

4 命題論理

本章では、最も基本的な論理として命題論理を取りあげ、その形式的体系について考察する。この論理体系と前章の計算体系との関係については次章で述べる。

4.1 命題

命題 (proposition) とは何か？

この問に対する通常の答えは、「命題とは、真か偽かが確定している文」である。あるいは「命題とは、真であるかどうかを考えることができる文」とも言えるだろう。実は、この答えはいつでも正しいとは限らず、どのような論理体系を考えるかによって変わってくるのであるが、当面は (普通の論理だけを考えている限りは) 良いだろう。たとえば、「日本で一番高い山」は命題ではなく、「日本で一番高い山は筑波山である」は (偽であるような) 命題であり、「日本で一番高い山は富士山である」は (真であるような) 命題である。

我々が日常の推論で使う命題はこのような単純なものだけではない。

「A君は怒られなければ勉強しない」のように複合的な命題がある。このような複合的な表現も、構成要素である「A君が怒られる」「勉強する」という命題の真偽が決まれば全体の真偽が決まるので、命題であると言える。さらに、「A君が怒られなければ勉強しない」と「A君が勉強している」という2つの命題から「A君は怒られたはずだ」という推論をしたとする。これは「A君が怒られる(た)」「勉強する(した)」という命題の真偽にかかわらず正しい推論であると言えよう。論理学は、このように「個々の命題の真偽にかかわらず推論方法が正しいかどうかを議論する学問」ということができる。

本章では、命題に関する論理を扱う。

命題論理は、基本となる命題 (原始命題, 基本命題, 素命題, アトムなどという) が成立するかどうかに関係なく、いつでも成立する命題とは何か、それをどうやって推論するかについて考える。言い換えれば、基本命題の内部構造には立ち入らず、それらを組み合わせる記号 (論理記号) についての法則を考えるものである。

この観点に立って、命題を形式的に定義しよう。前述したように、基本命題の中身が何であるかは問わないので、それらを単に K_1, K_2, \dots, K_n という記号であらわす。このほかに、 \perp という特別な基本命題があるとす。これは「矛盾」 (どんな意味論でも「偽」となる命題) を表す*4。

これらをもとに、命題は以下のように帰納的定義により定義される。

定義 6 [命題]

- 基本命題 K_1, \dots, K_n は命題である。
- A, B が命題であるとき、 $\neg A, A \wedge B, A \vee B, A \supset B$ は命題である。

\neg は否定 (「でない」, not, negation), \wedge は論理積 (「かつ」, 連言, and, conjunction), \vee は論理和 (「または」, 選言, or, disjunction), \supset は含意 (「ならば」, imply, implication) を表す論理記号 (論理結合子) である。

複合的な命題の場合、括弧の付け方が問題になることがある。たとえば、 $A \wedge B \supset C$ という命題は、 $(A \wedge B) \supset C$ であるか $A \wedge (B \supset C)$ であるかわからない (曖昧さがある)。これでは厳密な考察の対象として

*4 いつでも「真」となる命題は $\perp \supset \perp$ として作ることができるので、そのような記号は導入しない。

は問題があるので、括弧の補い方が一意的になるように、以下の規則を定める。

- 異なる論理記号の間の結合の強さは、 \neg , \wedge , \vee , \supset の順番とする。(\neg が一番強い.)
- 同じ論理記号の間の結合については、 \wedge , \vee は左結合的であり、 \supset は右結合的である。

たとえば、 $\neg A \wedge B \wedge C \supset \neg D \supset E$ は、 $(((\neg A) \wedge B) \wedge C) \supset ((\neg D) \supset E)$ をあらわす。

特に注意すべきは、 \supset が右結合的なことである。なぜそうなっているかはこの時点ではなかなかわからないが、あとで、論理と計算の対応を付けたときには「なるほど」と思うであろう。

4.2 形式と意味

コインの裏表のように、物事には形式と意味(内容)がある。日常言語では「形式的」というのは否定的なニュアンスを伴い、「内容」というのは肯定的なニュアンスを伴う言葉であるが、論理学では必ずしもそうではない。

論理学における形式とは、形式的体系(formal system)のことであり、これは帰納的定義等によって厳密に定められた構造である。直感や意味を排して決められるので、どんな人が読んでも同じ定義である。従って、コンピュータによっても処理することが可能である。人間はつい意味の世界に頼ってしまうが、厳密な推論をしたり(たとえば、数学者による証明)、コンピュータによる処理をする(たとえば、定理の自動証明)ときには、形式的体系が不可欠である。

