講　演

形　式　的　言　語*

——その歴史，現状および将来——

H. ツェマネク**　清　野　武　訳***

サンフランシスコに近いミューアウッズには，樹齢900年を超えるアメリカ杉の断面標本が置いてある．このような断面は，形式的構造と非形式的構造との間の関係を示すみごとな一例である．たしかに樹木の成長は極めて自然な過程であるが，しかもその結果は，円という形式的な観念に極めて近いものとなっている．この特別な場合には，年輪の系列は太陽を巡るわれわれの惑星——ついでながら，これら二つの球形物体はその外観が，またも，円という形式的な形状を示唆している——の円運動に関係づけられている．

ごく一般的にいって，形式的な表現はしばしば，非形式的なパターンのわずかな修正あるいは処理によってえられるのであるが，その共通の基盤は，（前者における）明確性と経済性から，また（後者における）単純性と労力最小化から来ているのである．非形式的な円は通常，平衡した過程の結果であり，形式的な円は，何か丸い閉じた形を記述するのに，最も短くて最も明確な，しかも最も単純な方法である——ほかのどの記述法も余分の情報を必要とする．

非形式的記述と形式的記述との間の密接なつながりは，形状に対して当てはまるだけではない．それは特に，コミュニケーションのためのわれわれの手段であり，また考えを表現するためのわれわれの能力である言語一般についてもいえるのである．言語の最も自然な形の中には，非形式的な面と形式的な面との注目すべき共存が認められるのである．

たとえば，文字は人体が極めて自然な方法で発生しうる音声によって示唆される．自然語は，組立てその他の人為的な手続きから創り出されたものではなく，その起源がすべて非形式的であるにもかかわらず，それは高度にディジタルな原理に従って，音声または文字から構成されている．綴りの誤りは，たとえそれを読む個人が，あまり訓練ないし教育を受けたことがなくても，構文的な修正を呼び起こすし，子供は意味を理解していない言葉でも，正しく話すことができる．

したがって，ある種の形式性およびある量の形式性は，自然であり，道理にかなっているように思われる．問題は，非形式と形式との間の適当なバランス，すなわち，どこまで形式性を押し出すべきか，である．この問いは二つの方向に向けることができる．その一つは，形式的誘導，すなわち存在する形式的構造からの形式的結論，とどこまでつき合うべきかであり，第二には，われわれが行なっていることの基盤の中に，形式的方法をどれだけ持ち込むべきかである．基盤が形式的によく定義されているほど，われわれがのちの形式的誘導に際して安全なことは明らかである．理想的な場合は全面的な公理化，すなわち構造を原子的な要素と，それらの間の論理的な結合にまで還元することである．しかし（実際の）仕事はつねに中ほどから出発する．われわれはアメリカ杉の断面とたいへんよく似た環境の中に生まれる：そこには形式的な観念が暗示されており，また自己暗示的な観念さえ存在するのであるが，はじめ観察したときには，それらの観念はあまり正確ではないのである．それらをもっと発展させると，しばしば一種の分離に導かれる——それは非形式的世界と形式的世界という二つの相反する世界への分離でさえありうるのである．摩擦と緊張と争いを避けるためには，形式化の効能と限界を理解し，またどのようにしてそれを現実の，本質的に非形式的な世界の中に，無害の形で埋め込むことができるかを理解することが必要である．このような理解は，現状を一瞥することよりも，歴史を考察することによって，より正しく与えられる．それゆえ私は，まず歴史を述べ，つぎに現状を一覧し——あなたがたの多く

* 1975年3月27日，東京で開催の情報処理学会講演会のための論文

** H. Zemanek, IBM 研究所（ウィーン），1971-1974年 IFIP 会長

*** 京都大学工学部情報工学教室　本会元会長

は，形式化の技術について私より詳しいと思うが，ここでは一般的な用語を使う——，そして最後に，将来の発展に対するいくつかの結論を引き出してみようと思う．

形式化の歴史

形式的な思考は技術と同じくらい古く，技術は人間の知性と同じくらい古いものである．しかし昔の表現手段は未熟だったので，あまり先へは進めなかった．したがって，カルディア人や古代ギリシャ人の形式的方法は，一般市民の日常生活にはほとんど役に立たなかった．昔の科学は哲学的性格のものにとどまったばかりでなく，形式的な建造物というよりも，つねに一種の芸術にとどまっていた．とはいえ，今日のわれわれの科学の基本的原理は，すべてその時代に据えられていたのである．

重要なのは，原子論と公理化についての二つの概念である．両者は十分詳しく考察するのに値するものである．なぜなら，2,000 年以上の科学の発展ののちにおいても，それらは依然として，自然を理解し，それを制御するために，われわれの知性が提供しうる最良のものであり，またそれらは情報処理の基本だからである．少数の基本的センテンスから，論理的法則に従って——たとえば幾何学のように——全世界を導出するという考え，ならびに，われわれの見るものはすべて，絶対的に論理的な法則の基盤の上で組合わされ，相互に影響しているという考えは，その本質において極度に形式的であるが，それらはギリシャの哲学者たちによって，自然語で表現されていた．ただ変数は記号で表わされ，数は形式的に（ただしまだ十進的ではなく）書かれていただけである．

