることはあまり起こりません。日本語では場面に応じて漢字１字を当てて、視覚的に意味を補う方法を使っています。日本語の「とる」は、「取・採・撮・執・摂・獲・盗・捕・録…」のような使い分けをしています。英語では、前置詞を補った句動詞(phrasal verb)で区別して使っています。これらの動詞は、言語ごとに使う習慣と場面が多様ですので、一対一に適切な翻訳を当てることが難しいのです。これらの問題を扱うのは、学問と言うよりは、話し方を含め、文章作文の技術と考えることができます。この冊子「易しくない論理学」では、この技術の説明を話題として取り上げません。英文和訳と和文英訳の場面で起こる種々の話題については、今までに興味深い著作が幾つも発売されています。これらの著作の要点をまとめて、作文技術の教材にすることは、現在計画中です。