一方、意味の世界は、意味論(semantics)あるいはモデル論と呼ばれる。何らかの形式的体系を考える場合、通常は、それに先立って「何かあらわしたい世界」があるはずである。それを「意図した意味論」という。形式的体系を作る際の目標は、「形式的体系が、意図した意味論とちょうどぴったり対応する」ように作ることである。もちろん、ひとたび形式的体系を作ってしまったら、「意図した意味論」以外の意味論を持ち得ることは多い。たとえば、自然数の形式的体系は、我々が通常イメージしている自然数の世界を意図して作ったのであるが、「無限に大きな自然数」のようなものを含む「非標準的な意味論」を持つことが知られている。

形式的体系と意味論の関係を記述するのは以下の2つの性質である。

- 健全性: 形式的体系で証明(導出)できることは、どんな意味(モデル)のもとでも正しい。
- 完全性: どんな意味(モデル)のもとでも正しいことは、形式的体系で証明(導出)できる。

これら2つの性質が成立するとき、形式的体系と意味論は「ぴったり一致している」ということができる。

4.3 命題論理の意味論

命題論理の意味論は、通常、真理値表で与えられる。(真理値表による意味論しかないわけではない。また、真理値表の意味論とぴったり合っているのは古典論理だけである。この点は後に説明する。)

命題に対する真理値(truth value)とは、「真」(true, T)か「偽」(false, F)のいずれかの値のことである。形式化する前の命題は、真か偽かが定まっている文のことであったので、形式化した後でも(記号で表現した命題に対しても)、真であるか偽であるかを定めれば、意味が決まったことになる。

命題論理の真理値表の意味論は、以下のように与えられる。

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \supset B$
T	T	F	T	T	T
T	F	F	F	T	F
F	T	T	F	T	T
F	F	T	F	F	T

真理値表については、学類1年生向け授業『離散構造』の資料に載っているので詳しくはそちらを参照してほしい。

<http://logic.cs.tsukuba.ac.jp/~kam/discrete/> (第1章「命題と証明」)

命題 A に対応する真理値表の列が全部 T のとき、すなわち、 A に含まれる基本命題の真理値がどのような値であっても A の真理値が T であるとき、 A を恒真式 (tautology) と言う。恒真式とは「常に真である式」という意味であり、数学における「定理」に相当する。数学者がたくさんの美しい定理を追い求めているのと同様、論理学者は恒真式を追い求める。命題 A が恒真であることを、

$$\models A$$

と書くことがある。

命題 A が恒真であることを決定することは、工学的にも重要である。たとえば、プログラムやシステム設計にミスがないかどうかは、ある種の命題 (仕様を表す命題) が常に正しいかどうか、という問題に帰着される。真理値表意味論により、この問題が常に有限時間で判定可能であることがわかる。ただし、真理値表を本当に書くと、基本命題の数を k とするとき 2^k (2 の k 乗) の行数が必要になるため、大変に計算時間がかかる。次節で形式的体系を導入するが、これを使えば、(最悪の場合の計算時間は改善されないが、通常の場合には) 計算時間が大幅に短縮される。

4.4 形式的体系としての命題論理

一般的には、形式的体系を導入する意義として以下のものが考えられる。

- 人間の推論に近い形の推論を表現できる。
- コンピュータ上で推論を実行できる。(命題論理の場合は、前節で見たように、意味論をコンピュータでシミュレートすることが可能であったが、より複雑な論理ではそれは必ずしも可能ではない。)

これら以外に、この講義の題目は「計算」と「論理」の対応関係を調べることであり、このためには形式的体系は必須のものである。

さて、命題論理の形式的体系は、以下のように定義される。

まず、形式的な命題は前節で導入したように、基本命題と論理記号から構成されるものである。

判断 (judgement) は、 $\Gamma \vdash A$ という形を取る。ここで、 Γ は命題の有限列であり、たとえば、 $K_1 \wedge (K_2 \supset K_1), K_3$ といったものである。 Γ として空列も許す。 Γ のことを宣言と呼んだり、文脈と呼んだりする。