数学が形式化され始めたのは，紀元 820 年以前ではなかった．この重大な一歩を踏み出した人は，当時コレズム (Khorezm) とよばれていたキヴァ (Khiva) 市（いまのウズベキスタン）に住んでいた一人のアラビア人 Abu Dshafar Muhammed Ibn Musa al-Khorizmi であった（この Khorezm というのは，アラビア語の母音を書くことのむずかしさから Khwarizm と綴られていたもので，この国の形式では，名前の最後の部分として市名が置かれているのである）．彼は，法律的身分の異なる 4 人までの妻を持った一人のアラビア人が死んで，妻たちや子供たちに大きな遺産を残したときに生ずる法律的な問題を記述しなければならなかった．遺産の分配はたいへんな数学的問題であって，ムハメッドが "*Kitāb al-jabr w'almuqābalah*" という標題の書物の中で，代数学を発明したのは，まさにこの種の問題のためだったのである．アルジェブラ (*algebra*) という術語はこの標題の *al-jabr* から，またアルゴリズム (*algorithm*) という術語は著者名（実は市の名 *al-Khorizmi*）から出たものなのである．この書物は 1,200 年頃ヨーロッパでラテン語に翻訳されたが，代数学が一つの形式的方法として真に受入れられたのは，16 世紀になってからである．その世紀にはソロバン主義者 (abacist) と算法主義者 (algorithmist) との間に，すなわち *calculi*（小石または硬貨のような計算用具）を使う具体的な計算と，紙の上でますます形式的になる法則を使っての抽象的な計算との間に争いが起った．算法主義者が勝利をえたのは，突如として，紙がはるかに低い価格で生産されるようになったからである——このことはまたもや，この種の精神的な面においてさえ，われわれがいかに多く技術に依存しているかを示している．

もう一つの重要な抽象化は建物の構造，すなわち建築様式の発展について起った．そこにも二つの流派の間で，すなわちイタリヤ様式とイギリス様式との間で争いがみられたのであるが，これは私がもっと詳しく調べたいと思っている問題の一つである．建築術の動向についてはまた立ち戻ることにして，ここでは建築学上の原理についての簡単な実例のみに限定しておく．その一つはイギリスとフランスの造園様式の間の違い，すなわちイギリス風庭園の非形式的な哲学と，フランス風公園の形式的技法との間の違いである．他の例はモスクワの赤の広場と，パリのヴォージュ広場との間の違い，すなわち絵画的なアンサンブルを形成するいろいろな建物の非対称的な組合せと，単一の建築家による広場全体の系統的なデザインとの間の違いである．これらの二つの図式にあと二つ，たとえばマンハッタンの任意に選んだ一連のブロックと，東京の一地区を加えれば，あなたがたは，技術がどんなに秩序正しく，あるいはどんなに粗暴に進むかについての考えを持つであろう．このことはわれわれの専門分野でも同じなのである．

抽象の世界の三番目は，設計が青写真に基づいて行われる機械の製作である．気がつかないかも知れないが，理想的な直線，円および他の要素から構成されている一枚の青写真が，製造すべき物体の形式的な定義なのである．この青写真の隅のどこかに，この物体を作るのにどんな材料を使うべきかの指定が書かれてい

る場合さえある.

形式化の世界はわれわれが見るとおり，単なる数学よりもはるかに大きいのであるが，数学について考えることにより，形式化の概念がどのように発展するかについて，ずっとよく理解することができる．数学はそれ自体の形式的定義の必然性を見出している．何世紀もの間にわたって形式的構造を使ってきたあとで，19 世紀に至り，それ自体の中での形式的構造の使用は，純潔な状況の保証ではない，ということが認識された——基本的論理と基本的観念の形式的定義が要求されるのである．数学史におけるこの一章は G. Boole の名で始まり，つづいて G. Frege の名が挙げられる．Russel と Whitehead は "*Principia Mathematica*" の中で数学の主要な部分を形式化するが，この本はまた，いかに多くの問題が残されているかを示している．Hilbert は彼のプログラムの中でこのことの弁明を与えたが，そのすぐあとで Gödel は，世界がかつて数学者が考えたほどには理想的でも秩序正しくもないことならびに，数値的数学のような世界でさえ，決定不能な点を有すること，を証明した．

ハードウェアはこの点に関してソフトウェアよりもずっとよくできている．なぜなら，ディジタルなハードウェアは，厳密に形式的な基盤の上で機能するからである．スイッチング代数学において，技術者が命題算を再発明した事実がある一方で，彼等はコンピュータ用のスイッチング回路を，数学の普偏的基礎——真にそれを知ることなしに——の上に立って作り，またそうすることによって，そもそもの始めから，コンピュータの普遍性を準備したのであった．

スイッチング代数学が日本で起ったことを知っているヨーロッパ人やアメリカ人は多くないが，榛沢正男と中島章が彼等の最初の論文を公表したのは 1936 年であって，これは 1938 年の Shannon の論文に先立つこと2年である．

スイッチング回路の形式的な性格のおかげで，ハードウェア技術者が自動設計と，それに続く自動生産へと向うためには，構造の哲学を少し変えるだけでよかった．小型化は形式的方法の必要性を増大し，ハードウェアを先端的分野へと力強く押し進めた．このことは，いかにハードウェアがソフトウェアに対するモデルケースとなったか，ということである．プログラマは形式的なテキストを使って仕事をするが，彼等はあまりにしばしば高度に非形式的な（不合理な，というつもりはない）方法を使うために，自動設計や自動生産がますます困難になるような事態を作り出しているのである．

私はあとでこれらの問題に戻ることにする．

数学以外での形式化

コンピュータ時代以前に形式化された分野はそれほど多くはないが，しかしそれが数学だけに限られていたわけでは決してないのである．二三の例をみればそれが明らかになるであろう．

楽譜が本当の形式言語であることを知っている人は少ししかないし，まして，音楽を二進的な関係で考える人はもっと少ない．ところが，ポピュラな歌は，童謡でもヒット曲でも，大部分が 16 小節の構造を基礎にしていることは，誰にもすぐわかる通りである．たとえば，

"Hänschen klein, ging allein"
"Ein Männlein steht im Walde"
"Weisst Du, wieviel Sternlein stehen"
"Ich fahr' mit der Post"

あるいは，

"Jack and Jill went up the hill"
"Little Jack Horner"
"Pat a Cake, Bakersman"

のような童謡，および

"Hüo ho, alter Schimmel, hüo ho"

