Algorism/C Compiler
オートマトンと形式言語

目次

1. 形式言語

オートマトン(automaton,複数形：automata)は、元来一般的に自動機械又は自動人形などの意味に用いられています。しかし、現在の情報処理の分野においてのオートマトンは、ある入力(input)に対し、その処理を自動的に実行し、その結果として適切な応答を出力(output)するようなデジタルシステムを、出来るだけ単純にモデル化したものを意味します。

切符の自動販売機を例に挙げると…入力は逐次投入される硬貨の系列であり、自動販売機は投入された金額の合計を計算･記憶し、その合計が所定の値に達したならば切符(および釣銭)を出力します。自動販売機ようなシステムの基本的構造は、順序機械と呼ばれるオートマトンとして表現されます。

ここで、順序機械としてモデル化出来るシステムの記憶容量は、あらかじめ定められた有限の範囲内と限られるため有限オートマトンと呼ばれます。

更に複雑なシステムをモデル化したオートマトンとしては、プッシュダウンオートマトン(Pushdown Automaton)、線形拘束オートマトン(Linear Bounded Automaton)、チューリング機械(Turing Machine)などがあります。

ここで、今日のいかなる高度な電子計算機によって計算できることも、チューリング機械によって必ず計算できることができます、この意味において、チューリング機械は電子計算機の基本的機能をモデル化していると考えられます。このチューリング機械の考えは、電子計算機の出現以前の1930年代に一般的な計算の手続きを記述する数学的モデルとして、イギリスの数学者Turing, A.M.よって提案されているものです。

なお、オートマトンへの各時点における入力は抽象的に0,1,2…あるいは、a,b,c…等の記号で表し、これを入力記号(input symbol)と呼びます。

個々のオートマトンについては、それが扱い得る入力をあらかじめ定められたある有限種類のものに限定し、それを入力記号の有限集合 $\Sigma=\{0、1\}$ (0と1の2記号からなる集合)のように明示します。刻々と入力される入力記号をつなぎ合わせてできる系列、010、01001等は入力信号列(input string)あるいは、入力系列(input sequence)と呼ばれます。出力についても同様の扱い方をします。

この他に、記憶機構および動作の仕方の規則を決定することにより、特定のオートマトンが定まります。

2. 形式言語

日本語、英語などの自然言語(natural language)およびPascal、Cなどのプログラミング言語(programming language)に対して、それらを抽象化して定められた言語を形式言語(formal language)、または数理言語(mathematical language)と呼びます。通常、オートマトンに対して述べられる言語とは、この形式言語を指します。形式言語を考える場合には、文を構成するための最小単位要素の領域を定め、それを抽象的に記号の集合 Σ で表す必要があります(ここで、Σ は有限集合とします)。

例えば、英語において、文を構成する最小単位を単語と考えた時、各単語を1記号とみなし、使用可能とする全単語をあらかじめ記号で指定しておくことに相当します。

さて、そうすると文(sentence)はそのような記号から構成された記号列とみなすことができます。さらに言語はそのような記号列の内で特定の条件を満足するようなものの集まりになります(特定の条件とは、自然言語の場合、文法上の正しさ、文としての意味等が挙げられます)。

具体的には、$\Sigma=\{a、b、c、d\}$(a、b、c、dの4個の記号から成る集合) 中の記号の重複を許して有限個並べて得られる

ω = abcddad

を Σ 上の記号列 (string over Σ) あるいは Σ 上の系列(sequence over Σ) と言います。

この記号列 ω の長さ(length)は、ω を構成している記号の数 n であると定義し、$|\omega|$ で表します(上記の場合、$|\omega|=7$ になります)。特に、長さが0 ($|\omega|=n=0$) の記号列を空記号列 (empty string) あるいは空系列 (empty sequence) と呼び、ε (epsilon)で表します。