以下の規則は、判断を木の形に組み上げるためのものであり、推論規則と呼ばれる。(木の1つ1つのノードが判断になっている。)

$$\frac{(\text{ただし } A \in \Gamma \text{ のとき})}{\Gamma \vdash A} \text{ assume} \quad \frac{\Gamma \vdash \perp}{\Gamma \vdash A} \perp E \quad \frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} \neg I \quad \frac{\Gamma \vdash \neg A \quad \Gamma \vdash A}{\Gamma \vdash B} \neg E$$

$$\begin{array}{c}
\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge I \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \wedge EL \quad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B} \wedge ER \\
\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee IL \quad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee IR \quad \frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C} \vee E \\
\frac{\Gamma, A \vdash B}{\Gamma \vdash A \supset B} \supset I \quad \frac{\Gamma \vdash A \supset B \quad \Gamma \vdash A}{\Gamma \vdash B} \supset E \quad \frac{\Gamma \vdash \neg \neg A}{\Gamma \vdash A} \neg \neg E
\end{array}$$

それぞれの規則の横棒の右隣に規則名を書いた。(assume, $\wedge I$ など)

ここで注目してほしいのは、上記の規則群がほぼ対称形であることである。すなわち、 $\neg, \wedge, \vee, \supset$ の各記号に対応して、導入規則 (Introduction Rule), 除去規則 (Elimination Rule) がそろっていることである。ただし、例外は *assume*, $\perp E$, $\neg \neg E$ の3つの規則である。

上記の規則を有限回適用して $\Gamma \vdash A$ が推論できたときに「宣言 Γ のもとで命題 A は導出可能である (証明を持つ)」という。(Γ が空の列のとき、「命題 A は導出可能である」ということもある。)

例 8 命題論理の導出の例をあげる。

$$\begin{array}{c}
\frac{\overline{A \vdash A} \text{ assume}}{\vdash A \supset A} \supset I \\
\\
\frac{\frac{\overline{A \wedge B \vdash A \wedge B} \text{ assume}}{A \wedge B \vdash B} \wedge ER \quad \frac{\overline{A \wedge B \vdash A \wedge B} \text{ assume}}{A \wedge B \vdash A} \wedge EL}{\frac{A \wedge B \vdash B \wedge A}{\vdash (A \wedge B) \supset (B \wedge A)} \supset I} \wedge I \\
\\
\frac{\overline{A \vee B \vdash A \vee B} \text{ assume} \quad \frac{\overline{A \vee B, A \vdash A} \text{ assume}}{A \vee B, A \vdash B \vee A} \vee IR \quad \frac{\overline{A \vee B, B \vdash B} \text{ assume}}{A \vee B, B \vdash B \vee A} \vee IL}{\frac{A \vee B \vdash B \vee A}{\vdash (A \vee B) \supset (B \vee A)} \supset I} \vee E
\end{array}$$

4.5 古典論理と直観主義論理

前節で述べたのは命題論理の中でも「我々が日常使っている普通の論理」、すなわち古典論理 (classical logic) の体系である。古典命題論理の形式的体系は、真理値表による意味論とぴったり一致することが知られている。

定理 3 (古典命題論理の健全性と完全性) 前節の体系で $\vdash A$ が導出できることと真理値表の意味論において A が恒真であること (つまり、どんな真理値割当てのもとでも真であること) は同値である。

従って、この観点に立てば、古典論理は最も自然な論理であると言える。しかし、古典論理における推論は必ずしも自然でないことがある。たとえば、排中律と呼ばれる命題 $A \vee \neg A$ を古典論理により導出してみよう。すなわち、判断 $\vdash A \vee \neg A$ を導出する。この判断には仮定はなく、結論の一番外側の論理記号は \vee なので $\vee IL$ か $\vee IR$ 規則を最後に使って導出したと考えるのが自然である。しかし、実際にはどちらを使っても導出することはできない。(これは当然で、何の仮定もなく A や $\neg A$ が導出できては、どんな命題も導出できてしまう。)

実は、 $\vdash A \vee \neg A$ を導くために最後に使う規則は、 $\neg\neg E$ なのである。

$$\frac{\frac{\frac{\frac{\vdash \neg(A \vee \neg A)}{\vdash \neg(A \vee \neg A)} \text{assume} \quad \vdash \neg(A \vee \neg A) \vdash A \vee \neg A}{\vdash \neg(A \vee \neg A) \vdash \perp} \neg E}{\vdash \neg\neg(A \vee \neg A)} \neg E}{\vdash A \vee \neg A} \neg\neg E$$

このような導出により確かに排中律を導くことはできる。しかし、 $A \vee \neg A$ より複雑な命題を多用するこのような導出は果たして自然なものであろうか？