のような歌は，いずれも完全な 16 小節構造を示している．

しかし音楽の二進的性格は，もっと先まで進んでいるのであって，それを私は作曲の抽象的アーキテクチャと呼んでいる．実際に私は，ベートーヴェンの第6交響楽（"田園"）の第1楽章が正確に 512 小節であるという事実を知って驚いたのである．それは九つの交響楽のうちで，小節数が正確に2の冪に一致するただ一つの楽章であることもわかったのであるが，しかし多くの楽章の小節数は2の冪に等しい二つの数の和であるか，あるいは二進の丸数に近いのである．ベートーヴェンや他の多くの作曲家は8または4小節の構成単位を好んだし，また 64 小節のブロックは，作曲における比較的大きなブロックのうちで最も頻度の高いもののように思われる．もちろん芸術家は決して——理性的に，あるいは直感的に——偏りも例外もない抽象的スキームを辿ったわけではない．しかし，たとえばヘンデルのハレルヤから "Halleluja" の叫びを取り出してみるがよい．そこには完全に二進的作曲が残るの

である．

音楽の二進的性格に対する説明は，かなりの部分まで，作曲の原理における対象性から与えられる．すなわち，1ビットを加えるごとに，もう一つの対象性がえられるのである．作曲ならびに作曲のアーキテクチャを二進的に眺めることによって，さらに多くの結果がえられることが期待される．

音楽のディジタルな特性は，音楽を自動的に発生するためのディジタルな自動機械が，600年以上も前から使われてきたという事実からも導かれるかも知れない．形式的な演奏については，チャイムから硬化式自動ピアノに至るまで，長い歴史がある．エディソンの蓄音器は，音楽のアナログ的な蓄積とその再生という新しい世紀を開いたのであるが，それほど遠くない将来に，ディジタルなシステムが取って代わるであろうと私は確信している．

二進的な工芸および芸術のもう一つの例は織物である——そしてこれもまたオートメーションの出発点となった分野の一つである．織機のために発明された穿孔カードは，どの民族の間でも何千年にわたって使われてきた織物の装置のただ一つの組織的な改良なのである．私の同僚の一人は最近，高部オーストリアの一地方の博物館で，非常に古い時代における（カードに至るまでの）中間的なステップを発見した．それは麻糸の閉じたループからなる一つのプログラミング単位で，その上で横木が織機の操作を制御するようになっている．この装置は1740年に作られたものであるが，その発明はたぶん1690年以前にさかのぼるであろう．

織物の二進的性格の説明は，2本の糸の交わりがディジタルな織模様の中の1点をきめるということである．数学的思考は，人間のすべての思考と同じく，時系列的である．コンピュータや織機は決して時系列にしばられるものではない——コンピュータのプログラミングが，どうすれば過剰な時系列化から逃れられるかは，たぶん織機ならびに織物のプログラミングから学び取ることができるのではないだろうか．

形式化の第3の例は簿記であるが，これも何世紀にもわたる形式的な伝統をもっている．簿記係は，本質的には書式の延長である彼の帳簿の正しさのみに専念する．彼は“貸方”と“借方”の両方の欄についての総和が同じであることを主張するが，彼のデータの意味の検証は帳簿の外部に残している——これはまさにプログラマがコンピュータについて行っていることと同じである．

同様に複雑な現実を簡単な形式に，続いて簡単な処理に還元するのは人口登録すなわち国勢調査である——これもまた自動計算のもう一つの原点である．

1889年にHermann Hollerithは，国勢調査の処理のために設計された穿孔カードシステムに関する特許を申請した．そして1890年における合衆国の国勢調査は自動的に作業された最初のものとなった——むしろ，最初に行なわれた二つのうちの一つ，というべきである，なぜなら，1890年のオーストリアの国勢調査も，やはりホレリス機械で実施されたからである．この機械は，1896年に始めて“コンピュータ”のプログラミングに関する特許をえたOtto Schäfflerによって輸入され，改良されたものである．私はこのオーストリアの先駆者の一生を再現するのに3年を費した．

私は前に青写真のことを述べたが，この青写真を形式的に定義しようとの企ては，以前にもあったのである．すなわち1907年にスペインの先駆者Leonardo Torres y Quevedoはウィーンで開かれた科学アカデミー代表者の会合に論文を提出して，機械的な構造の形式的な記述のための記法に関する提案を行なった．一例として彼は，1年前にその説明が出版された一つの小さな機械，すなわち二つの複素数の積を自動的に計算するための装置，を形式的に定義した．この形式的定義はほとんど今日のAPTのようにみえるものであって，そのとき提案された言語では，形的式記述は31の式からなり，その中にはこの装置の箱まで含まれていた．Torresは彼の論文の中で，形式的な定義のため論拠を示しているが，それは60年たった今日でも，依然として正しいものである（アカデミーはこの論文を受理しなかった）．

形式化に対する最後の例として，私は科学一般——科学の哲学を含む——を取りあげたいと思う．

200年あるいは300年このかた，科学と工学は宇宙の形式的モデルと呼びうるものの上で活動してきた．その方法は，基本的には原子論および公理化という古代ギリシャの概念である．今世紀の最初の20年間に，この方法は成功の頂点に達した．RussellとWhiteheadは数学を形式的に定義した．物理学と化学において原子は確立された現実となり，自然の法則は物質のあらゆる種類を支配するためだけではなく，同じく原子的性格を有することが証明されるはずの，あらゆる種類のエネルギーをも支配するために使われてきた．心理学においては“観念連合説”と呼ばれる方向が最も優勢であった．それは，われわれが感じ，記憶

し，考えるものはすべて，原子的な知覚入力（知覚要素）に基づくものであり，観念連合の諸法則によるそれらの組合せから成り立っていることを教えている．1910 年頃の科学の状況は，それを原子的ならびに論理的性格の科学の哲学にまで一般化してくれる誰かを求めていた．

