自然言語処理の原理

自然言語処理の原理

橋　本　直　樹

§1. 序論

自然言語をコンピュータで認識しその言語処理を行うことが広く行われている。最もよく知られているものの中に機械翻訳があり、特に英語とフランス語の相互の翻訳が深く研究されている。同様に英語と日本語の翻訳にもすぐれた研究がある。本稿では自然言語をコンピュータで処理する原理的な考え方を議論する。厳密な論理展開は行わなので、必ずしも正確に結果を論証することにはならないため一つの主張としての論説である。

言語は自然言語と人工言語に大別される。人工言語としてよく知られているものに人間どうしの会話に使われるエスペラント語があるが、ここでは人と計算機械との対話であるプログラミング言語を対象とする。プログラミング言語は主にコンピュータに処理をさせるための言語であるのでエスペラント語とは目的が異なるが言語学の知見に基づいて作成されているので興味深い。本論説では、言語翻訳ソフトウェアが行っているように、「なぜ自然言語がプログラミング言語で書かれたソフトで処理できるのか」という本質的な疑問にアプローチする。この疑問は現在までになされた研究を組み合わせることでほぼ理解できる。しかし厳密な意味ではまだ不十分な所も残る。

以下§2でチョムスキーの言語分類を示す。そして§3でチューリング機械を概説し、§4で主張を提示する。

§２．チョムスキーと言語処理

現代の科学的に言語を扱う方法の基盤は、チョムスキーによる生成文法[1,2,3]にある。チョムスキーは、言語の成立を生成文法という方法で4つに分類できるとした。当初の研究は自然言語を対象としたものであったが、文法を極めて厳密に記述しているため人工言語の分類に応用することが行われた。形式言語理論[4]という名で今日も発展している。なお、チョムスキー自身による自然言語の研究は、生成文法の不十分性から変形生成文法等の変遷をして今日に至っている。生成文法でチョムスキーが分類した文法の階層は、それぞれ言語生成に対応し、厳密に生成規則により定義される。その結果次の分類がなされている。

（ⅰ）帰納的可算集合（タイプ0言語）

（ⅱ）文脈依存言語（タイプ1言語）

（ⅲ）文脈自由言語（タイプ2言語）

（ⅳ）正規言語（タイプ3言語）

各言語の文法により生成される言語間の包含関係は、

タイプ0言語 ⊃ タイプ1言語 ⊃ タイプ2言語 ⊃ タイプ3言語の順である。より小さい言語集合ほど生成条件すなわち言語制約は強い。生成規則は記号で記述されるが本稿ではこの生成規則による証明は行わないので詳細は略する。

素朴に、自然言語はチョムスキーの分類によるどのクラスに属するかという問題が生じる。チョムスキー自身はこの問題のために生成文法を考案し、それをより完全にするために種々の拡張をしている。これらは実際の自然言語の事例を規則に当てはめて調べる以外に研究方法はなく、多くの人々により研究されている。1つの特定した言語でもその表現方法は無数にあり、結論を得ることが不可能である可能性もある。現在までの研究結果では、ほとんどの自然言語は文脈自由文法を用いているとされている。他方、一部の自然言語の中に文脈自由文法では記述できない例があるとの主張もある。しかし、そのような特別な自然言語は真に文脈自由文法でな

くても、例外として扱うことが可能である。従って、通常の言語処理の対象とする自然言語は文脈自由文法に基づく言語と限定してよい。

§３．チューリング機械

コンピュータの理論的研究はA. Turing[5]に始まり Von Neumannにより実際的な計算機の構成原理が与えられた。コンピュータ自身の研究とその上のアルゴリズムに関する研究が理論的研究として存在する。前者はいわゆるVon Neumann型コンピュータや新しいパラダイムに基づく量子コンピュータなどの計算論であり、後者は目的に応じたプログラム作成の方法論を数学的に実証、構成するものである。本稿は前者の立場に基づく内容の延長線上にある議論をする[6,7]。

チューリング機械は有限個の状態の１つを取ることができる有限制御部とセルに分割されたテープがあり、各セルには有限個の記号のいずれかが書かれている計算機のモデルである。