更に、記号列 x、y がそれぞれ次の様な場合…

x = 1234

y = abcd

記法 xy 、yx は、

xy = 1234abcd

yx = abcd1234

と成り、それぞれ記号列の後に別の記号列をつなぎ合わせた結果を取ります。そしてこの記法を x と y の連接(concatenation)と言います。

また、記号列 $\omega=xyz$ (x、y、z はそれぞれ任意の記号列と仮定します) において、x を ω の接頭辞(prefix)、z を接尾辞(suffix)と言います。また、x、y、z はおのおの ω の部分記

号列(substring)と言います。

Σ = { a } であるとき記号列 ω が取りうる全文字列は以下のように表現することが出来ます。

Σ^* = { ε 、a、aa、aaa、aaaa、aaaaa、a.........a、 .... }

と表現することが出来ます。これは空記号列 ε を含めて集合 Σ 上で作り得るあらゆる記号列の全体から成る無限集合を記号 Σ^*として表したもので、スター閉包(star closure)と呼びます。例えば集合 Σ = { a、b、c} のスター閉包は、

Σ^*={ ε 、a、b、c、aa、ab、ac、ba、bb、bc、ca、cb、cc、aaa、aab、aac、aba、abb、abc、aca、acb、.....}

と表現できます。

以上の記法を用いると Σ^*の中からある特定の条件を満たす記号列だけを集めた集合 L、すなわち Σ^*のある特定の部分集合 L が形式言語理論における言語であると言えます。

そして、これを Σ 上の言語(language over Σ)と言います。

Σ 上の言語 L は一般的には無限集合であり、これを有限の記述で Σ^*から規定するための１つの方法として、形式文法と呼ばれる文法が用いられます。つまり、その文法規則に従うと、言語 L 内の文だけが正しい文として生成されるような文法を指定することにより、言語 L が規定されます。

また、別の方法としてオートマトンによって言語 L を規定する方法があります。そのためには入力し尽くした後に L 中の記号列に対してだけ(true、yes 等の)出力をするオートマトンを指定すればいいわけです。

例えば集合 Σ = {0、1、2、3、4、5、6、7、8、9} があり、集合から生成される記号列のうち自然数を言語 L として仮定した時、スター閉包だけでは自然数を表現することは出来ません(00123 等、0 が先頭に来る自然数は無いため)。そして、この集合から自然数を取り出す(生成する)ために形式文法やオートマトンが使用されるわけです。

オートマトンと文法は言語を介して密接な関連を持ちます。また、Σ 上の言語をオートマトン側で考える場合には、Σ は入力記号の有限集合、終端記号の有限集合などのように一般にある指定された 1 個以上の記号からなる有限集合はアルファベット(alphabet)と呼ばれます。そして、アルファベット上の記号列は語(word)と呼ばれることもあります。

形式言語は、その構造に従って、正則(正規)言語、文脈自由言語、文脈依存言語、句構造言語と階層的に分類されます。

3. 基本的な数学的準備

オートマトンでは数学の「集合」についての知識が少し必要なので、簡単にまとめてみます。

- 記述方法
- 演算
- 集合族

集合(set)とは、ある確定された対象物の集まりです。例えば、0と1の集まりも集合です。また、あるオートマトン $\boldsymbol{M}$ に対し、それが入力し尽した後に"出力 1(yes)"の応答を出すような記号列の集まりも集合で、$\boldsymbol{M}$ の受理する言語 $\boldsymbol{L(M)}$ と呼びます。ここで 0、1 などの集合に属す個々の対象物のことを、その集合の要素(element)と言います。1 つの集合には、同じ要素が重複して含まれることはありません。有限個の要素より成る集合を有限集合(finite set)、無限個の要素より成る集合を無限集合(infinite set)といいます。