実際のそのような人が現れたのである．それは前述の Hollerith が特許を申請した年に生れたウィーンの若い技術者 Ludwig Wittgenstein であった．第一次世界大戦の終り頃，Wittgenstein は "*Tractatus Logico-Philosophicus*" と題する論文を発表した．これは，論理および論理的原子という言葉によっての宇宙の形式的定義のためのアルゴリズムにほかならない．これらの原子を彼は要素的センテンスとなづけた（ウィーン・サークルはのちにプロトコル・センテンスという術語を使った）．彼の一般的アルゴリズムはつぎのように述べることができる：すべての要素的センテンスについて可能なすべての論理的組合せを試みよ；それらが事実に則して真であるか偽であるかという検証によってチェックせよ；事実として偽であるものを捨て，真であるものを集めよ；そうすれば，完全にして完璧な宇宙の形式的記述をうるであろう．このアルゴリズムから Wittgenstein は，いみじくも哲学の終りを結論した．その理由は，もしこの原理を働かせれば，そこにはそれ以上考えるべき何物もない，ということである．そしてこの *Tractatus* の最後の文章は，その帰結としてつぎのように述べている：われわれの語ることのできないことについては，沈黙しなければならない．

Wittgenstein の誘導は間違っていなかったのであるが，彼の仮定のあるものは間違っていた．ここで，Gödel の名を挙げて，すべての論理的な決定プロセスが一つの決定に達するとは限らない，ということを主張してもよいであろう．あるいは検証のむずかしさについて語るのもよいであろう．しかし，はるかに本質的なのは，論理の外には原子がないという事実である．われわれはこのことを物理学および化学から教えられている．そこでは，最終的には小さな粒子のクラスに達するが，それは物理的宇宙の完全な記述を約束さえもしないのである．同じ事実が心理学に対しても確立されうる．とりわけそれは言語に対して真である．言語の最小単位（たとえば一つの身振り）の中にさえ，われわれは長い物語りを見出したり，含めたりすることができる．Wittgenstein はこの事実を 1933 年頃に認め，哲学Ⅱを出発させたが，その中では，たとえば一つの語の意味は，それが中で使われている言語の目的に依存している．

計算機科学にとって重要なのは，ディジタル・コンピュータが完全にこの *Tractatus* の世界を具体化しているという事実である．コンピュータ・システムの中で起ることは何でも，情報の本当のアトム——ビット——の組合せと系列であり，ビットによって言い表わせないことについては，コンピュータは本当に沈黙を守るのである．それを明確にするために，Wittgenstein の哲学Ⅱの言葉でいえば，プログラムの意味は，それが用いられるコンピュータの目的に依存している．われわれのシステムの中の形式的な宇宙と非形式的な現実との間には，またプログラムの領域とアルゴリズムとの間には，さらに人々の生活と共同体との間には，永久にギャップが存在するであろう——そのギャップは，機械的で形式的な道具より以前に，またそのような道具の外部で，人間によって橋がかけられることを待っているのである．

コンピュータは数学の一部の分野の中でも，また数学の外部でも，形式化を促進し，さらに——単純に，コンピュータ自身が形式的であるという事実によって——形式化を実際上も必要なものとした．

処理の単位である命令語のセット，すなわち機械語は，電子技術者が用意したスイッチング回路によって規定された構文と意味とをもっている絶対的に形式的なシステムである．そして電子技術者は，実際的な問題を解きうる道具を必要としていた数学者によって，プログラムを与えられたのである．コンピュータの汎用性は，全面的に実用主義的な動機づけからの結果——起こりえた最良の場合——である．

命令語のセットからプログラミング言語への道を歩んだのも，本質的には実用的なステップである．流れ図の考えは，プログラムの一般的な線を——任意の細部に至るまで——はっきりさせるのに役立った．このことに関連して述べなければならないのは，Zuse の "Plankalkül" である．これは数学をはるかに超えた完全に形式的な言語であって，Zuse はチェスのゲームを彼の形式で書いて，このことを証明した．残念なのは，彼がこの研究を続けなかったこと，ならびにそれに対してあまり注意が払われなかったことである．それはプログラミング言語の開発を著しく促進していたかも知れないのである．式の変換を自動化するためのアメリカ人の努力とは独立に，ヨーロッパで算法的

言語の口火を切ったのは Rutishauser の "*Rechenplanfertigung*" においてであり，アメリカでの成果は FORTRAN，後者の成果は ALGOL であった．

言語の理論

この論文で私は，(標題である) 形式言語よりも，形式化について多くを語っている．しかしこれは十分に許されることである．なぜなら，形式化は形式言語にくらべて，より一般的な観念だからである．われわれが慣れているような種類の形式言語を設計することなしに，形式化を行なうことは，たしかに可能である．しかし，ひとたび言語という観念を考察すれば，その下限を定義する方法のないことを，直ちに見出すであろう——記号の形式的なシステムはどんな種類でも一つの言語であり，どんな種類の形式化も，ある種の言語を生み出すのである．

本来なら，この論文のこの点で，私は実際に割込みをかけ，言語の理論に関する副論文を挿入すべきである——しかしそれは容易に本文と同じくらい長くなってしまうであろう．それ故，私はただ二三のキーワードを示すにとどめ，あなたがたが，この理論についてすでに知っているか，あるいは，間もなくそれについて読む機会のあることを仮定したいと思う．私には，それ以外の選択の道がないのである．

Peirce, Moris およびウィーン・サークルによって展開されたような科学的言語の理論は，三つのレベルの研究を区別しているが，私はさらにレベル 0 を加えたいと思う．それは歴史的に，つねに言語の記述における最初の項目であったアルファベットである．ここにわれわれは言語が原子的な概念から出発していることを知るのであるが，それはしかし，その本質的な内容に対しては役に立たないのである．

任意のアルファベットに対して一つのコードを組立てるために使える真の原子がビットだけであることをわれわれは知っている．したがって，文字は分子——文字ごとに異なるビットの組合せ——のように考えることができる．つぎのレベルは一つの中間的なもので，それには番号をつけないが，このレベルでは，文字の組合せが語を生み出す——これは人間の知性と同様に神秘的な言語の単位であって，聖書が Verbum という語に付与しているような特別の意味を支えている．一つの語を符号化することはごく容易であるが，その意味を捕えること，伝統的な語の意味するところを形式的に定義すること，はごく困難である．組立てられた意味のみが，形式的定義の対象となる．