古典論理では、背理法「 $\neg A$ から矛盾を導ければ A と結論してよい」という推論法則も成立する。しかし、これも、意味から考えれば自然であっても、形式的体系においては、 A を推論するためにそれより複雑な $\neg A$ を仮定する、という意味で自然な推論方法ではないだろう。これらの「不自然な推論法則」を列挙すると以下のものがあげられる。

- $\neg\neg E$ 規則
- 二重否定除去: $(\neg\neg A) \supset A$
- 排中律: $A \vee \neg A$
- 背理法:

$$\frac{\Gamma, \neg A \vdash \perp}{\Gamma \vdash A}$$

- Peirce の法則: $((A \supset B) \supset A) \supset A$

実は、古典論理からこれらの法則を取り除いた論理が知られており、直観主義論理 (intuitionistic logic) と呼ばれる。言い換えれば、直観主義論理は、前に述べた古典論理の規則のうち $\neg\neg E$ 以外の 12 個から構成される論理体系である。

形式的体系としては、直観主義論理に基づいて、古典論理やそのほかの論理を見つめた方が、より自然であるので、本講義でもそのようにする。直観主義論理を基礎とした時、上記の 5 つの「不自然な」推論法則はすべて同値である。すなわち、直観主義論理に、上記の 5 つのうちのどれか 1 つを加えると古典論理が得られる。

4.6 直観主義論理と構成的解釈

前節で直観主義論理を導入したが、不整合な点を感じなかつただろうか？真理値表意味論と「ぴったり一致するもの」として古典論理の形式的体系を導入したのであるが、その古典論理の形式的体系に含まれる「不自然な推論」を排除して新たな体系を作ってしまった。この体系—直観主義論理の体系—は、古典論理の体系よりも弱いので、それに「ぴったり一致する」意味論は真理値表意味論ではないはずである。すなわち、以下の表の『?』を埋めるものは何であろうか？

論理	形式的体系	意味論
古典論理	$\neg\neg E$ 規則を含む 13 個	真理値表
直観主義論理	$\neg\neg E$ 規則以外の 12 個	?

ただし、真理値表の意味論も、直観主義論理の形式的体系に対応する意味論の1つである。なぜならば、直観主義論理の推論規則はすべて古典論理の推論規則でもあるので、直観主義論理で $\vdash A$ が導出できれば、それは古典論理でも導出できる。従って、命題 A は真理値表意味論で恒真式となり、健全性は成立している。

問題は、この逆が成立しないことである。真理値表意味論で恒真である命題の中には、直観主義論理では導出可能ではないものがある(たとえば、排中律)。つまり、完全性は成立していない。いま欲しいのは、健全性も完全性も成立する「ぴったり一致する意味論」である。

そのようなものの1つとして構成的解釈が知られている*5。ここでは、数学的に厳密な意味論を展開する余裕はないので、informalに構成的解釈を説明する。

定義7 [構成的解釈 (命題論理に限定したもの)]

- 命題 A が正しい、とは、命題 A の証拠が(具体的な計算によって)構成できることと定義する。証拠が構成できないときは命題 A は正しくない。
- 命題 $A \supset B$ の証拠とは、「命題 A の証拠をもらうと、命題 B の証拠を返す関数(プログラム)」のこと。
- 命題 $A \wedge B$ の証拠とは、「命題 A の証拠と命題 B の証拠の対(ペア)」のこと。
- 命題 $A \vee B$ の証拠とは、「命題 A と命題 B のどちらか成立しているか」をあらわす1bitの情報と、命題 A の証拠の対、もしくは、命題 B の証拠の対のこと。
- 命題 \perp の証拠は存在しない。
- 命題 $\neg A$ の証拠とは、 $A \supset \perp$ の証拠のこと。

なお、「 p が $\neg A$ の証拠である」ことを言いかえると、「 p が、 A の証拠をもらって \perp の証拠を返す関数」ということになるが、 \perp の証拠はないのだから、これは、「 A の証拠が存在しない」と同値である。この場合、 p はどんな関数でもよい。

上記の構成的解釈の定義は、「関数(プログラム)」としてどのようなものが許されるかに依存しているが、ここでは、本講義の最初で述べた「計算可能関数」とする。つまり、Turing機械、型のないラムダ計算、帰納的関数等のいずれかの方法で定義可能な関数のことである。

構成的解釈の例:

