ロンドン数学会において
アラン・マジソン・チューリングが
1947年2月20日に行った講演[1]の日本語訳と注解

墨　岡　　学

自動計算をするエンジンが現在N. P. L[2]において設計されていますが，それはたいへん大規模なデジタル計算機であります。たった一度の講義でこの特定のマシンについて技術的な細部を語ることは不可能でありますが，現在，計画されている同型の他のマシンについても，これから私がお話しすることの大部分はあてはまると思っています。

数学者としての観点から述べますと，デジタルであるということの性質は電気的であるということ以上に興味をそそるものです。電気的であるということは，これらのマシンが高速であるということにおいて重要なものであり，その高速性がなければ，その製造費用に対する資金援助がなされることはないでしょう。ですが，あらゆる実用面において，これは大切なことです。しかしながら，デジタルであるということは，さらに繊細で深い意味を持っています。それは，まず数が数字の並びで表現されることで，そのために望めば数を幾らでも長い数字の並びで表すことができます。必要なだけの精度で計算することが可能になります。この精度向上は，マシンの部品の調整や温度管理などによってではなく，マシン内部の数字のための装置を増やすことにより得られるのです。数の精度を2倍にしようとすれば，必ずその装置を2倍までに増設し，そ

1）“Charles Babbage Institute reprint series for the History of Computing”, MIT press, 1986 第10巻に収録された“Lecture to the London Mathematical Society on 20 February 1947 by A. M. Turing”からの翻訳。
2）英国テディントンにある National Physical Laboratory のこと。

の結果として各々の仕事をなしとげるための時間は長くなります。これは，アナログのマシンと対照的で，連続変量をとるマシン，たとえば微分解析機[3] のようなものだと，10進の数字の数を増やすごとにマシン全体の設計をやり直して，1桁増やすたびにおそらく10倍の費用がかかります。2つ目のデジタル計算マシンの利点は，その応用がある特定の分野に限定されないということです。微分解析機はこれまでに生み出されたアナログ計算マシンの中では最も汎用的なものですが，その使途においてデジタルとは比べ物にならないほど限定されたものです。ほとんどすべての常微分方程式を取り扱うことができますが，偏微分方程式の取り扱いは全くできませんし，線形同時方程式の大部分と多項式のゼロ問題を取り扱うこともできません。デジタルな計算マシンは，それと違い文字通りあらゆる計算の問題に取り組むことが可能です。実際に動いている素晴らしい例としてACE[4] がありますが，それは人間が計算できるジョブをその千分の1のさらに10分の1の時間でできます。この時間予測はかなり信頼できるものですが，ジョブがACEでやらせるにはあまりにも簡単すぎるものは別です。

もう何年か前になりましたが，私は今日の言葉で述べればデジタル計算機械の理論的な可能性とその限界について研究をしていました。私はある種のマシンについて考察していました，それは中央処理機構を持ち，無限の長さを持つテープに収めた無限のメモリーを持つものです。この型の機械が十分に汎用的であることが研究の結果わかりました。私の結論のひとつは，「目の子算（おおざっぱな人間的な）処理」と「マシン処理」は根本的には同じであるという

3）最初の微分解析機の発明は，1876年にJames Thomsonによる。実用的なものは，1927年にMITにおいてH. W. NiemanとVannevar Bushにより稼働した。微分解析機は，機械的なアナログ・コンピュータであり，積分によって微分方程式を解く。NiemanとBushによるレポートが1931年に公表され，英国においても微分解析機が製作された。Turingの講演は，これらのことを念頭においている。

4）ACEは，Automatic computing engineの略。エンジン（engine）の語が，計算機の意味で用いられるのは，英国人Babbageによる，最初の自動計算機の原型Analytical engineに由来する。

発見でした。ここでの「マシン処理」とは，もちろん私が考案した型のマシンで行われるものです。この理論的な考察において本質的であったのは，メモリーが無限になければならないということでした。そうでなければマシンは周期的な計算を繰り返すことしかできないということが容易に証明されました。ACE はマシンとしては，先に述べたのと同型のマシンを実用化したものとみなすことができます。少なくとも非常に近い類似性があります。デジタルの計算機械はすべて，1 つの中央処理機構と幾つか非常に拡張性に富む形式のメモリーを持ちます。メモリーは無限である必要はありませんが，非常に大きなものであることが確実に必要となります。一般的に無限長のテープにメモリーを配置するのは実用的機械では満足にできませんが，そのわけはある情報の断片が必要となったその瞬間にそれが格納された位置に移動してテープを上げたり，下げたりするためにはたいへんな時間がかかってしまうためです。ある問題が軽く 300 万項目の格納庫を必要とし，各々の項目が次に必要とされる項目に移動するまでに平均 100 万項目を動かさなければならないとするなら，これは実用的機械ではとても耐えることができません。その機械には，要求された項目へ一瞬にして到達することができるようなメモリーの形式が必要です。この困難は当然ながら本がパピルスの巻物に書かれていた時代のエジプト人を悩ませていました。彼らが本の中を探すときには手間が相当にかかったに違いありませんし，現代の本が書き物としてどこをとっても一瞬に開くことができるのはありがたいことです。テープへの格納やパピルスの巻物はどちらかというとアクセスしにくいものであると言わざるを得ません。それらはある項目を見つけるのに時間がかかってしまうからです。本のようなメモリーの形式はもっと便利で，さらに人間の目で読み取るには最適です。私たちは，本に準じた取り扱いができるメモリーを持つ計算機械を想像することができます[5]。それは決

5）Vannevar Bush がマイクロフィルムによる memex を考案したのは 1930 年代といわれている。このメモリーを拡張した（memex は，memory extender の合成語）という装置はハイパーテキストの前身ともいわれる。

して取り扱いは容易ではないかもしれませんが，一本の長いテープよりは好ましいものです。ここでは，本をメモリーとして利用するための困難は克服されているものとし，したがって，目的の本の目的のページ等々を見つけるための機械装置が開発済みだとしましょう。あたかも人間の手や目を模倣したようなものです。人は目にも留まらぬ速さでページを捲ろうとすればページを破いてしまいますし，人がさらに移動量を増やし，その速度を上げれば，そのために消費するエネルギーは膨大になってゆきます。したがって，もし私たちが一冊の本を1ミリ秒毎に10メートル動かし，その重さが200グラムだとすると，毎秒消費される運動エネルギーは10^{10}ワットにもなり，それは我が国の電力消費量の半分にもなります。私たちが真に高速な機械を持つためには，情報あるいはその一部を，本で得られるよりももっとアクセスしやすい形式にしなければなりません。それはコンパクト性と経済性の代償でしか得られないようにおもわれます，すなわち，本から各々のページを切り取り，それぞれを個別の読み取り機械に入れることです。しかし現在いくつも開発中の格納庫の手法は，このやり方ではありません。