研究の本来のレベルは構文論，意味論および慣用論 (pragmatics) である．構文の最も純粋な定義では，それはすべての well-formed sentence を定義するための規則の集合である．それはいかなる意味からも絶対的に独立して，文字または語の組合せを記述する．よく構成された言語の完全な構文は，文章が well-formed であるかどうかについての機械的なチェック，ならびに構文的な誤りの排除を可能ならしめるものである．

意味論はセンテンス (well-formed sentence) の意味を研究する．理想的な原理は，各センテンスの事実に則した正しさを検証することであろう．*Tractatus* は (またウィーン・サークルおよび論理的実証主義は，この *Tractatus* と共に)，検証がセンテンスに対して (少なくとも原理的には) 可能であると仮定したが，時がたつにつれて，それを行う有効なプログラムは，多くの困難によって拒否されてしまうことがわかった．オーストリア系イギリス人の哲学者 Sir Karl Popper は，検証が不可能なのだから，せいぜいセンテンスが偽りであることを示すことを試みるように強制されている，という結論に達した．この偽りを実証することに耐えたセンテンスだけが，われわれの一般的な知識の籠の中に残ることを許される．そして，形式的なシステムの内部での同義語反復的な真を除いては，最終的な真は存在しないのである．

自然言語の自動翻訳の失敗は，まさにこの種の困難に関連している．論理学および数学——およびコンピューター——という組立てられた世界の中においてのみわれわれは安全な地上にいることができる．そこでは意味を形式的に定義すること，および意味論的な正しさを証明することの機会をわれわれは持っているのである．

言語の理論の第 3 のレベルは慣用論——構文論および意味論ではカバーされないすべてのもの，たとえば言語の用法，その使用者，流行語や方言，言語の歴史 (もし適用できるとすれば)，など——である．換言すれば，設計された言語を使って達成できるものが慣用論なのである．このことは再び，慣用論が言語研究の始まりであり，終りであること，また構文論や意味論はその中間部分にすぎないこと，を示している．第二に，慣用論を形式化する望みのないことは明らかであるから，構文的ならびに意味論的構造が，本質的には現実生活の中に埋め込まれていることを，言語研究は

重ねて明示しているのである．われわれは何事についても，全体的な形式化を達成することは決してできないであろう．

ここで自然語と人工語の間の区別——私がすでに使ったことのある区別——について簡単に述べておこう．それは，自然言語は生きたり死んだりすることができる，ということである．世の中には自然言語と形式言語との混合物が存在する．その例は医学的言語であって，そのうちの若干の形式的な部分，たとえば体温の指示は，他のすべての物理的測定値と同様に，形式的な処理，したがってコンピュータによる処理を受けることができる．医学的知識の多くの，しばしば非常に進んだ数学的構造は，どんな工学的構造にも劣らず形式的であるが，医学には別の部分がある．たとえば，高度に非形式的な診療記録の中にみられるものがそれである．それらを保存や再生以外の自動処理にゆだねることは極度に危険である．今日，全面的に形式化され，計算機化された医療に身をまかせたいと思うものが一人もいないことは確かである．このことは医学の情報処理の重要問題についての一つのアイデアを示している．

最後に私は言語，メタ言語，メターメタ言語，以下同様（meta$^{(n)}$ 言語）の間の区別について述べなければならない．自然言語が任意レベルのメタ言語として使用されうることは明らかである．未解決の問題は，われわれがいつの日か，それ自身で記述できる程度にまで回帰的な，したがって最終的なメタ言語として自然言語を使わなくてすむような，形式言語を知るかどうかということである．いままでのところ，私はまだこれを達成する有望な企てのあることを知らない．

さてここで，コンピュータにからんで重要性を持つに至った形式化の三つのレベルの検討に戻ることにしよう．これら三つのレベルとはつぎのものである．

（1） 形式的記法
（2） 形式的定義
（3） 正しさの証明

第1のレベルは長い間にわたって使われてきたものである．それは，機械的誘導を許し，したがってその結果について一般的な合意に達するこのとの可能な，形式的言語の導入である．もしその規則が正しく守られるならば，その結果はすべての人に受入れられるはずである．挑戦を受けるとすれば，それはその仮定あるいは，ある条件のもとにおいて，そこに使われた観念である．これらの観念は，もし矛盾すなわちパラドクスを導くことが可能なとき，攻撃されるべきである．このとき，つぎのレベル，すなわち観念の形式的定義が導入されなければならない．これは——繰り返していうが——人工的言語の中においてのみ可能である．人工的言語はさらに，それらの知識が母国語で習得できるという利点をもっている．したがって，形式的な世界では，コミュニケーションは外国語の（コンピュータの世界では英語の）コマンドとは無関係になり，また，一人の著者の過剰な語学的技巧（これは結局——つねにではないことを私も認めるが——多くの場合，反啓蒙主義と同じである）とも無関係になるのである．

形式化の第2のレベル，すなわち応用形式言語の形式的定義は，実際上つねに，抽象的機械——最もよく知られている例は Turing 機械である——を使って行われる．

抽象的機械はいろいろな状態（状態の集合）をとることができる，また完全に定義された遷移関数に基づいて，一つの状態からつぎの状態に移行する．一つの遷移の間にすべてのパラメータが変化するという理由はないので——むしろ遷移は多くの場合，全体の構造のうちの小さな部分構造のみに関して起こるので——状態を部分状態の複合システムとして組織すること，ならびに，状態（部分状態）および入力（または，つぎに実行されるべきテキストの部分）によって条件づけられた遷移関数の全集合を導入することが便利である．特別の制御機構（それもこの抽象機械の一部でありうる）がこれらのステップの正しい系列を維持する．

抽象的機械は，形式的言語あるいは形式的システムの構文および意味を，オペレーショナルな，あるいはアキシオマティックな原理に基づいて，定義することができる．ウィーンの IBM 研究所が PL/I の意味論について，一つの記法，定義の方法を設計し，そして最後に形式的定義を行ったことは，あなたがたすべてがすでに知っておられるとおりである．したがって私が話しているのは，可能性のことではなく，現実の結果（しかも無意味でないような規模の結果）なのである．