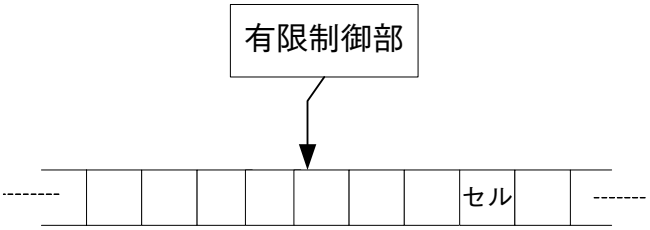

図１　チューリング機械

入力は有限の長さの記号列がテープ上に書かれている。テープの他のセルは左右に無限に続いていて、そこには空白と呼ばれる記号が書かれている。また、テープ上のセルのいずれかの位置の上にテープヘッドがあり、機械はそのセルの記号を読むことができる。この機械は次の動作を行う。

① 有限制御部の状態をかえる

② セルにテープ記号を上書きする

③　テープヘッドを右または左に移動する

これらのことを図を介さずにより厳密に形式的に書くと次のようなる。

定義[7] チューリング機械とは、7つ組　M＝(Q, Σ, Γ, δ, q_0, B, F）で、各成分は次を満たす。

Q：有限制御部の状態の有限集合

Σ：入力記号の有限集合

Γ：テープ記号の有限集合（Σ ⊆ Γ）

δ：遷移関数

δ（q, X）に対しqは状態の変数、Xはテープ記号とする。このときδ（q, X）の値は（p , Y, D）で次をみたす。

①　q∈Qは動作後の状態

②　Y⊂Γは、読んでいるセルに上書きされる記号

③　Dはヘッドが動く方向を表す（LまたはRをとる）

q_0：初期状態（q_0∈Q）

B：空白記号　（B∈Γ − Σ）

F：受理状態の集合（F⊂Q）

形式的な議論は、この7つ組によりその証明等がなされる。ただし、遷移関数は、遷移図という図式による証明も行われている。同様に以下の議論に登場するプッシュダウンオートマトンについても形式的な定義を述べる。

定義[6]　Pがプッシュダウンオートマトンとは7つ組

P = （Q, Σ, Γ, δ, q_0, Z_0, F）

である。ここで

Q：有限制御部の状態の有限集合

Σ：入力記号の有限集合

Γ：スタックに記録できる記号の有限集合

δ：遷移関数で次の条件を満たすδ（q, a, X）でその値は（p, γ）である。状態pは次の状態を表し、$\gamma \in X$は、スタックの一番上のXを置換する要素である。

① $q \in Q$

② $a \in \Sigma$ または　$a = \varepsilon$ （空列εは入力記号ではない）

③ $X \subset \Gamma$ でスタック記号という。

q_0：初期状態

Z_0：開始記号

F：受理状態の集合

チューリング機械とプッシュダウンオートマトンの定義を比較したときその大きな相違点は前者が無限に長いテープへの操作ができるのに対し、後者はそれが有限であること、またスタックという記憶装置を持つことである。これらの関係は十分に研究されているので略す。

§４．自然言語の情報処理

本節で、英語等の自然言語に対して、人工言語の一つであるプログラミング言語によりなぜ翻訳などの自然言語処理が可能なのかという問いについて議論する。

前述した通り自然言語は一部の例外を除き文脈自由文法で記述できると主張されている。しかし、Pullum-Gazda[8]らによりこれは未解決問題であるとの主張もある。この問題は、文脈自由文法の一般化とともに研究されているがはっきりとした結論は出ていない。このため、本稿では文脈自由文法で記述される自然言語を対象とすることを仮定する。

はじめにプッシュダウンオートマトンが認識する言語クラスについての結果を述べる。

定理１[9]　文脈自由言語のクラスと、あるプッシュダウンオートマトンによって最終状態で受理される言語のクラスは一致する。

この定理の直接証明は難しいので、あるプッシュダウンオートマトンにより空スタック（スタックを空にするような入力集合を持つ）で受理される中間的な言語クラスを用意し、上の2つのクラスとの同値性をそれぞれ示すことにより間接証明される。

プッシュダウンオートマトンには、前節の定義から明らかにチューリング機械に含まれる。文献7（p28）に示されているように、いかなる1個のスタックを持つプッシュダウンオートマトンでも認識されない言語をチューリング機械は認識できる例が構成されていることからも明らかである。

一方、C言語などのプログラミング言語では、一般のプログラムの構文は文脈自由文法で定義される。しかし、変数の宣言文などは文脈自由文法で記述することはできない。この部分は文脈依存言語に属する。この部分は非常に特殊なものなので一般化されたものとは扱わず、プログラミング言語の理論では属性（Attribute）という名のもとに文脈自由文法に条件を付加することにより処理される。この結果、プログラミング言語は文脈自由言語として処理できる。