もし格納庫機構のアクセスを極限まで高めようとするとコンパクト性と経済性において計り知れないほどの金額を払わなければならないことに気付きます。たとえば，最もアクセスが良いと知られている格納機構は真空管フリップフロップまたはジョルダン－エクル[6)]の回路です。これは1個の数字を格納でき，2つの値をとり，2個の熱イオン管を使用します。このような装置によって通常の小説を格納しようとすれば数百万ポンドの費用がかかってしまいます。私たちは明らかに紙やフィルム等よりもアクセスしやすく真空管をそのまま使うよりは場所とお金を節約できる融合的なものを必要としています。もう1つの望ましい特徴は計算機械の中でメモリーに記録できるようなものであること，これが格納庫にすでに何かがあるかどうかにかかわらずできるようにす

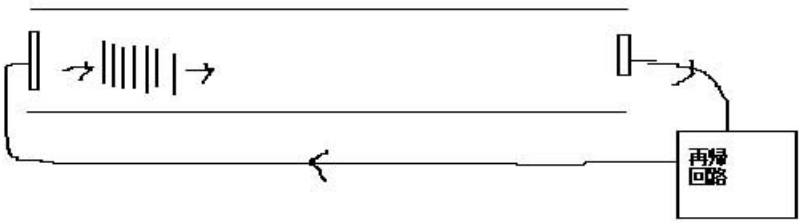
6）Eccles-Jordan回路ともいう。現代では，すたれてしまったが，1919年に発見されコンピュータの論理回路の基礎となった。

ること，すなわち，格納庫の中は消去可能でなければならないことです。

すでに 3 つの型の格納庫が開発されていますが，これらの特徴を多少持っています。磁気ワイヤーはとてもコンパクトで，消去可能で，機械の中での記録ができ，かつアクセス性は中程度です。陰極線管（CRT）の画面に電荷パターンを記録する方式の格納庫があります。これはおそらく究極の答えでしょう。ジョルダン－エクルの回路に近いアクセス性があります。3 番目は，音響遅延回路によるものです。これのアクセス性は磁気ワイヤーよりも良く，CRT 型よりも劣ります。アクセス性はほとんどの目的にとり問題がありません。その利点は，すでに改良が進んでいることです。いまのところ ACE の主メモリーは音響遅延回路により提供されることになっており，水銀タンクが使われます。

メモリー装置に音響遅延回路を使うアイデアは，私がフィラデルフィア大学のエッカートを信頼しているためです，彼は Eniac のチーフ・エンジニアでありました。そのアイデアとは，情報を水銀を柱状に圧縮波として移動させるものです。液体や固体は，音波を驚くほど高い周波数で伝道することが可能です，これは 5 インチ真空管に毎秒 1000 パルスを送り込むことに匹敵します。信号はピエゾ圧電気により水銀に伝えられ，それは反対側で水晶結晶板により検出されます。パルスの列，情報の表現は，水銀の中を移動しながら，そこに格納されているといえるでしょう。情報の列の取り出しが必要となるまで何度も繰り返しながら柱状にして元に戻されます。

このためには「再帰回路」が必要となります，タンクから出てきた信号を読



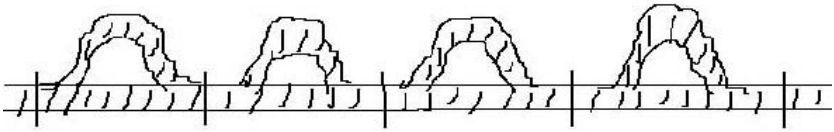
み取り，それを増幅してもう一度タンクに返すためのものです。これが単純な増幅アンプによって行われると，タンクとアンプの性質からして，せいぜい10回繰り返すのがやっとであるということは明らかです。実際にはこの再帰回路が行っているのは，やや特殊なことです。それがやっていることを正しく表現するには点集合のトポロジーを使うのがいいでしょう。図中の平面はあらゆる信号の空間をあらわしています。もちろん私はこれが2次元であることを示そうとはしていません。この信号空間を変数とする関数fが定義され，その関数の値もその信号空間に含まれると仮定します。実際に$f(s)$は，信号sがタンクに伝達され再帰回路を通って影響を受けたものをあらわすと仮定します。しかしながら，熱攪拌の影響により再帰の結果は，$f(s)$の周り半径δの範囲内の任意の点にあると仮定します。したがって，タンクがN個の異なった信号を区別して格納するための必要かつ十分な条件は，N個の集合$E_1 \ldots E_N$が存在し，F_γがE_γとの距離ε内にある点集合で

$$s \in F_\gamma \supset f(s) \in E_\gamma$$

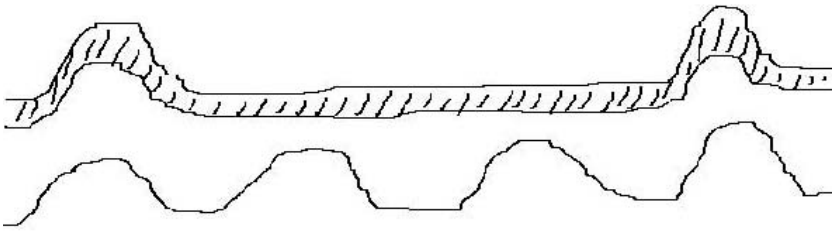
でかつ，集合F_γは互いに交わらないことです。

ここで明らかな十分条件は，初めに与えられた信号が再帰回路を何度通っても，集合F_γに属しているということで，これを満たせば信号を混同する危険性はありません。必要条件は，$s_1 \ldots s_N$がそれぞれ異なる意味を持つ信号で，かつ機械に入れられ，その後読み出されても混同の恐れがないことです。