- $A \supset A$ の構成的解釈: その証拠は「 A の証拠をもらって、 A の証拠を返す関数」である。そのような関数は実際に $\lambda x.x$ として作ることができるので、 $A \supset A$ は正しい。
- $(A \wedge B) \supset (B \wedge A)$ の構成的解釈: その証拠は「 A の証拠と B の証拠の対をもらって、 B の証拠と A の証拠の対を返す関数」である。そのような関数は実際にプログラムとして記述できる(対の左右をひっくり返すプログラム)ので、 $(A \wedge B) \supset (B \wedge A)$ 正しい。
- $(\neg\neg A) \supset A$ の構成的解釈: まず、「 p が $\neg\neg A$ の証拠である」ということを考えると、これは、「 $\neg A$ の証拠が存在しない」ということである(p はどんな関数でもよい)。また、「 q が $\neg A$ の証拠である」ということは、「 A の証拠が存在しない」ということである(q はどんな関数でもよい)。これらから、「 p が $\neg\neg A$ の証拠である」とは、「 A の証拠が存在しないことはない」ということである。これは「 A の証拠が存在する」ということである。*6ここで p はどんな関数でもよいので、 p には何の情報もはっていないことがわかる。

*5 話の都合上、このような順番で導入したが、歴史的には、構成的解釈や「直観主義」という哲学的な考え方が先にあり、それに対応する体系の1つとして直観主義論理体系が発生したのである。

*6 注意深い人は、この推論で、 $\neg\neg E$ 規則を使っていることに気付くだろう。

最初にもどって $(\neg\neg A) \supset A$ が構成的解釈で正しいということは、「 $(\neg\neg A)$ の証拠をもらって A の証拠を返す関数」が存在するということである。「 $(\neg\neg A)$ の証拠」があるとしたら、上で一生懸命やった推論により、「 A の証拠は存在する」ことがわかるが、その証拠がどのようなものであるかは、さっぱりわからない。 $(\neg\neg A)$ の証拠である p 自体には何の情報もはいつていなかったため、 p を使ってもしょうがない。）

というわけで、 $\neg\neg A$ の証拠をもらって A の証拠を返す関数というのは、作れそうもない。 $(A$ が正しいからといって、その証拠はいつでも自動的に生成できるとは限らない。) すなわち、 $(\neg\neg A) \supset A$ は、構成的解釈のもとでは、正しくなさそうである。

- $A \vee \neg A$ の構成的解釈: その証拠は「 A が正しいという情報」と「 A の証拠」の対であるか、「 A が正しくないという情報」と「 $\neg A$ の証拠」の対である。しかし、そのような情報をあらゆる A に対して作ることができるとは思えない。(命題 A だけを見て、 A が正しいかどうかを判定するプログラムを書く必要がある。)

以上のような構成的解釈のもとで、 $\neg\neg E$ 規則を除く 12 個の規則は全て、「正しい判断から正しい判断のみを導く」ということが証明できる。すなわち、直観主義論理の体系は、構成的解釈に関して健全性を満たす。また、 $(\neg\neg A) \supset A$ や $A \vee \neg A$ などは、構成的解釈のもとで正しくなさそうであることがわかる。^{*7}

構成的解釈は、実は、完全性も満たしている、ということが知られている。すなわち、本節の冒頭の表における『?』は構成的解釈である。構成的解釈は、後に、型付きラムダ計算と直観主義論理の対応を考える際にも重要な役割を果たす。

4.7 古典的な存在証明と構成的な存在証明

ここまで、古典命題論理と直観主義命題論理の差について、形式的体系の観点と、意味論の観点から議論してきた。ここでは、具体的な例題として有名な定理をあげる。

定理: 無理数 p と q で、 p^q が有理数となるようなものが存在する。

この定理は以下のように証明できる。

証明: $R = \sqrt{2}^{\sqrt{2}}$ とおく。これは、有理数であるか無理数であるか、どちらである。そこで、場合分けする。

Case-1: R が有理数のとき、 $p = \sqrt{2}$, $q = \sqrt{2}$ と置けば、定理の条件を満たしている。

Case-2: R が無理数のとき、 $p = R$, $q = \sqrt{2}$ と置けば、 $p^q = R^{\sqrt{2}} = \sqrt{2}^{\sqrt{2} \times \sqrt{2}} + \sqrt{2}^2 = 2$ となり、定理の条件を満たしている。(証明終わり)