チューリング機械と実際のコンピュータは、いずれも帰納可算言語を受理できる。しかし、コンピュータという概念を数学的に厳密に定義することはできていない。このためにチューリング機械とコンピュータとの対比は、厳密な証明が不可能なので直観的な議論になるが次のことが示されている。

定理2[7]

（ｉ）　コンピュータはチューリング機械を模倣できる。

（ii）　チューリング機械はコンピュータを模倣できる。

この証明においての問題は、チューリング機械のテープは無限の長さを持つが、コンピュータの記憶装置、主記憶、ディスクその他の記憶装置は有限であることである。無限の長さのテープを有限の量の記憶装置で模倣可能かどうかという問題が生じる。この種の議論で通常使われている説明は、使用可能なディスクの数の上限がないので、満杯になったらそれを取り替えることによりいくらでも多くのディスクが使えると仮定できるとすることである。ディスク上に2つのスタックを用意し、チューリング機械テープの左に位置するテープの中とテープヘッドの右に位置するデータ内容を保持する。この結果チューリング機械をコンピュータで模倣できると主張されている。また、チューリング機械がコンピュータを模倣することも文献7で示されている。しかし、数学的に厳密に定式化するには無理がある。

次にプログラミング言語がチューリング機械を模倣できるか否かという問題が生じる。近年、幾人かの研究者により提唱されてきた言葉にチューリング完全性という概念がある。これは、あるプログラミング言語がチューリング機械を完全に模倣できるという意味に用いられている。たとえ、ディスクメモリなどのハードウェア資源が有限のものであっても、プログラミング言語自身はそれを意識せずにプログラム化できる。このためチューリング機械の模倣ができる可能性が高い。実際はSQL言語などを除く多くのプログラミング言語（C言語等を含む）はチューリング完全性を持つことが示されている。証明は、そのプログラミング言語でチューリング機械の模倣を実際に構成する。いわゆる構成法による証明がなされている。これらの結果よりC言語などの多くのプログラミング言語はチューリング機械を模倣できる。上の定理 2 のチューリング機械はコンピュータの動作をすべて模倣できることと、チューリング完全なプログラミング言語がチューリング機械を模倣できること、及び定理1を用いると次の結果が得ら

れる。

主張　文脈自由文法で記述されているすべての自然言語はチューリング完全なプログラミング言語で認識できる。

前述したように自然言語が文脈自由文法か否かという問題は未解決の部分があるので、自然言語を文脈自由言語に限定することで主張を示した。しかし、文脈依存言語であっても上の主張は成立すると考えられる。これはチューリング機械が認識する言語が文脈依存言語も含むからである。プログラミング言語が文脈自由言語であることを考えると、チョムスキーの意味でより狭い言語でより広い言語のクラスのすべてが認識できるという一見不可思議なことが生じる。これらのことが「なぜそうなるのか」ということはもう少し深い考察が必要であると思われる。また、現存の多くのプログラミング言語により自然言語処理ができることになる。本稿ではその処理の時間的速度については考察していない。実際には処理が早いプログラミング言語が選択されており、原理的な理解とは異なる。

自然言語に対するプログラミング言語による認識の原理が、人が幼い時からより難しい概念を年を追って獲得していくのと同じ原理ではないのかと予想する。

参考文献

1．N. Chomsky, 1956, *Three Models for the description of language*, IRE Trans. Inf. Theory, IT-2,pp.113-124

2．N. Chomsky, 1959, *On certain formal properties of grammars*, Inf. & Control, 2,pp.137-167

3．S. Ginsburg,1966, *The Mathematical Theory of Context-Free Languages*, McGraw-Hill

4．嵩忠雄・都倉信樹・谷口健一　「形式言語理論」　コロナ社
5．A. M. Turing,1936, *On computable numbers with an application to the Entscheidungsproblem*, Proc. London Math. Society 2:42, pp.230-265
6．J.ホップクロフト・R.モトワニ・J.ウルマン著「オートマトン　言語理論　計算論Ⅰ」(サイエンス社)
7．J.ホップクロフト・R.モトワニ・J.ウルマン著「オートマトン　言語理論　計算論Ⅱ」(サイエンス社)
8．G. K. Pullum and G. Gazda, 1982, *Natural languages and context-free languages*, Liguistics and Philosophy, 4 , pp.471-504
9．J. Evey, 1963, *Application of pushdown store machines*, Proc. Fall Joint Computer Conference, AFIPS Press, Montvale, NJ, pp.215-227