ここで E_γ を関数 f を連続して適用して得られた s_γ からなる信号の集合とし，距離は ϵ 以下しか離れていないとします。したがって集合 E_γ は交わりません。水銀遅延回路で $N = 16$ の場合，集合は斜線部の範囲内のすべての連続した信号からなります[7]。



集合の中の 1 つはすべての連続した信号が下図の中に含まれるものからなっているでしょう。これは信号 1001 を表しています。



このような再帰的システムを効果的にするにはメモリーシステムに時刻信号を与えることが本質的となります，これによってパルスが生じた時間を識別することができるようになります。たとえば上の図に示すように時間を合わせる正弦波を与えるのが自然です。

プロセス f のアイデアを述べてきましたが，これは格納装置としてはごく一般的なものです。格納の「再生」として知られています。それは常時何らかの形で現れますが，時に再生は自然に起き，事前の注意が払われていません。別のケースではこのような f プロセスを改良するために特別の注意を取らないと

7）この図では 4 つの区画を持ち，$2^4 = 16$。

いけません，さもないと記憶は薄れてしまいます。

遅延回路において再生プロセスのために時計が重要であることを興味深いちょっとした定理で示すことができます。条件 $s \in F_\gamma \supset f(s) \in E_\gamma$ の代わりに，もっと強いもの，すなわち $f^n(s) \rightarrow c_\gamma \ \ if\, s \in E_\gamma$ を仮定します，これはどういうことかといえば，理想的な識別可能な信号の形式があり，各々の許容される信号は再帰回路を繰り返すうちに理想的な形に収束するということです。つぎに，時計がなければ，理想的な信号はすべて定数であることを示すことができます。たとえば，U_α が原点をずらすことを表すとします。すなわち，$U_\alpha s(t) = s(t+\alpha)$ です。つぎに，時計がないことから再生回路の性質は時間が経過しても常に同じであり，したがって f は U_α と可換となります。故に，$fU_\alpha(c_\gamma) = U_\alpha f(c_\gamma) = U_\alpha c_\gamma \ \ for\ f(c_\gamma) = c_\gamma$ となります，なぜなら c_γ は理想の信号だからです。しかし，このことは $U_\alpha(c_\gamma)$ が理想の信号のひとつであることを意味しています，したがって十分に小さな α については，これは c_γ とならなければなりません，なぜなら理想の信号が離散的だからです。したがって，任意の β と十分に大きな u については β/u は十分に小さくかつ $U_{\beta/u}(c) = c$ となります。しかし，反復により $U^u_{\beta/u}(c) = U_\beta(c)$ すなわち，$c(t+\beta) = c(t)$ となります。これは，理想的な信号がある定数となることを意味します。

私たちは，次のように言ってもいいかも知れません，時計は時間に離散性を導入することを可能にした，それはある目的のためには，時間は連続した流れではなく，瞬間が続いているとみなすことができるようになったと言えます。デジタルマシンは本質的に離散的なものを取り扱わねばなりません，ACE の場合には時計を使ってこのことが可能になりました。その他のすべてのデジタルマシンについても，私が知っている人間やその他の脳を除いて，同じです。時計を使わないやり方を考え出せるかも知れませんが，技術的には劣るものです。私は ACE における時計の利用は，再帰回路のみでなく，ほとんどの部品で使われていることを述べておきます。

同じように水銀遅延回路を私たちが利用するにつれて，それに接続されるこ

とになったいくつかの物体について述べておきましょう。私たちは5フィートの長さの真空管，内径は半インチですが，それをいくつか使用します。各々1024桁の2進数を格納することができます。私がここで格納（ストレージ）容量に使う単位は自明でしょう。格納機構がm桁の2進数の容量を持つということは，0か1からなるm個の数字の列を記憶できるということです。ストレージ容量は，それが記憶できる異なる信号の2を底とする対数で表されます，すなわち，$\log_2 N$ です。数字は1マイクロ秒の間隔で配置され，したがって波が真空管を伝達するのに要する時間は1ミリ秒あまりとなります。この速度は秒速1.5キロメートルとなります。アクセスの遅延時間あるいは情報を取り出すための待ち時間は約0.5ミリ秒です。実用的には，この時間は150マイクロ秒まで短縮することが可能です。水銀遅延回路によるACEで利用可能なストレージ容量は，2進数で200,000桁となります。これは，おそらくミノウの記憶容量に相当します[8]。

私は，この講演でメモリーの問題についてかなりの時間をさいてきました，なぜならば，適切なストレージの供給がデジタル・コンピュータの問題を解決するキーであると信じているからで，それらが真の知性を持つことを示すためには，すでに利用可能なものよりも，もっとはるかに大きな容量が提供されなければならないことは，間違いありません。私の主張は，より大きなメモリーを短期間に用意することが，演算，たとえば乗算の高速化よりも，もっと重要であるということです。演算速度は，マシンが商用として使われるなら価値があります，しかし，トリビアルな演算をさせる以上のことをさせるためには，より大容量のストレージが必要となります。ストレージの容量は，したがって，より本質的に必要なものです。

8）ミノウ（minnow）は，ヨーロッパ産コイ科の小魚。この講演でチューリングが解説するMercury Acoustic Delay Lineは，ランダム・アクセス・メモリーの最も初期のものであった。もっとも，ミノウが日本語化しているのは，釣りの世界。小魚の形をしたルアーをミノーと呼ぶ。

さて，ここで無限長のテープを持つ理論的計算マシンとの類推に戻りましょう。そのようなマシンは，ただ 1 つで，あらゆる仕事をすることが可能なことが証明できます。実際，他のマシンのモデルとして動作するように作ることもできます。この特別なマシンはユニバーサル・マシン（万能機械）と呼ばれますが，それは次のようなきわめて単純な方法で動作します。シミュレーションをしようとするマシンが決定すれば，その記述を万能機械のテープにパンチします。この記述では，そのマシンがすべての状態において，それが次に行う動作を明らかにしておきます。万能機械は常に，次に何をするかを，その記述を読みに行き見つけるだけです。したがいまして，シミュレーションをしなければならない機械の複雑さは，テープにすべて入っており，万能機械に固有のものはどこにもありません。