この証明は数学の証明としては完璧である。しかし、この証明をいくら眺めても、具体的な p の値は計算できない。なぜなら、Case-1 と Case-2 のどちらになるかがわからないからである。たとえば、 p が 1.5 より大きいかわかりか小さいかわからないことすらわからない。せっかく証明を与えたのに、これでは、あまり役に立つとは言えない。すなわち、計算機科学の観点からすると、有意義な証明でない。

このことの原因を分析する。上の証明は、最初に「 $(R$ が有理数) \vee (R が無理数)」ということを使った。これは、「 $(R$ が有理数) \vee \neg (R が有理数)」と言いかえるとわかるように排中律 $A \vee \neg A$ の一種である。すなわち、上の証明は、直観主義論理では成立せず、古典論理における証明となっている。

実は、以下のことが成立する。

^{*7} ここで言えるのは、「正しくないとされる」というだけであり、本当に正しくないことを示すためには、「プログラムによって計算できる関数とはなにか」ということを知らなければならない。ここでは計算可能関数の議論に深入りしない。

- 古典論理の方が、直観主義論理よりも証明できるものは多い。($A \vee \neg A$ や $\neg\neg A \supset A$ などが証明できる.)
- 直観主義論理で証明できたものは、構成的解釈を満たすので、計算にとって有用な情報 (存在定理の場合は、その存在するものを具体的に計算する情報) を含んでいる。

従って、もし、上記の定理を直観主義論理で証明していれば、「... となる p と q が存在する」という形の定理からは、必ず、 p と q を具体的に計算する関数 (プログラム) が得られたはずである。面倒だからといって手を抜いて $A \vee \neg A$ を使ってしまったので、情報量の少ない古典論理の証明しか得られなかった、ということになる。

このように、「何かを具体的に求めたい」と思うなら、直観主義論理を使うべきである。一方、「具体的な計算は良いので、正しいかどうかだけを知りたい」と思うなら、古典論理を使った方が簡単である。

4.8 導出の簡約 (計算)

ラムダ計算などの計算体系は「計算」という本質的に動的なものを記述する体系である。では、論理体系に動的な概念はあるだろうか? その答えは YES であり、導出自身を変形していく計算について考える。

まず、導出の無駄という概念を考えよう。

$$\frac{\frac{\frac{D}{\vdots} \quad \frac{E}{\vdots}}{\Gamma \vdash A \quad \Gamma \vdash B} \wedge I}{\Gamma \vdash A \wedge B} \wedge EL}{\Gamma \vdash A}$$

このような導出は、その部分導出である D の部分で既に $\Gamma \vdash A$ という結論が出ているので、最後の 2 回の推論は無駄である。すなわち、上記の導出は、結論となる判断を保ったまま、以下の (より簡単な) 導出に変形することができる。

$$\frac{\frac{\frac{D}{\vdots} \quad \frac{E}{\vdots}}{\Gamma \vdash A \quad \Gamma \vdash B} \wedge I}{\Gamma \vdash A \wedge B} \wedge EL}{\Gamma \vdash A} \sim \frac{D}{\vdots} \quad \Gamma \vdash A$$

これは、 $\wedge I$ の直後に $\wedge EL$ を適用した、という形を除去したという変形である。同様に、 $\wedge I$ の直後に $\wedge ER$ を適用した、という形を除去する変形が考えられる。

$$\frac{\frac{\frac{D}{\vdots} \quad \frac{E}{\vdots}}{\Gamma, A \vdash B} \supset I}{\Gamma \vdash A \supset B} \supset E}{\Gamma \vdash B} \supset E$$

この導出は、 D の部分で B を結論とする導出ができているので、それに変形できそうである。しかし、いきなり D に置きかえてしまえば、 Γ だった仮定が Γ, A になってしまい、(仮定が増えるということは、全体としては主張が弱くなるので) まずい。しかし、 A の証明は既に E の部分でできているので、これを張り合わせ

ればよさそうである。張り合わせる先は \mathcal{D} の中で仮定 A を実際に使っているところ、すなわち $\Gamma, A, \Delta \vdash A$ の形の assume 規則である。ここでは、assume 規則が 2 回使われているとした場合の例をあげる。

$$\frac{\frac{\overline{\Gamma, A, \Delta_1 \vdash A} \text{ as} \quad \overline{\Gamma, A, \Delta_2 \vdash A} \text{ as}}{\Gamma, A \vdash B} \supset I \quad \frac{\Gamma \vdash A}{\Gamma \vdash B} \supset E}{\Gamma \vdash B} \quad \sim \quad \frac{\frac{\begin{array}{c} \mathcal{E} \\ \vdots \\ \Gamma, \Delta_1 \vdash A \end{array} \quad \frac{\begin{array}{c} \mathcal{E} \\ \vdots \\ \Gamma, \Delta_2 \vdash A \end{array}}{\Gamma \vdash B} \mathcal{D}}{\Gamma \vdash B}}$$