しかし形式的定義は形式化ゲームの終りではない．つぎの（第3の）レベルを特徴づけるのは，解の正しさの証明である．人間の思考はナノ秒単位のステップで走る過程を監視することはできないから，間違った仮定が過程の細部から見つけ出される機会は全くない．もちろんわれわれは，われわれのプログラムの中ではすべての可能性が事前に認められているので，未

知の事が観察されるはずがないとうぬぼれている．しかし，本当にそうであることがつねに確信できるであろうか．純粋に人間的なオペレーションにおいてすら，出された解についてはかなり信用できるのであって，多くの場合，一つの解に異議を唱えることはむずかしいであろう．なぜなら，状況全体が暗黒のままの非形式的な成分からできているからである（ときには，解が用意されていて，それに当てはまる問題が未知である場合がある．このような場合における一つの論理はつぎのようなものである．答よりも問いのほうがたくさんあるので，問いとは独立に答を考えるほうが経済的である……）．

正しさの証明は，形式的な問題と形式的な答との間の鎖を形成する．それは答が本当に問いに適合すること，その解が本当にその問題を解決していること，を明らかにする．しかしながら，この目的に対しては，形式的な記法と形式的な定義とは，必要条件なのであって，それだけでは十分でない．解の過程に対する形式的な環境が確立されていなければならない．そのようなときに正しさの証明がなされるならば，利用者ははじめて形式的システムの信頼性を享受することができるのである．

これが形式化の最高のレベルであるが，それはオペレーションについて極度に透明な場を必要とする．そして計算機科学に関するすべての著者がこのために闘っているとは絶対にいえないのである．この場合の悲劇は，極めてわかりにくい経路に沿う形式性のために闘うことも可能である，ということである．

一方において，容易には論破できないような，見掛け上は確固たる主張をもって，非形式性のために闘うことも可能である．自然言語によるプログラミングはたいへん魅力的な，たいへん役立ちそうな，たいへん親しみ易い響きをもっている．なぜそれが可能にならないのか．それは一般的な利用者のための解決とはなりえないのか．自然言語は利用者が知っているものであり，利用者が（ちょうどプログラミング言語のように）おぼえる必要のあるものである．にもかかわらず，自然言語によるプログラミングは——取るに足りない応用を除けば——素朴な間違いである．しかもこの間違いを犯し，あまつさえ育生しているのは，最高に知性的な情報科学者なのである．

もし問題が利用者の頭脳を超えたならば，助けになりうる者は——もしあるとしても——プログラミングの専門家ではない．なぜなら，最も知能の高い専門家といえども，彼にわかっていることしかプログラムできないからである．

形式的言語は，より弱い知能の無器用なスタイルに対しては一つの助けである．しかしどんな場合にも，無器用な非形式的スタイルが，形式的言語のコマンドの欠除に対する助けとなることはないのである．

はっきり考えることができ，それを表現することのできる人なら誰でも，データ，エネルギーおよび物質の流れを，自動的な機能によって要求される精度で定義するために，プログラミング言語を学習することができる．この能力が欠けていれば，最も進んだコンピュータプログラムでもついていけないような記述あるいは命令を生み出すであろう．それでコンピュータが正しいことを行うとすれば，それは単なる偶然（しかもわずかしなチャンス）によるのである．

残るただ一つの出口は“姿なきプログラミング”ともいうべきもの，すなわちプログラムは全面的に箱の中にあるが，プロセスは——簡単なものも複雑なものも——些細な行動（磁気カードまたは硬貨を入れること，重みをかけること）によって開始される，というような自動システムの設計である．

コンピュータは，もし入力側に知能があるとすれば，知能の増幅器である．コンピュータは，もし無知によって指令されるならば，反知能の増幅器である．

コンピュータの利用者を不要の学習から保護することは一つの良い目標であり，たぶん一つの必要である（技術一般にいえることであるが，とりわけコンピュータ技術は，学習について利用者に過大な負担をかけている．それは多数に対してもいえるが，特に少数の人にますます過重になっている）．しかしながら，自動装置を扱う場合には，一種の心理的経済というものがあって，それからあまり遠くへは離れられないのである．コンピュータの疑似知性的外見から引出されるのは欺瞞だけである．

形式化に対する賛成論と反対論

形式化に対する賛成と反対のキーワードとしては，つぎのようなものを挙げることができる．

（賛　成）	（反　対）
明　確	明　確
経　済	学　習
安　全	危　険
自　由	自　由
一般性，エレガンス	高価，非実際的

明瞭性と非曖昧性とは，形式化の最も重要な特質である．形式的システムの抽象的観念は，自然言語の中で使われる具体的な観念とは違っている．前者は言外の意味とか，暗黙の仮定などを含まない——それは当然いわれるべきものを，それ以上でもなく，以下でもなしに，表現しうるだけである．

しかしこのことこそまさに，形式化に反対する論拠になのである．過度の明確性は，関連するすべてのものを結びつけ，その量はしばしば極めて不都合となり，少なくとも煩わしいものである．もちろん形式的な方法にも，あとで指定されるべき部分を残しておき，正確さを失わないようなものもある——．しかしそのような方法は，ある知識，すなわち容易には得られないような形式化のコマンドを必要とする．

もっと重要なものとして，非形式的な表現は形式的な表現にくらべて，より多くの情報を支えている，という議論がある——これは真実である．前に挙げた医学の例を考えてみよう．医学的記述や医学的知識は，分析され，形式的システムの中に変換されうるが，疑いもなくこの過程のうちで，いくらかの情報が失われる．明確性は実際にネガティブな論拠となりうるのである．

表現の簡潔さ，機械的単純化および再現性の反映である経済性は，明らかに強力な論拠である．特に再現性は節約に対する基盤を生み出し，またつぎのような自動化の前提条件でもある：