この万能機械の性質からすれば，機械による処理も，人間的な処理も元は同じものであるという事実を受け止めることができれば，万能機械はそれに適当な命令を与えれば，どんな人間的な仕事もこなすようにすることができます。この特長は ACE のようなデジタル計算機において生かされています。それらは事実，万能計算機の実用的なものであるということです。そこには確かに電子装置のプールがあり，大きなメモリーがあります。任意のある問題が取り扱われることになれば，その計算プロセスに含まれる命令が ACE のメモリーに格納され，そのプロセスを実行するために「セットアップ」されます。

私は，デジタル計算機の背後にある戦略的なアイデアのおもなものを主張してきましたが，これを裏付けるため ACE について手短に説明しましょう。議論を明確にするために次の部分について分けることにします。

メモリー
制御部
算術演算部
入力と出力

すでにメモリーについては，私はもう十分にお話ししましたから，簡単に ACE のメモリーが主にそれぞれが 1024 ビットからなる 200 個の水銀遅延回路から構成されていることを繰り返しておきます。制御部の目的は，メモリーから正しい命令を取り出して，その意味を調べ，それを実行できるようにします。そこには，ある「命令コード」が書き留められていると理解されますが，各々の「ワード」や 32 ビットの組み合わせがある特定の演算を記述します。制御部の回路はコードとうまく合うように作られた結果，正しい動作を生みます。また，私たちはほとんどのケースで，コードを決めてから回路を設計するようにしていました，これはすなわち，「最適コード」を想定してから，それを動作させるための回路を見つけ出すということだけではなく，コードを犠牲にしても回路を単純化することがしばしばあったということです。さらに，何の具体的な回路も頭の中に置かず「全くの抽象」でコードのことを考えることは非常に困難です。算術演算部は，加算，乗算など特別な回路で実行するほうが，制御部による単純な装置を通して実行するよりも効果的な処理となります。算術演算部と制御部の区別はあいまいですが，マシンは少なくとも加算器と乗算器を持っている必要があります，結果的にそれらが制御部の一部であるとみなされることがあるかもしれません。これは重要な点ですが，マシンは 2 進法を使って操作されるのですが，2 つの必要条件があります。入力へ外部から提供されるデータは 10 進数です，さらに人間の眼のために出力されるものも同じです，ただし ACE で再び処理されるために出力されるものはそうではありません。これが第 1 番目の必要条件です。2 番目は 2 進法で処理するように設計されたものですが，10 進処理をさせないというものではありません，理由は，ACE が万能機械だからです。2 進処理は大規模な計算機にとっては自然なやり方です。2 進法で動かすことは他の進法よりもはるかに容易です，なぜなら，２つの安定した状態をとる機構を作るのは割に簡単で，２つの状態をそれぞれ０と１にみなせばよいのです。図に示すようにレバーやジョルダン・エックル回路，サイラトロン[9] になどがその例です。

もしも小さな計算機の設計にかかわれば，2進法を使うことに，少なくとも1つの真剣な反対意見が出ます。実用化のため，数を2進から10進に変換，あるいは，その逆に変換するためのコンバータを設計する必要があります。これは，2進計算よりもずっと負担がかかります。大規模な計算機では，このために配慮する割合はずっと少なくなります。それは，まず，変換機は単に小さな道具にすぎないことと，つぎに，それは本質的には必ずしも要るものではないからです。この最後の言葉はまったく矛盾したように聞こえます。しかし，そのことは，これらの機械がそこに適当な命令を記憶することによって，どんなあいまいな処理でも行うことができると言う事実を知れば，そこから得られる単純な結論になります。特に，2進10進変換を行うことも可能です。例として，ACEの場合に，変換機を提供するには，メモリーに2つの遅延回路を追加するだけでいいのです。これは，ACEの場合についての特別なケースです。注意を払わなければならない細かなことがらで，めんどうなものがたくさんあるときは，工学的な常識にしたがって特別な回路が必要となります。そのようなとき，私たちは，マシンそのものを修正することなく，純粋に机上の作業により，結果として適当な命令を追加することで解決することができます。

マシンの各部についてもどることにしましょう。私は，それは2進法で動くことができると言いました。電気的なパルスが1を表し，パルスがないのは0

9）ガス入りの電子管。グリッドにより電流をオン，オフする。

を表すと約束することは不自然ではありません。こうすれば，数字の列 0010110 は次のような信号で表現されます。

ここで，時間間隔は 1 マイクロ秒です。さて，2 進法での加算がどのようなものかその過程をみましょう。通常の 10 進の加算では，私たちは常に右からはじめますが，おなじようにするのが 2 進でも自然でしょう。私たちがこうしなければいけないのは，常に下位の桁からはじめなければ，桁上がり（キャリー）をどうするか決めれないからです。同様のことが電気的な加算でもあてはまります。したがって，下位の桁から電気的なパルスが来ることになります。この結果として不幸なことですが，私たちは 2 進数を最下位の桁を左から書き始めなければならなくなります，あるいは，図の時間軸の流れを右から左にするかです。後者を選べば，私たちは右から左に加算をするときに書くことになりますが，ここでは最下位の桁を左端に書くことにしました。それでは，加算の例をやってみましょう。桁上がりを，加えられた数の上に書くことにします。

```
Carry   0 1 1 1 1 1 1 0 0 1 1
      A 0 1 1 0 1 1 0 0 1 0 1 ...
      B 0 1 1 1 0 1 0 0 1 1 ...
      ---------------------------
        0 1 0 0 1 1 0 0 0
```

ここで加算は，データのほんの一部を見るだけでできることに注意してください。加算を電気的に行うためには，私たちに必要なのは 3 つの入力と 2 つの出力です。

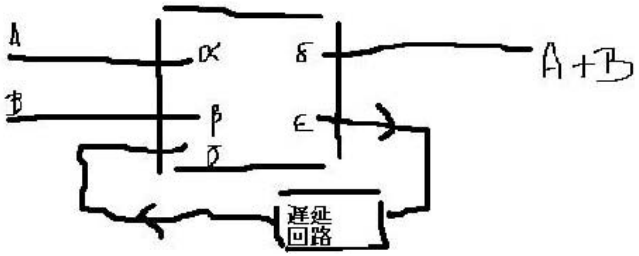
入力　　出力

被加算数　A　α　　和　δ

被加算数　B　β　　桁上がり　ε

前の桁からの桁上がり　γ

この回路は，次のようになります。



		和	桁上がり
	0	0	0
	1	1	0
入力 α，β，γ 中の 1 が	2	0	1
	3	1	1

入力のパルスの数に応じて適当な電圧を発生させる回路を作るのはとても簡単で，それは 4 つの状態を識別し，それに対する適当な和と桁上がりを出力すればよいだけです。私はそのような回路を説明しようとはおもいません，とても簡単に違いないからです。そのような回路があるとすれば，それをフィードバックに接続すればよいだけです，これで加算器ができあがります。