しかしこれでもまだ問題がある。 \mathcal{E} はもともと $\Gamma \vdash A$ を結論とする導出だったのに、変形後は $\Gamma, \Delta_i \vdash A$ 等を結論とする導出になってしまっていて、うまく張り合わせるができない。また、 \mathcal{D} の仮定も少し違っている。そこで、 \mathcal{D} と \mathcal{E} に以下の変形をしたものを \mathcal{D}' と \mathcal{E}'_i とする。

- \mathcal{D} に含まれる仮定列から A を取り除いたものを \mathcal{D}' とする。
- \mathcal{E} に含まれる仮定列に Δ_i を付け加えたものを \mathcal{E}'_i とする。

このようにして以下の変形を得る。

$$\frac{\frac{\overline{\Gamma, A, \Delta_1 \vdash A} \text{ as} \quad \overline{\Gamma, A, \Delta_2 \vdash A} \text{ as}}{\Gamma, A \vdash B} \supset I \quad \frac{\Gamma \vdash A}{\Gamma \vdash B} \supset E}{\Gamma \vdash B} \quad \sim \quad \frac{\frac{\begin{array}{c} \mathcal{E}'_1 \\ \vdots \\ \Gamma, \Delta_1 \vdash A \end{array} \quad \frac{\begin{array}{c} \mathcal{E}'_2 \\ \vdots \\ \Gamma, \Delta_2 \vdash A \end{array}}{\Gamma \vdash B} \mathcal{D}'}{\Gamma \vdash B}}$$

この変形では、 \mathcal{E} を 2 つコピーしているのだから、必ずしも導出の大きさを小さくする変形ではないが、 $\supset I$ - $\supset E$ 規則が連続する部分の無駄を省く効果はある。

次は、 \vee の場合である。

$$\frac{\frac{\frac{\mathcal{D}}{\Gamma \vdash A} \quad \overline{\Gamma, A, \Delta_i} \text{ as}}{\Gamma \vdash A \vee B} \vee IL \quad \frac{\mathcal{F}}{\Gamma, B \vdash C} \vee E}{\Gamma \vdash C} \quad \sim \quad \frac{\frac{\begin{array}{c} \mathcal{D}'_i \\ \vdots \\ \Gamma, \Delta_i \end{array} \quad \mathcal{E}' \quad \mathcal{F}}{\Gamma \vdash C} \vee E}{\Gamma \vdash C}}$$

この変形においても、 \mathcal{E} の中で仮定 A を使う assume ルールを $i = 1, 2, \dots$ して、 \mathcal{E}' は \mathcal{E} に含まれる仮定列から A を取り除いた導出、 \mathcal{D}'_i は \mathcal{D} に含まれる仮定列に Δ_i を付け加えた導出とする。

同様に、 $\vee IR$ - $\vee E$ の変形も定義できる。また、 $\neg A$ は $A \supset \perp$ の省略形と見なすことにより、上記変形操作を適用することができる。

一般に、 $\supset, \wedge, \vee, \neg$ という論理記号に対する導入規則と除去規則が連続して現れている部分は、導出の中で

の「無駄」である。無駄のある導出に対して、上記のような操作によって無駄を除去する操作を導出に対する「計算」(簡約)と定めることができる。

ところで、通常人間が書くのは、無駄のない導出のみなので、上記の変形操作にどんな意味があるのか、こんなものを計算と呼んだところで極めて特殊な計算に過ぎない、と疑問に思うかもしれない。実はそうでない。

定理 4 (直観主義命題論理の導出の変形) 上記の変形操作を「計算」と見なすと、合流性、(強い)停止性を満たす。

この定理自身、驚くべき内容である。というのは、 \supset や \vee に関する変形操作で見たように、変形後の導出は変形前より小さくなる保証が全くないからである。(コピーをいくつも作成するので、しばしば、非常に大きくする効果をもつ)。

