第11回補足資料

12月２１日

前回の復習から:

最後の２つの推論規則が加えられると、証明できる論理式の範囲がちょうど真理表の意味論におけるトートロジー(恒真）な論理式の範囲と一致します。

定理 3.1 (健全性 (soundness)) 証明可能な論理式はすべてトートロジーである．この性質を持つ論理体系を**健全である**という．よって命題論理の自然演繹体系は健全である．

定理 3.2 (完全性 (completeness)) トートロジーな論理式は，すべて証明可能である．この性質を持つ論理体系を**完全である**という．よって命題論理の自然演繹体系は完全である．

真理表による論理記号の意味論と推論規則による証明の関係　（基本教材p.25）

今回２つの推論規則を追加したことにより、「完全性定理」も成立します

定理 3.1 (健全性 (soundness)) 証明可能な論理式はすべてトートロジーである．この性質を持つ論理体系を**健全である**という．よって命題論理の自然演繹体系は健全である．

定理 3.2 (完全性 (completeness)) トートロジーな論理式は，すべて証明可能である．この性質を持つ論理体系を**完全である**という．よって命題論理の自然演繹体系は完全である．

- 健全性定理は、前提なしに（前提だった論理式があったとしてすべてカギカッコで前提なしの状態となって）証明可能な論理式は、真理表でトートロジーになる」ことを示しています。簡単な数学的帰納法で証明できます。実はその一部の議論は、真理表意味論から証明論に移るところでs目しています。最上級者の方はこの証明を完結するにはどうしたらよいか、考えてみてください。

定理 3.1 (健全性 (soundness)) 証明可能な論理式はすべてトートロジーである．この性質を持つ論理体系を**健全である**という．よって命題論理の自然演繹体系は健全である．

定理 3.2 (完全性 (completeness)) トートロジーな論理式は，すべて証明可能である．この性質を持つ論理体系を**完全である**という．よって命題論理の自然演繹体系は完全である．

完全性定理は「トートロジーであるどんな論理式も前提なしに証明できる」という、健全性定理の逆の形です。このこれらの２つの定理より、真理表のトートロジーと証明論の証明可能性の２つの概念の一致が示されます。完全性定理は、前回第１０回の補足資料で追加した２つの推論木鋪を加えて初めて証明できる定理です。

最上級者への注意：完全性定理の証明も数学的帰納法で示せます。ただし、健全性定理の証明は証明の長さによる数学的帰納法ですが、完全性定理の証明では論理式の複雑さ（論理式に現れる論理記号の数）に関する数学的帰納法となります。

証明はすこしだけ込み入っていますが、証明してみたい人がたら知らせてください。ヒントや参考文献をお知らせします。

完全性定理は「トートロジーであるどんな論理式も前提なしに証明できる」という、健全性定理の逆の形です。このこれらの２つの定理より、真理表のトートロジーと証明論の証明可能性の２つの概念の一致が示されます。完全性定理は、前回第１０回の補足資料で追加した２つの推論木鋪を加えて初めて証明できる定理です。

この２つの推論規則があって初めて、「ならば」と「否定」の真理表と証明可能性とが一致します。

下の１は上級者だけへの練習問題です。２は一般の履修生への問題です：

一部は既に補足資料で例示しましたが、
１．どんな論理式AとBとにつぃしても、
（A→B）→（¬A∨B）と（¬A∨B）→（A→B）の両方が（前提なしに）証明できること、

２．どんな論理式Aに対しても、A→¬¬Aとが¬¬A→Aの両方が（前提なしに）証明でキリコと、
を示してみてください。

特に二重否定消去規則を推論規則に含めない論理を直観主義論理と呼びます。

- 直観主義論理は計算論的観点や我々の認識力の観点により適合する論理と考えられます。一方で、二重否定消去規則を含めた、真理表のトートロジー概念と一致する論理は、古典論理と呼ばれ、我々の認識能力の観点や計算の観点を考えずに、世界の側で命題の真偽は決まっているという立場の論理と言えます。
- この違いについては、例えば円周率の命題に関して出した例も参照してください。あの例は直観主義論理側の議論として出されたものです。
- Aか¬Aかを確かめる計算や認識による検証法がなくてはA∨¬A