ここで私たちは，各桁の数字の加算について同じ過程が用いられること，さらに電気回路の性質が時間のずれについて，それがクロック数の倍数であるならば不変であることなどを利用しているのは明らかです。ここで言っておかな

ければならないことがあります，それは，これらの時間をずらせた群と，実数の倍数の群とが異種同形であることを利用していることですが，この原理が他に応用できるかどうかは疑問に思っています。

このような加算器でおわかりのように，加算は最も初歩的な 1 桁ずつを加えるというステップに分解することができます。この各々は 1 マイクロ秒です。私たちがあつかう数は 32 ビットです。したがって，2 つの数の加算は 32 マイクロ秒です。同じようにして，私たちは乗算を 1 と 1 あるいは 1 と 0 等々の加算を続けていくことにより行うことができます。2 つの 32 ビットの数の乗算は，1024 回程度の加算によってなされますから，乗算にかかる時間は約 1 ミリ秒と予測されます。実際に ACE に使用されている乗算器は 2 ミリ秒よりも時間がかかります。これは，演算単位が 1 マイクロ秒に過ぎないことと比べるとやや長い時間のように聞こえるかも知れませんが，実際にはマシンはこのような見方に対してはかなりバランスの取れたものです。というのも，乗算の時間は重大なボトルネックではないからです。コンピュータが，数を書き出したり次に何を実行するかを判定するための時間は，常に，実際の乗算時間と同じくらいかかりますし，それは ACE においても同様です。長い時間が，数をストレージに入れたり出したりすることや，次に何を実行するか判定するために費やされます。四則演算のうち，減算は補数と加算により行われ，除算は次のような反復法により行われます。

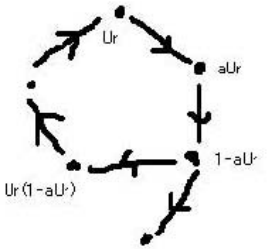
$$u_n = u_{n-1} + u_{n-1}(1 - \alpha u_{n-1})$$

u_n は，$|1 - \alpha u_0| < 1$ であれば α^{-1} に収束します。誤差は各ステップでべき乗される結果として急速に収束します。この過程はもちろんプログラムされています。すなわち，余分に必要な道具は関連する命令群を格納するための遅延線のみです。

算術演算部の次に残っているのは，入力部と出力部です。入出力のためには，私たちはホレリス・カード[10] を選択しました。私たちは，特別な装置を開発

することなく使うことに成功しました。そこで得られた速度は，電子装置の動作速度とは比較になりませんが，実際に興味ある長い計算のあとに出てくる簡潔な結果をみるには十分です。ホレリス装置に見合った低速度で提供される情報を，ACE で要求される高速度に合わせて変換するのは大変な困難がありそうですが，実際にはとても簡単です。ホレリス装置の速度は非常に低速なので，いろいろな処理からするとその速度はゼロか，あるいは停止しているとみなすことができ，このために数の変換は，静止した数字の列をパルスの流れに変えるような簡単なものになります。この仕事は整流器によって行われます。

マシンの説明についての概要を終える前に，プログラミングにおいて戦略が必要となる状況に遭遇した場合について述べたいと思います。そのうちの 2 つについて，さきほど述べた反復計算を例に図解できます。1 つは，反復サイクルについてのアイデアです。この式で u_r から u_{r-1} に移行する毎に，私たちは同じ計算手順を繰り返します。そこで，ストレージに入れた同じ命令を使用できれば，ストレージの節約になります。この結果，次のような命令サイクルをぐるぐる回ります。



しかし，この命令サイクルには，そのループにはまってしまい抜け出すことができなくなる危険性がありそうです。この問題を解決するには，また別の戦略を持ち込む必要があります。それは，条件判定[11] をプログラマが入力デー

10）ハーマン・ホレリス（Herman Hollerith）がジャカード・パンチカードを 1890 年のアメリカの大規模な人口調査に用いたのがはじまりで，コンピュータの入出力用パンチカードとして 1980 年代まで使用された。

タを利用するのではなく，マシンそのものから得られる結果を利用して行うことです。先ほどの例では，条件判定は$|1-\alpha u|$の値が十分に小さいこととなります。これは，まるで航空機が飛行場の周りを旋回するたびに，着陸許可を得ようとするようなものです。この反復のアイデアは，あまりにも単純な思いつきのようですが，最高度に重要な概念です。メモリーの中の命令のほとんどは，ものすごく何度も実行させなければならないことが分かれば，命令を反復させるというアイデアは，むしろなくてはならない基本的なものとなります。もしも，メモリー全体が命令群で占領されてしまうと，数やその他のデータのために使うことができなくなってしまいますし，それぞれの命令が 1 回だけ実行されて，もし最長時間かかったとしても，マシンの動作時間は 16 秒ほどにしかなりません。

もう 1 つの重要なアイデアは，構築した命令にマシンを従わせることです[12]。これは条件判定にかかわるまた別の戦略です。さきほどの例で，もし$|1-\alpha u|$が$2^{-3.1}$よりも小さいならば 1 で，それ以外は 0 であるような値が計算できるとしましょう。この値を命令の分岐点に付け加えることにより，命令は$1-\alpha u$が十分に小さなものになるまで実行され，結果としてその命令そのものを完全に別のものに変えたような効果があります。

おそらく命令表に関する最も重要な点は，標準となる「補助テーブル[13]」でしょう。複数のある過程は，あらゆる種類の組み合わせで何度も繰り返して使用されます。そのため，私たちはできればいつもメモリーの同じ場所から，同じ命令を利用したいと願います。したがって，私たちが，できるだけ多くの異なる関数の計算に補間を利用しようとすれば，補間のために同じ命令表が常に利用できなければなりません。私たちは，これをどうやればいいのかを一度だ