自動プログラミング
自動ジェネレーション
自動シミュレーション
自動ドキュメンテーション

経済性に対する反論はつぎのようなものである．たしかに上の議論はすべて正しい；しかしそれは生産の際にも，応用する側でも，かなりの量の教育を必要とし，本来の利点をそこなうことになる．これは実際には一つの重要な点なのであるが，熱心な形式化論者がしばしば見逃しているところである．

形式的な扱いによって得られる安全性および信頼性に対しても，上と同様の二面的な議論がみられる．構造は例外を含まなくなり，機械的評価も完全な構文もすべて良いことばかりである．解釈は形式的構造の外部で行なわれ，生きたどのような言語の個人的なコマンドとも無関係になる．形式化されない場合には，ものごとはそのような生きた言語で処理されなければならないのである．

これに対する反対論はまたもや，新しい原理には危険が伴うことを主張する．そのような原理が本当にうまくゆくということは誰も保証することはできないので，全体としての利得は疑問である．

形式化のもう一つの優れた性質は，一般性およびエレガンスである．問題の状況は，発展の初期の段階からも事前に理解され，概念の集合への恣意的な参加は存在しない．またそれらの概念はいずれも，最も一般的な形でのみ容認される．何かについて働いているものは，それが適用される分野ならどこでも応用することができる．

これに対する反論は，一般性は高価につくということである——それは金銭についてだけでなく，たとえばコンパイル時間などについてもいえることである．エレガンスはしばしば非実際的であり，事実，極めて実際的なものは高度に非エレガントである．多くの人々により，特殊な解よりも一般的な解が選ばれるのは，経済性が究極的な理由ではない．もしあなたが大きな商売をやろうと思えば，市場にある他のすべてのものに共通した機能を含む一般化されたボタンによるよりも，市場にあるものとは違った特別なパテントのボタンによって試みるであろう．

最後に，自由は形式化によって，その後のステップのために開かれるものである．なぜなら，システムは理想的に透明であり，すべての可能性が，長い試みなしに，見通されるからである．しかしこの議論は，形式化が以前の記述を再解釈する自由を排除するということ，すなわち“あなたは私を誤解している．私が本当にいおうとしたのはつぎのことなのだ．……”と述べる可能性を除いてしまうということ，を主張して，逆転されるかも知れない．

もちろんこのような再解釈を設計原理として売出す者はないであろうが，しかし直観的には，それはしばしば形式と非形式との間の争いにおける一つの力となりうるものであろう．

抽象的アーキテクチャ

これらの議論が一緒になると，設計の基本原理の一つが導き出される．それはアーキテクチャという名前をもっているが，私は抽象的アーキテクチャとよぶことにする．それはアーキテクチャという術語が現在意味しているもの——すなわち多くの部分をつなぎ合わせるずさんな方法；以前には論理設計とよばれていたもの——と区別するためである．

つぎのような問題も興味があると思う．大英百科辞典を見て，建築技師がアーキテクチャという術語により，何を意味したかを読み取ることである．それは

文明的な人々の実際的で，かつ表明された
要求を満たすようなビルディングの技法と
テクノロジー

として定義されている．その主なゴールは

人類が使用するのに適していること，
その構成の安定性と永続性，
形態による経験と思想の伝達

である．

これらのすべての考えは，われわれが行っていること，あるいは行うべきことに完全に適用できないだろうか．

この百科辞典のアーキテクチャというキーワードに対する説明の項目は，つぎのようになっている．

利用——タイプ
　——計画
技術——材料
　——方法
表現——内容
　——形式

また，“需要のない（あるいは潜在的需要だけの）仕事は，経済がそれぞれ防止する”とも書かれている．

コンピュータ・アーキテクチャという概念および術語は，1962 年にコンピュータ “Stretch” に関する本の中で導入されたのが始めであり，F. B. Brooks がそこで与えた定義は全く明らかで申分のないものである．それは IBM システム/360 のハードウェア設計に応用されたが，この術語と考え方は，間もなく文献から姿を消してしまった．それ以後アーキテクチャの概念について公表したのは，システム/360 の創始者の一人であって，彼は何が良いアーキテクチャを作るかを指摘している．彼によれば，設計は三つのステップ

アーキテクチャ，
インプレメンテーションおよび
リアライゼーション

に別けられる．アーキテクチャは利用者にとってのシステムの機能的な姿（何が起こるか）である．インプレメンテーションはアーキテクチャを実施する論理構造——その詳細——（それはいかにして起こるか）である．そして，リアライゼーションはハードウェアの構成（それはどこで起こるか）である．

鍵となる思想は無撞着性（consistency）である．もしアーキテクチャがコンシステントならば，システムの部分的な知識だけで残りの予測が可能になる（このことは私が，抽象的アーキテクチャを創造的冗長性として定義したいと思う理由である——反復と対称が物を作る原理であるのは，何もテクノロジーの世界に限ったことではなく，芸術，特に音楽においてもそうなのである）．

良いアーキテクチャはまた，直交性，妥当性および一般性によっても特徴づけられる．直交性とは，本質的な要求に固有の概念や機能の間の不必要な結合がないことを意味している．一般性とは，導入されなければならない個々の概念，性質または機能が，その最も一般的な形で導入されていることを意味する．

良いアーキテクチャは，開放性と完全性を示すであろう．前者は，不必要な制約がなく，また後日の，および将来の発展に対する十分なスペースのあることを意味する．後者は任意選択のないことを意味する——もし集合の一部であるならば，それは “full set” である．その他のキーワードとしては，対称性，透明性，互換性，信頼性などがある．良いアーキテクチャは激励的，自己教育的である．

もちろん，完全なアーキテクチャが設計はされても，最終的には貧弱な性能という性質を示す危険がある——しかしそのような悪い結果は，もしも全設計がきれいな形式的方法の利点の上に基礎づけられているならば，起こりえないであろう．換言すれば，システム・アーキテクチャは，徹底的な形式化を必要とするのである．それは “自然の進化” にまかせておくわけにはゆかないのである．そしてこの議論は “一般システム理論” との興味ある関係を開くものである——しかし私はこの理論にはこれ以上深入りしないことにする．