しかし、ここではこの定理の証明は置いておき、その意味を考えてみよう。この定理から、直観主義命題論理では、導出可能な判断に対して、無駄のない導出が必ず存在することがわかる。このことは、命題の導出において大きな意味を持っている。以下の導出が無駄のない導出であったとしよう。

$$\frac{\frac{\overline{J_3 - 1} \quad \overline{J_3 - 2}}{J_2 - 1} R_2 \quad \frac{\overline{J_3 - 3} \quad \overline{J_3 - 4}}{J_2 - 2} R_1}{J_1}$$

この導出のうち、結論の判断 J_1 から、一番左の道を下からたどってみよう。そこにあらわれる、判断と推論規則名の列を $J_1, R_1, J_2 - 1, R_2, J_3 - 1, R_3, J_4 - 1, \dots, R_n$ とする。この系列では、 R_k が除去規則(規則名に E がつく規則)で、 R_{k+1} が導入規則(規則名に I がつく規則)であるような状況が起きないことを示す。そのような状況が起きたとすると、 R_k は、 J_{k+1} の結論となる命題の主たる論理記号(一番外側の論理記号)に関する除去規則であり、 R_{k+1} は、その同じ論理記号に関する導入規則である。すなわち、この部分は「無駄」になっているので変形可能であるが、上記導出は無駄がないと仮定したので矛盾である。よって、上のような状況は起きない。すなわち、 R_1, R_2, R_3, \dots という規則名の系列を取りだすと、 R_1, \dots, R_m が導入規則、 R_{m+1}, \dots, R_{n-1} が除去規則、 R_n が assume 規則となっているはずである。ただし、 $m = 0$ (導入規則が一度も使われない)や、 $m = n - 1$ (除去規則が一度も使われない)ということもある。

この解析により、何らかの判断を導出する場合、主たる道筋では(つまり、導出の一番左側の道では)

- 導入規則を何回か使って結論をばらばらにする。
- 次に除去規則を何回か使って仮定に到達する。

という道筋だけを検討すればよいことがわかる。そのような道筋で導出できれば良いし、導出できなければ、他の道筋を検討することなく、「導出不能」ということがわかる*8。

これは定理の自動証明をする上で極めて重要な知見である。たとえば、 $\vdash A \supset B$ という形の判断を導出するという問題が与えられたとすると(つまり、それが導出可能であれば)、その最終規則は必ず $\supset I$ である導出が存在する。そのほかの導出もあるかもしれないが、考えなくてよい、ということである。

このことを精密化させると以下の定理が成立する*9。

*8 そんな重要な情報は演習をやる前に教えてほしかった、と思う人がいるかもしれないが、学問とはそんなものである。簡単にできる方法を最初に学んでしまうと、その方法がいかに偉大なものがわからない。

*9 本節で述べた「変形」の範囲では、一番左側の道についてしか上のような良い性質は言えない。以下の定理を証明するためには、変形操作をさらに増やして精密化させる必要があるが、ここでは省略する。

定理 5 ((直観主義命題論理の導出における) 部分命題の性質) $\Gamma \vdash A$ が導出可能なとき、 Γ と A の部分命題のみから構成される導出が存在する。

命題 A の部分命題とは、 A を構成する途中に出現する命題のことであり、たとえば、 $A \supset (B \wedge C)$ の部分命題は、 $A, B, C, B \wedge C, A \supset (B \wedge C)$ の 5 つである。上記の定理は、一般には「部分論理式の性質」とよばれる定理であり、何かの命題を証明したいときには、その命題の部分論理式のみを対象に導出を構成することを考えればよい、というものである。この定理が成立するとき、定理証明の探索の手間が劇的に減ることが想像できよう。直観主義論理は部分論理式の性質が成立するという意味で非常にたちのよい体系である。

一方、古典論理では、上記のような良い性質は成立しない。実際、 $\vdash A \vee \neg A$ に対しては、いきなり除去規則 $\neg\neg E$ を適用して、結論を $\vdash \neg\neg(A \vee \neg A)$ にしてから導入規則を使うような「不自然な」導出しか存在しないことを既に見てきた。この場合、 $\neg\neg(A \vee \neg A)$ は、 $A \vee \neg A$ の部分命題ではないので、上記定理の反例となっている。

付記: 導出に対する変形操作 (仮定列を増やしたり減らしたり) を厳密に述べるのはかなり面倒なので、本節では極めて informal に述べた。詳細は、論理学の本格的な教科書を参照されたい。ただし、次章で、導出に対する変形操作と本質的に同じ操作について説明する。