11）原文では，discrimination。現代のプログラム言語の while... do, repeat... until, for... next など条件反復のもとになったアイデアである。
12）ここでのもうひとつの discrimination（条件判定）は，if... then... else にかかわるもので，コードの流れを分岐させるためのもの。
13）原文では，Subsidiary table，これはサブルーチンのもとになった概念。

け考え，後は，それがどうやってできたのかは忘れてしまいたいのです。補間をしなければならなくなる毎に，そのテーブルが保存されている場所がメモリーのどこにあるかを思い出せばよく，その補間をしている命令表の中から適当なものを参照するのです。それでは，ここで例として，私たちが $J_0(x)$ の値を求めるための命令表を作成しようとしているとしましょう。そこでは，このやり方で補間テーブルを用いることにします。補間テーブルは $J_0(x)$ を計算するための補助的なテーブルであると言ってもかまいません。このようにテーブルにはある種の階層が存在します。補間テーブルは，J_0 テーブルから指令を受け，その答えを報告するものとみなされます。主人と召使の関係に似ていますが少し違います。なぜなら，召使よりも主人の方がたくさんいて，たくさんの主人が同じ召使を共有しなければならないからです。

それでは，ここで，このマシンがどのように動くのかを具体的に説明させてください。ある顧客によって持ち込まれた問題から始めましょう。はじめに，その問題についての事前準備からとりかかります。その問題が適切な形式になっているかどうか，さらに矛盾がないかどうか調べます。それが済むと，とても粗っぽい計算のための手続きを作ります。次に，テーブルを用意する段階になります。ここでは，例として，次の方程式の解を

$$y'' + xy' = J_0(x)$$

初期値 $x = y = 0$, $y' = a$ のもとで求めることにします。これは，次のような方程式を解く，特別な場合とみなすことができ，

$$y'' = F(x, y, y')$$

そこでは，命令表がすでに用意されているものとします。私たちが必要なのは関数 $F(x, y, z)$（ここで $F(x, y, z) = J_0(x) - xz$ については，おもに $J_0(x)$ を作るためのテーブルがもうすでにあるものとしましょう）を生成するためのもうひとつのテーブルとなります。これに付け加えなければならない細かなことがい

くつかあります。それは，境界条件や弧の長さなどですが，それらも $J_0(x)$ のテーブルがもうすでにあるものから得られたように，すでにある命令表の在庫品の中から見つけ出すことができるでしょう。そのジョブのための命令群は，わずかばかりの特殊なものを除いて，ほとんど在庫の中から取り出すことができるでしょう。標準的なプロセスの命令群をパンチしたカードは在庫品として用意されていますが，新しいものはそれらとは別にパンチされることになります。これらのカードすべてが集められ，チェックされ入力のための器械，それは単にホレリス読取器です，に運ばれます。それらのカードは重ねられてホッパーに入れられ，ボタンが押されるとカードは順番に読み込まれます。ここで忘れてはならないのは，マシンの初期状態では，何の命令もそこには入っていないことです。したがって，普通なら当然ありそうな機能もなく，初期状態では，まったく何もないのです。したがって，最初に読み込まれるカードについては,このことについて注意深く考えたものでなければなりません。それらは，イニシャル入力カードといわれ，常に同じものです。それが読み込まれると，ごくわずかな基本命令がマシンにセットされるようになっています。それによって，マシンは，特別な目的のジョブのための命令を読み込むことができるようになります。これが実行された後，マシンが次になにをするのかについては，あらゆる可能性があり，それは，ジョブがどのようにプログラムされているのかによります。マシンは，もちろん，そのままストレートに動作し，ジョブを実行し，要求された答えをすべてパンチあるいは印刷し，これらが全部完了した後に停止することもありえます。しかしながら，たいていの場合は，マシンは命令表が読み込まれたなら即座に停止するようになっています。これにはどのような利点があるかといえば，まず，メモリーに読み込まれた内容が正しいかどうかをチェックし，命令実行手順のさまざまな可能性についての検討ができるということです。[14] これは，仕事を中断するのにちょうどいい頃合で

14）デバッグに相当する。

す。私たちは，まだ他にも中断をしてもかまわないのです。たとえば，あるパラメータ α の値に関心があるとします，その値は実験的計算により得られるものであるとすれば，パラメータの値が得られたあと一旦停止し，それを別のカードで入力するというようなことを何度も繰り返すことになります。あるいは，カードを全部ホッパーに用意しておき，ACE が必要になるたびに，そこからカードを読み込むというようなやり方をしてもいいのです。誰もが，各自が望むようなやり方ができますが，各自で自分のやり方を決めねばなりません。マシンが停止するごとに「ワード」すなわち 32 ビットの数字の列がネオン管に表示されます。このワードは，マシンが停止した理由を示します。私はすでに 2 つの理由をあげました。その他の起こりそうな理由の大部分は，チェックにより提供されるものです。プログラミングは，ACE が頻繁に自分の正当性についてすべてを調べるようになされていなければなりません。これらのうち 1 つでも失敗したならばマシンは停止して，その失敗した検査の結果を示すワードを表示する必要があります。

ここで，しでかしそうな間違いが山のようにあることが，お分かりになるでしょう。困難なことのひとつは，ある間違いの原因を探す規則を作ったとしても，全部は追跡しきれないということです。きちんとした仕事をするためには，図書館の有能な司書のような役割持つ人々が私たちに必要となります。

最後に私は，電子的なデジタル計算機が数学に与える反響について 2，3 の推測をしておきたいと思います。私は，すでに ACE はおよそ 10,000 台のコンピュータ[15] に相当する仕事をするだろうということを述べました。そのことにより，人手による計算のほとんどは死滅すると予想されます。コンピュータは，ちょっとした計算，たとえば，公式に値を代入するといったようなときには役に立ちますが，その計算が人間によって数日もかかるような場合にはいつでも電子計算機が人間の代わりをすることになるでしょう。これは，電子計