という論理式（「または」の文）は真偽判定以前に無意味である、と考える立場と言えます。

このページは最上級者へのページ

- 二重否定消去規則のない直観主義論理では、例えば

A∨Bが前提なしに証明されると、その証明の中の情報から、Aの方が証明できるのか、それともBの方が証明できるのかが分かります。つまり、A∨Bの証明からAの証明かBの証明を再構成できます。条件はもう少し弱められますが、ここでは簡単のために前提がないA∨Bの形の証明を考えます。「A∨Bの証明が直観主義論理で証明できれば、Aの証明を構成できるかBの証明を構成できる」ことは、基本教材の正規化定理を前提にすると証明できます。(直観主義論理のハロップ性と呼ぶ)

本学期は深く立ち入りませんが、命題論理言語の「∨」の述語論理言語への拡張に「∃xA」という形の表現（存在量化子と呼ばれる）があることは、すこしだけ、型付きλ計算の拡張の方向の一例として挙げました。その時触れた、仕様の∃証明からのプログラム抽出もこのことを意味しています。

型付きラムダ計算も基本部分は二重否定消去規則の入っていない論理証明に対応する部分です。

推論規則による直接的（操作的）意味論

- 二重否定消去規則は実際、とても例外的な形をしています。
- 他の論理記号（論理的結合子）は、導入規則（I－規則）と消去規則（E-規則）の対としてできますが、この二重否定消去規則だけは論理的結合子¬の対にさらに付け加わる形で出てきます。
- 真理表意味論の「否定」の（表示的）意味が先か、推論規則の「ならば」の（操作的）意味が先か、という問いかけを前にしましたが、後者について少し解説しておきます。

上級者向け「導入規則に基づく(真偽概念を使わない）意味論の例としては、認識的解釈や計算論的解釈があります。今回は認識的解釈の位置バージョンを取り上げます：∧の例
∧-I規則が∧の意味とする。
∧-Iの２つの上式を「Aであると分かる」「Bであることも分かる」と読むと、
下式「A∧B」は、「Aであると分かり、Bであることも分かる」ことを意味していると取れます。
∧-E規則はこの∧-Iによる意味で正当化される：上式A∧Bの意味から、AであることもBであることも分かっていることを意味しており、特に「Aであることが分かる」も言えることとなります。。

命題論理の推論規則

1. $\wedge\mathbf{-I}$：２つの論理式 A と B から論理式 $A \wedge B$ を推論してよい．これを次のように表記することとする．

$$\frac{A \quad B}{A \wedge B}\ \wedge\text{-}I$$

2 , 3. $\wedge\mathbf{-E}$：論理式 $A \wedge B$ から A を推論してよい．これを次のように表記することとする．

$$\frac{A \wedge B}{A}\ \wedge\text{-}E \qquad \text{又は} \qquad \frac{A \wedge B}{B}\ \wedge\text{-}E$$

→導入規則による→の意味と、その意味による→消去規則の正しさ：
A→Bの意味は、「Aであることが分かれば、そのことからBが分かる証明が与えられていること」と→-I規則を解釈します。。
このA→Bの意味から、→-E規則は次のように正しいく適用できる：
Aが分かっていれば、上記のA→Bも成立していれば、上記A→Bの意味からBであることが分かる。