形式的言語の将来

私は形式的言語が形式化の問題の中にどのように埋もれているかを示したいと思う．事実，形式的言語およびプログラミング言語の現状は，言語問題だけというよりも，むしろ形式化の問題とみなすことを支持している．単一の万能プログラミング言語が，プログラミングの問題の大部分を解決してくれるであろうという夢は，プログラミング言語，システム言語，制御言語その他さまざまな言語の洪水の下で消え去ってしまったし，またわれわれはバビロン（バベルの塔）のよ

うな言語の混乱の中で，何とか生きてゆくことをおぼえたのであるから，形式的言語の将来はすでに 1966 年に Landin 教授が予言し，そして恐れたような，何百というプログラミング言語（“つぎに来る 700 のプログラミング言語”）どころではなく，さまざまな種類の何千という形式的言語の極めて荒々しい流れとなるであろう．

それらを理解し，最後にそれらを支配することは，形式化一般の発達の極めて深い理解によって始めて可能となるであろう．そしてそれは，形式化の本質とその原理についての深い理解を必要とするのである．

プログラミング言語の将来は，5年あるいは 10 年前に誰もが考えたよりもずっと強く，過去の投資からの影響を受けるであろう．誰かが一つのプログラミング言語を勉強したら，彼は改めて“より良い”言語へ移行するよりもむしろ，以前の言語で自己を表現しようとするであろう．そればかりでなく，すでに書かれた一群のプログラムが存在するということは，どんな新しいハードウェアおよびソフトウェア・システムの上でも，それが使えるように維持しようとする強力な経済的理由になるであろう．換言すれば，われわれは現在の言語の多様性と共に生きてゆかなければならないのであって，かりに本当の万能プログラミング言語が使えるようになるとしても，それは遠い将来のことであろうから，それについて今日多くを語ることはできない．

それ故，この論文の最後の部分では形式化の将来について扱うことにする．またそれによって，プログラミング言語および他のあらゆる種類の形式的言語の将来も，最もよく考察されることになるであろう．

形式化の将来

形式化は，少数の特別な専門家の間の隠者的，秘儀的な娯楽，理論のための理論ではない，それは間もなく，

生産の道具，
管理の道具，
応用の道具

となるであろう．

コンピュータは，直観によって，あるいは行き当りばったりの方法によって，生産し，組織化し，応用するには，あまりにも強力にすぎる．情報処理は，装置が速くなり，大きくなり，安くなるほど，あらゆるレイアウトにおいて，より高度の精密さを要求する——この強力な道具が必要とする緊密な制御は，機械化を通してのみ得られるものである．そうでなければ，その成長において，またその無限の応用において，組織的にも財政的にも，道を誤ることになるであろう．

形式化はそれ故，本来の情報処理装置の環境にまで拡がるであろう．中でコンピュータが使われるシステム全体の形式的定義が必要となるであろう．全企業，全機関あるいは他のあらゆる組織——機械的な部分だけでなく，その人間的な部分までを含む——を，その自然の構造に至るまでも表現するモデルとなるような抽象的機械が必要であろう．

これは形式化の問題を順次に鮮明にしてゆくことであろう．多くの問いが示されるであろう．形式的モデルと有機体との間の関係は何か．あらゆる人工言語と自然言語との間の関係は何か．私は前に，複雑な性質の例として医学的な構造について述べた——複雑な状況に対するもう一つの例として法律的な構造がある．法律的言語および法律的思考は，ある種の形式的性格を持ってはいるが，形式化についての法律的哲学は，論理的ならびに技術的形式化とは本質的に違っている．コンピュータに法律的構造を入れることは極めて必要なことであるが，論理的思考と法律的思考の間のギャップの故に，それは極度に困難である．形式化とコンピュータ応用に対する問題の領域について，ほかにもたくさんの例を挙げることができる．われわれの前には，たいへんな分量の仕事が積まれているのである．

形式化は，ますますその重要性を増し，領域を拡げるであろう．そしてそれは，これが応用された構造の上に，多くのフィードバックを及ぼすであろう．形式化はつぎのような事項に影響を与えるであろう．

形式化された過程から得られる生産物，
形式化された過程を応用している企業や機関，
その仕事の形式的な処理へ進まなければならないような職業，
本質的なコンピュータ応用が，形式化されたシステムの緊密なネットワークを生み出すような国家．

形式化は——強制的に促進されて，あるいは計算機化の副次効果として偶発的に——抽象的エンジニャリングの技法と新しい技術者のタイプを生み出すであろう．それは古典的な機械技術者とは非常に違ったものであるが，しかし必要性と可能性との間，量と経費と遅滞のない納期との間，数学的正確さと非論理的直観

との間，の妥協という共通の特性を基礎としてのるのである．

形式化は社会に影響を与える．この惑星の人口増加のもとでの生活は，進歩したテクノロジーによらなければ維持されえないし，また良いメカニズムが——具体的なものも抽象的なものも——絶対的に必要である．しかしこの人間は機械以上のものであるから，人間のふるまいが，一つの形式化された世界の中に全面的に水没してしまうことから守られなければならない．かりに敏腕な社会技術者によってサービスされれば完全に働くような機械があっても，社会はそれ以上のものである．形式化および機械的展開からは隔離されなければならない世界がある．そして抽象的技術者はある種の職業に対して，人生と仕事について別の原理が追求されるようなエンジニャリングの管理とシミュレーションに落込まないように警告を与えるべきである．

形式化は人間の精神にもフィードバックを及ぼすであろう．それは極度に強力な抽象的メカニズムの形式的方法の進歩によって定まり，またそれは規則性を好み，例外を嫌うように人間を変えてしまうことであろう——ところが，そのような世界は，われわれの世界の人間性とは反対である．われわれの世界では，例外が規則よりも大切なのである．例外を好み，それを理解することは，つねに人間のための最も人間的なタスクとして残るであろう．また，情報処理の形式化におけるわれわれのすべての研究の中で，われわれは決して人間的存在のこの側面を忘れてはならないのである．