15）原文で computer とあるのは，当時の機械式の計算機のことをさす。

算機を興味を持って所有する人々には必然的なことですが，そうではない人々にも影響は及びます。遠隔地にあるコンピュータをコントロールできるようにお膳立てすることはそのうちに必ず可能になります，たとえば電話線を使うような手段があります[16]。このための端末として入力と出力ができる装置が数百ポンドで買えるようになるはずです。しかしながらこれらの電子計算機による仕事の大半は，手ではとても計算できそうもない，大規模な計算になります。この計算問題をマシンに提供するためには，私たちはたくさんの能力ある数学者を必要とするでしょう。この数学者たちはその問題について事前の準備として研究を行う必要があり，計算のための形式化をすることになります。アナリストが広範囲な分野で必要です。人間コンピュータが問題を解いているときは，彼はいつも常識を働かせることによって，自分の解答がどのくらい正しいのかを知ることができます。ところがデジタル・コンピュータではもはや常識に頼ることはできず，誤差の範囲をすでに証明されている不等式をもとにするしかありません。私たちは，私たちのためにアナリストが適当な不等式を見つけてくれるのを必要としています。その不等式は，必ずしも明瞭でなくてもかまいません。すなわち，計算が始まる前には，はっきりとわからない程度のもので，鉛筆と紙を使って誤差がこの程度の大きさだろうと言える程度でかまわないのです。誤差計算は，ACE が責任を取るべきものの中で重要な部分を占めています。ある程度までは，誤差を統計的な推測値で置き換えることは可能でしょう。たとえば，電子ルーレットのような装置で乱数によって制御しながら毎回違うやり方で誤差を丸めるというような方法でもってジョブを何度も繰り返して統計的に推測することです。そんな統計的推測のやり方には，しかしながら，多くの疑問があり，マシンタイムの無駄でもあり，結局，誤差がどうしようもなく大きくなったとき，どうしたらよいのか何ら示すことができません。統計的な手法は，アナリストを助けはしますが，その代わりをすることは

16）今から 60 年前に広域のコンピュータネットワークと電話線のような具体的な媒体を指摘している。

できないのです。

アナリストは私たちが ACE のために適切な数学者を選ぶための手段のひとつです。ざっとしたお話をすれば，ACE に関わる人たちは ACE の主人と ACE の召使に分けられます。ACE の主人はそのための命令表を計画し作成します，そのためにはそれの使い方について深く深く思考しなければなりません。召使はそれをカードにして ACE に要求されるごとに提供することになります。彼らはうまくいかないようなものでも，即ちにそれを ACE に入力します。彼らは ACE が要求するならデータをかき集めます。召使は，言われたままに動く手足のようなものです。時代が進めば，やがて計算機[17] そのものが主人や召使の役割を取って代わることになるでしょう。召使は，機械や電気の知覚器官をもつ手足に置き換わるでしょう。たとえば，いろいろな曲線を直接読み取ってデータにすることができるようになり，女の子が数値を読み取ってカードにパンチするようなことに取って代わります。主人についても，しだいにあらゆる定型のものについて命令表[18] システムを電子計算機自体が考案できるように技術が進めば[19] やがてそれに取って代わられるでしょう。しかし，もしかすると主人たちはこれを拒むようなことが起きるかもしれません。彼らは，自分の仕事をこのようなやり方で奪われるのをいやがるかもしれません。そのような場合，彼らは自分の仕事を神秘のベールで包み，言い訳をして，ちんぷんかんぷんな話に閉じこもり，自分らを危うくさせる提案からうまくのがれようとします。私は，この種の反応は真に大変危険なものだと考えています。このトピックは，計算機械がどこまで人間の行動をシミュレートすることが原理的に可能なのかという問いかけに自然となってしまいます。このことについては後で触れることにしますが，その前に，数学におけるこれらのマシンの影響に

17）原文では，calculator。ここで computer でなく，あえて calculator としたのは，擬人的な意味とコンピュータの将来像を当時の computer と区別するためと思われる。
18）この instruction tables は，命令表と訳してきたが，プログラム，現代で言えば機械語のプログラムのことであることがわかる。
19）コンパイラのようなシステムによって，実現されることになった。

ついてもう少し議論をしてゆきます。

私は，デジタル計算機が相当な関心を記号論理と数理哲学においても同様に刺激するものと期待しています。これらのマシンと人間がコミュニケーションをとる言語，すなわち命令表での言語は，ある種の記号論理の形式をしています。[20] マシンは言われたとおりに解釈して実行し，ユーモアのかけらも常識もありません。人間がマシンとコミュニケーションをするときには，厳密にその意味を与えなければ，その結果にはトラブルが付きまといます。実際に，人間はこれらのマシンとコミュニケーションすることがどんな言語を用いてもできますが，それは，言い換えると，いかなる記号論理を用いてもコミュニケーションすることができることでありますが，マシンにはその論理体系を解釈するための命令表が与えられていることが条件となります。このことは，おそらくもっと実用的な見通しが，論理体系について，これまでの過去のものを超えるようなものがでてくることになるでしょう。マシンを使って実際に数学的な式を操作するような試みがおそらくいくつも行われることになるでしょう。このためには，その目的に合った特別な論理体系が開発される必要があります。この体系は通常の数学的な手続きと似たようなものでなければなりませんが，同時にできるかぎり曖昧さをなくす必要があります。数理哲学については，マシンが次々と数学をやってゆくので，人々の関心はもっともっと根本的な哲学的な問題へと向かうでしょう。

ずっと言われてきたことですが，計算マシンは命令されたプロセスしか実行することはできないと言われています。これは，もしマシンが命令された以外の何かをしたなら，それはなんらかの誤りをしたとみなすのならば，たしかに真実です。このようなマシンを組み立てる意図が，手始めにマシンを奴隷とし

20）実際に Turing と同じ時期に計算可能性の問題を追及した Alonzo Church の lambda calculus をもとにした，プログラム言語 Lisp が作られたのはこの講演の 11 年後にあたる 1958 年のことで，MIT の John McCarthy による。Lisp はいわゆる高級言語としては 2 番目に古いもので（FORTRAN が 1 番目）であることからも，この講演での指摘は正しい。