A→Bの意味を¬A∨Bと同値と置いた真理表の意味論との違いに注意してください。
∨と¬についても同様に導入規則で意味をとり消去規則をその意味で正当化することができます。

$$\frac{\begin{array}{c}[A]^n \\ \vdots \\ B\end{array}}{A \to B} \to\!-I,n$$

$$\frac{A \quad A \to B}{B} \to\!-E$$

計算時の心の状態と外部との関係

```
  354
+ 438
-------
  792
```

左端の列を上から一段ずつ下に移動し、一桁の足し算を行っていく。途中で繰り上がったら「１つ繰り上がったぞ」という心の状態に変わる。列の下まで到達すたら一桁目の足し算結果を書き込み、一つ左の列を上から見ていく。「繰り上がりなし」の状態なら通常の一桁の足し算を進め、「１つ繰り上がったぞ」の状態であれば、３＋５を９とするように、通常の繰り上がりのない時の一桁の計算とは１つだけ異なる１桁の足し算の計算表を使う。

これを各列くりかえしてゆく。

- 左のような計算の手続は小学校低学年児童から大人まで従っている手続でではないでしょうか。いくつ繰り上がったかを直接ペンで書きこんでよいことにすれば、桁数が多く、足し算の項も多い場合は繰り上がりの数を直接書き込むのがよいやり方でしょうが、いまチューリングマシンとはなにかを考えるうえでは、繰り上がりの数がいくつであるかは「心の状態」と考えるのが自然です。

10項以内の足し算で、桁数はどんなに多き場合でも、繰り上がりについての心の状態１０種類とそれぞれの状態に対する１桁の足し算表盤１０盤があれば、計算していけます。心の状態変化と読み取り、書き込み。視線の移動の組み合わせを用いて規則を与えるのがチューリングのチューリングマシンのアイデアです。

上記足し算規則は少数の規則の集合で定式化できます。

チューリングは人の心の状態変化の代わりに機械の内部状態という概念を導入しました。

チューリングマシンとの比較
どちらも一次元の模様パターンを変換する規則を計算とみる。1936年にほぼ同時に誕生(完成）した、（Paper上の）万能計算機

- ラムダ計算

代入（β簡約）だけが計算規則。

データ構造やプログラムをこの唯一の計算規則に適合させる。

関数型プログラム言語の基本とも捉えられ、ソフトウエア的計算モデルと言える。

- Turingマシン
- 人の計算状態を内省することから完成させた。計算時の心の状態と、知覚的入力・計算動作・出力の関係性を、(人の心の状態の代わりに）マシンの内部状態と置いた。
- 記録テープと記号読み取り用スキャンヘッドの仮想敵ハードウエアを持つ計算モデル・

どちらも一次元上のパターン変換を計算とみています。例：簡単な足し算

- 基本教材で見たラムダ計算の足し算

((+3)2)は、

＋…111…11..のようなパターンをしていて、β簡約と言われる代入規則だけで、

…11111..のような形のチャーチ数となり既約となりました。

- チューリングマシンの対応する計算
- テープ上で

０００[1]１１０１１００００

囲みの部分にヘッドがあり、１を見ています。機械の内部状態は「さあ、計算を始めるぞ」という初期状態です。０は実際には何も記号が書き込まれていない、テープ上の空白のコマを表します。

０００[1]１１１１００００

２つの引数間の空欄のコマを一つつぶして、元の場所にヘッドが戻り、「計算が終わった」という内部状態で機械は止まり、あす。Λ計算の時同様、２本と３本の模様から開始し、５本続いた模様になり、２＋３の計算結果、５を表します。

チューリングマシンは、
必要なだけ延長できるコマで区切られたテープ（計算空間量無限）
有限個のコマには１という模様が書き込まれています。他のコマは空欄です。１以外の模様は書き込まれません。
但し
テープの読み取りや書き込みができるのはヘッドのある一コマだけ

ヘッドの移動はきとコマ右（Right)へか、一コマ左（Left)へかのいずれか

人の心の状態のような内部状態を機械に設定する
チューリング機械の各計算プログラムには**有限**の状態の集合が前提され、それらは通常の集合論表記で、{}と表される。ここで　は初期状態、　は計算が停止する終止状態。
プログラムは次の形をした規則の有限集合です。。