3-1. 記法

要素 0、1 からなる集合を表記するには、{ 0, 1 } と表します。つまり、要素が 0～9 である場合の集合は、

{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

と表記します。

ただし列挙する順序は任意で大丈夫です。

また、ある条件(性質)を満たす集合 $\boldsymbol{X}$ を表記するには、

{ X ¦ 条件 or 性質 }

と表記します。これは、無限集合であった場合や、条件で表記したほうが集合の意味が理解しやすい場合に用います。

また条件 A と条件 B を同時に満たす集合 $\boldsymbol{X}$ を表記するには、

{ X ¦ 条件 A、条件 B }

と条件を続けて表記します。

例えば、自然数の偶数集合 X を表すには、

{ X ¦ X=2n、 n≧0 }

と、記述できます。

集合 Z の 1 つの要素が z であるとき、「z は Z に属す」あるいは、「Z は z を含む」と言い

$z \in Z$ あるいは $Z \ni z$

と記述します。例えば、9∈{ 1, 3, 5, 7, 9 } です。

集合 $\boldsymbol{A}$、$\boldsymbol{B}$ があり、集合 $\boldsymbol{A}$ の要素は必ず集合 $\boldsymbol{B}$ の要素ともなっている場合は、$\boldsymbol{A}$ は $\boldsymbol{B}$ の部分集合(subset)[図 1 参照]と言い

$A \subseteq B$ あるいは $B \supseteq A$

と記述します(ちなみに、集合 $\boldsymbol{A}$ は自分自身の部分集合でもあるので $A \subseteq A$ と記述できます)。

また、集合 $\boldsymbol{X}$ と集合 $\boldsymbol{Y}$ の関係が $X \subseteq Y$ かつ $Y \supseteq X$ であるとき $X=Y$ となります。

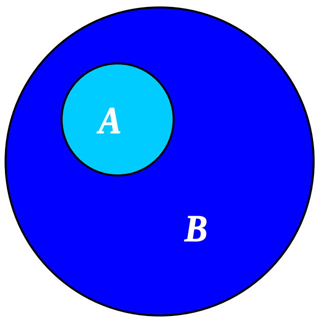

図 1. 部分集合

最後に、何も要素を持たない集合を空集合(empty set)と言い、φ で表記します。

3-2. 演算

集合 $\boldsymbol{A}$、$\boldsymbol{B}$ に対し、これらに属する要素を全て集めた集合を $\boldsymbol{C}$ とすると、集合 $\boldsymbol{C}$ は

C = { c ¦ c∈A、あるいは c∈B }

と表記でき、***C*** は ***A*** と ***B*** の和演算(union) [図 2 参照]と言い

C = A ∪B

とも表記できます。

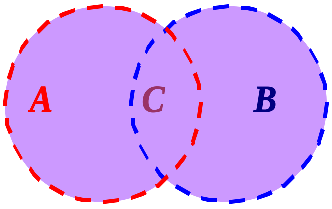

図 2.和集合

集合 ***A***、***B*** に対し、共通に含まれる要素を全て集めた集合を ***C*** とすると、集合 ***C*** は

C = { c ¦ c∈A、かつ c∈B }

と表記でき、C は A と B の共通集合(intersection)、または積集合(product) [図 3 参照]と言い

C = A ∩B

とも表記できます。

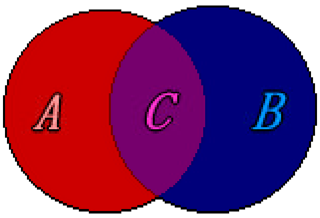

図 3.積集合

3-3. 集合族

集合を、その要素とする"集合"を、集合族(family または class)といいます。

例えば、形式言語の全体は、その要素が言語である集合族を形成します。

特に、集合 ***A*** の全ての部分集合全体を考えたとき、それは集合の"集合"です。

特に、集合 ***A*** の全ての部分集合を考えたとき、それは集合の"集合"ってことになります。つまり、{ B ¦ B ⊆ A } は集合族をなします。この集合を A のべき集合(power set)と呼びます。

簡単な集合論はこれで終了です♪