てあつかう，つまり，些細なことまで考え抜かれたものだけをジョブとしてマシンに与える，あるいは，マシンの利用者が常に実行されていることを完全に理解しているというようなものをジョブとして与えるのは，それはそれで正しいことです。現在に至るまで，マシンはこのような形で利用されてきました。しかし，いつまでもそのような使い方をされ続けなければならないのでしょうか？ では，ある初期命令表をもち，正しく必要な時にはその命令表が自分自身を書き換えることができるようなマシンを，私たちがこしらえたとしましょう。そのマシンがしばらく運転された後のことを想像してみると，命令は私たちがまったく知らないものに変更されてしまっているのですが，そのマシンはそれでも，まだ役に立つ計算を続けていると人は認めざるを得ないのではないでしょうか。おそらく，それは，マシンが最初にこしらえられたときに期待された型の結果をまだ得つづけている可能性がありますし，もしかするともっと効率的な方法でやっているかも知れません。そのような事態になったとき，人は，元の命令を与えたときには予測できなかったマシンの進化を認めざるを得ないでしょう。それは先生から多くのことを学んだ後，さらに自分で学習してより多くのことをそれに付け加えた生徒のようなものです。このようなことが起こったならば，私は，そのマシンは知能を示すようになったと人がみなすのが当然だと感じています。人がそのための十分なメモリーを提供できるようになれば，ただちにこの線に沿った方向で実験をすることができます。人間の脳の容量は，おそらく 100 万ビットの 1 万倍のオーダーです。しかし，おそらくこの大半は映像の記憶に使われ，その他は，無駄になっているのでしょう。このことから数 100 万デジット[21] で真の進化が期待できます。それは，特に限定された分野での研究，たとえばチェスゲームなどの探求においてでしょう。おそらく平均的なプレイヤーに勝つ命令表を見つけるのは，割とやさしいのではないでしょうか。実際にベル研究所のシャノン[22] が私に簡単なルールでゲ

21）ここは，binary digits ではなく，digits と原文ではなっている。ビットではなく，文字。

ームに勝ったと話してくれました。ただし，彼の対戦相手の能力は明かされませんでしたが。しかし，私はそのような勝利にあまり意味があるとは考えていません。私たちが欲しいのは，経験から学習することができるマシンです。マシンが自分の命令を変えるようなら，そのためのメカニズムが要りますが，これについては，もちろん私たちはこれ以上分かりません。

次に，知性を持つマシンの概念については，根本的な矛盾があって，それを議論しなければなりません。「機械のように振舞う」ということは，適応能力がないことと同義語であるのは間違いないでしょう。しかし，合理的な説明はあいまいです。過去のマシンは，ほとんどメモリーの格納庫を持っていませんでしたから，マシンがそれ自体で判断を下すというようなことは論外でした。しかし，この議論は，より挑戦的な形式にしてしまえるのです。それは，たとえば，ある論理系において，その系の中で証明可能な式と証明不可能なものを区別するマシンは存在しないことが証明されたことです。これは，すなわち，命題をこの 2 つのクラスに分割するようにマシンを適用できるテストは存在しないということです。したがいまして，もし，マシンがこの目的のために製造されたなら，それは必ずある場合に答えを出せなくなるということです。一方で，数学者がこのような問題に直面したとすると，彼は探し続けやがて証明の新しい方法を発見するでしょう。したがって，任意のある式についての判定に結局は必ず到達するはずであります。これが先ほどの議論でしょう。それに対して，私はマシンにもフェアプレイでなければならないと主張します。時々答えを出せなくなるかわりに，私たちはマシンが時折間違った答えをするようにできます。しかし，数学者も同様に新しい技法を使おうとするとき，へまをしでかします。このへまを見逃して，彼にもう一度チャンスを与えることは容易ですが，マシンは容赦されることがありません。言い換えれば，もしマシンが間違いを犯さないとしたら，それはまた知的ではないことになります。数学の

22）ベル研究所の Claude Elwood Shannon は，現代情報理論の基礎をつくった。A. M. Turing よりもずっと長生きし 2001 年 2 月 26 日に 84 歳でこの世を去った。

定理で,ほとんどその通りだと主張するものが幾つかあります。しかしながら,これらの定理はマシンが間違いをするふりをしないかぎり,どれだけ知的かを示すことができません。マシンの IQ をテストするときには,私は「マシンのためのフェアプレイ」を続けて嘆願します。人間の数学者は,常に徹底的な訓練を受けてきています。この訓練は,マシンに命令表を入れるようなものとはどうやら違うものだ,とみなされるのではないでしょうか。したがいまして,人は,マシンが膨大な命令表をそれ自身の上に築き上げてゆくような期待をしてはいけないのです。人間は誰も知識そのものを追加してゆこうとはしないのにもかかわらず,なぜマシンをもっとたくさん欲しがるのでしょうか? 同じ問題を異なる観点からみましょう,マシンは人間とお互いの標準となるものにマシン自身を適応させるためには,人間とのコンタクトが許されなければならないのです。チェスゲームは,マシンの対戦相手の動きが自動的にこのコンタクトを提供しているということから,おそらくこの目的にかなり適ったものではないでしょうか。

参 考 文 献

参考文献の 106 ページから 124 ページまでに掲載されたチューリングの講演全文を訳出した。注は,原文にはまったくなく,すべて訳者の責任である。文中にもある ENIAC については比較的最近出版されたもので開発者の人間ドラマを描いたものを参考文献の [2] にあげた。現代情報理論の先駆者である Claude Shannon が 1949 年に書いたテキストは,現在でもまったくそのまま通用する不滅のテキストである。これを参考文献の [3] にあげた。Shannon との交流は,この講演中でもチューリングは触れているが,ENIAC の開発の数学的なバックグラウンドを設計した John von Neumann については,チューリングは講演の中でまったく言及していない。チューリングは,この講演の最後に人間と計算機の知性について短い議論をしている。3 年後にこの内容は,“Computing Machinery and Intelligence”として論文で発表された。Neumann も同じようなタイトル“The Computer and the Brain”で講演をしている。参考文献 [4]。これらは非常に興味深いが,これの内容については別の機会に論考する。

1 B. E. carpenter and R. W. Doran, eds., A. M. TURING'S ACE REPORT OF 1946 AND OTHER PAPERS, The MIT Press, Cambridge, Mass., 1986.

2 Scott McCartney, ENIAC The Triumphs and Tragedies of the World's First Computer, Berkley Books, New York, 1999.

3 Claude E. Shannon and Warren Weaver, THE MATHEMATICAL THEORY OF COMMUNICATION, University of Illinois Press, Urbana and Chicago, 1949.

4 John von Neumann with a Forward by Paul M. Churchland and Patricia S. Churchland, The Computer and the Brain, Yale University press, New Haven and London, 1958.