$q_」m \Rightarrow nQq_K$

ここで、ｍとｎは1か0を表しています。1はテープのコマの１という模様を表し、０はテープのコマが空欄であること（模様１が書きこまれていないこと）を示します。決して０が書き込まれていることお示すものではありません。QはRかLを表します。RはRight（右へ移動）、LはLeft(左に移動）を表します。上の規則は、「いま機械の内部状態がｑ」で、ヘッドが見ているコマがｍであったら、そのコマをｍにして、Qの方向に一マス分ヘッドを移動し、内部状態がq_K

に変わる。

内部状態と読み取りかけ消し移動

$q_j m \Rightarrow nQq_k$

「いま機械の内部状態が q j で、ヘッドが見ているコマが m であったら、そのコマを m にして、Q の方向に一マス分ヘッドを移動し、内部状態が qk に変わる。」

- $q_J 0 \Rightarrow 1Rq_K$ は、ヘッドが見ている空欄のコマに 1 をプリントして右日ヘッドを移すこと、

$q_J 1 \Rightarrow 1Rq_K$ は、ヘッドの位置で見ている 1 を消すことなくそのままにして右に移動すること

- $q_J 1 \Rightarrow 0Lq_K$ は，ヘッドがの位置で見ている 1 を消してそのマスを空欄にし、左へ移動すること、をそれぞれ示します。
- 計算は初期状態 q_1 で第一引数の最左端の 1 にヘッドがあるところから始まり、計算結果の最左端の 1 の位置で終止状態 q_{FIN} で終わります。

足し算（λ計算の時同様、二つの棒パターンを直接つなぐ）

$q_1 1 \Rightarrow 0 R q_2$

$q_2 1 \Rightarrow 1 R q_2$

$q_2 0 \Rightarrow 1 L q_3$

$q_3 1 \Rightarrow 1 L q_3$

$q_3 0 \Rightarrow 0 R q_{Fin}$

ここでこの足し算規則で用いられるチューリングマシン内部状態の集合は、

$\{ q_1, q_2, q_3, q_{Fin} \}$です。初期状態 q_1ではヘッドは第一引数の最初の１を見ているところから計算が始まります。終止状態q_{Fin} に有限ステップで到達し、その時１の列の一つだけの連結の最初の１にヘッドが来ていれば、計算が成功し、計算結果は連結した１の長さが著す数です。

３＋２の計算：二つの引数を表す１の連結列を区切る一コマの空白を１で埋めて、方向転換して逆戻りし、最初の空欄のコマまで来たら、右に戻って停止します。これだけでは、空欄を一コマ埋めたために引数の和よりも一つ１が多くなってしまいます。ですので、最初に右に動き出すときに、最初観ていた１を空欄に変えておきます。このことにより、ちょうど二つの引数の和と列となって収支状態になります。

0 0 0 0 1 1 1 0 1 1 0 0 0 0 0

$q_1 1 \Rightarrow 0Rq_2$

0 0 0 0 1 1 0 1 1 0 0 0 0 0

$q_2 1 \Rightarrow 1Rq_2$

0 0 0 0 1 1 0 1 1 0 0 0 0 0

$q_2 1 \Rightarrow 1Rq_2$

0 0 0 0 1 1 0 1 1 0 0 0 0 0

$q_2 0 \Rightarrow 1Lq_3$

0 0 0 0 1 1 1 1 1 0 0 0 0 0

$q_3 1 \Rightarrow 1Lq_3$

0 0 0 0 1 1 1 1 1 0 0 0 0 0

$q_3 1 \Rightarrow 1Lq_3$

0 0 0 0 1 1 1 1 1 0 0 0 0 0

$q_3 0 \Rightarrow 0Rq_{Fin}$

0 0 0 0 1 1 1 1 0 0 0 0 0

練習問題

- ラムダ計算の時のSuc関数に当たる関数のTuring Machine Programプログラムを与えよ。

Suc　は一引数関数で、入力の数の次の数を返す関数でした。

一引数関数なので、Turing Machineのテープは

00000<u>1</u>1111100000000000000

のような１の列が一箇所だけありヘッドは１の左端を見ているわけです。

練習問題の答え

- 規則により模様を＋１の結果になるように変えれば用だけですので、いろいろなプログラムをつくれます。たとえば前の足し算のプログラムの動き方を再利用しようとすれば、
- 最初の規則で

００００<u>１</u>１１０１１０００００

$q_1 1 \Rightarrow 0Rq_2$

００００<u>１</u>１０１１０００００

として最初の１を消した代わりに１を残すことにして、

$q_1 1 \Rightarrow 1Rq_2$

という規則と入れ替えれば、それだけでSucのプログラムです。

この規則は２つ目の規則$q_2 1 \Rightarrow 1Rq_2$

と同じ動作をする規則ですので、q_2 をq_1 と表せば、規則を一つ少なくできます。

別解答の例

- 前のページでは足し算で使ったプログラムの動きを用いて、引数の１１１１の列の右端に新たに１を追加した例でした。
- 左端に１を追加すると、たった一行の次の規則で書けます。

- $q_1 1 \Rightarrow 1Lq_1$
- $q_1 0 \Rightarrow 1Rq_2$
- $q_2 1 \Rightarrow 1Lq_{Fin}$
- 二つ目に規則でRではなくLをとることもできます。その場合は、計算全体で一マス分だけ多い計算スペースを使うことになります。

他の練習問題

- 上級者へ
- 型付きλ計算の場合は引数の数が型付けの仕方で定まっていたので、３＋５＋１２＋８＋１７
- または、+(3,5, 12,8,17)のように任意有限個の引数をとる足し算は、型付きの足し算を何度も重ね合わせて行うことになりました。
- 一方で、先にTuring Machineの足し算のプログラムは少し工夫すれば、任意の数の引数に対する足し算としてプログラムすることができます。
- やってみてください。

引き算

- 実はラムダ計算では減少関数を取り扱うには、型なしであってもすこし技術的工夫が必要です。
- 一方、Predecessor(一つ少ない数）や引き算なTuring Machineでは簡単にプログラムできます。
- テープに何も書き込まれていない形で終止状態となったときに０を表すことにしておきます。
- この時のPredecessorプログラムは、
- $q_1\,1 \Rightarrow 0Rq_{Fin}$

の一規則だけのプログラムとなります。

上級者向け練習問題

- 引数の数が任意有限個であるこちを許す足し算のプログラムを作成してください。
- 0001111011110111111101110000

例えば上は+(4,4,6,3)の入力表現に当たります。各引数間には厳格に一コマ分だけの空欄のコマがるとします。引数がいくつであっても、計算の開始時のヘッドの位置は、第一引数の左端です。

上級者向け練習問題

- 引き算m-nのプログラムを作成してください。
- 問題設定の単純化のため、ｍ＞ｎと仮定してよいことにします。

- ヒント；例えば、ｍの塊とｎの塊の間を行ったり来たりそながら、ひとつづつ双方から１を減らしていき、ｎの塊がなくなったら、ｍの残りを表示して終止状態になればよいわけです。

最上級者への練習問題

- 掛け算はラムダ計算では簡単でした。チューリング機械プログラムでは工夫が必要になります。最上級者の方t値への練習問題とします。

- 関数型プログラムの名前呼び出しに当たることをλ計算ではしばしば使いましたが、チューリング機械プログラムでエンコードするのは複雑になります。
- 自然数の入力に対して停止して、自然数を出力する関数に対しては、λ計算（型なし）とチューリング機械との計算力が同等であることが証明され、そしてこれをもって(逐次的)計算可能性概念の定義とされています。（チャーチ＝チューリングのテーゼ）