ライスの定理について（の事前知識）

λ x.x（@lambda_x_x）

2018/9/23(日)

概要

本発表では，ライスの定理について，その概要と応用（いくつかの系）を紹介する．証明の詳細には立ち入らないため，仮定する知識は，理工学部の 1 年で習う程度の離散数理（集合と論理）の知識と，プログラミングに関する簡単な知識と直感，本稿で述べる，決定不能性と指標の考え方のみである．

ライスの定理とは，簡単に言ってしまうと「プログラムに関する決定不能な述語をまとめて示す定理」であり，情報系の学生ならば，必ず学ぶと言っていいほどに有名かつ重要な計算論の定理である．

停止性問題

停止性問題については，今更解説するまでもなく有名になっている．インターネットの数ある文献よりわかり易くここに解説できるかは怪しいが，発表内容にも絡む部分があるので解説をする．

まず，停止性問題を理解する上で，プログラムの符号化について述べる．

プログラムの符号 (自然数) 化

任意の有限長文字列は，以下のような方法で符号化することができる ．

- 文字列に利用可能な文字を $1...m$ までに対応させる e.g. b は 2 番めの文字
- 文字列 $a_1a_2...a_n$ に対応する文字列全体から自然数全体への関数 enc を以下のように構成する
- $enc(a_1a_2...a_n) = p_1^{(a_1\text{の数})+1} + p_2^{(a_2\text{の数})+1} + \cdots + p_n^{(a_n\text{の数})+1}$
  ・ $p_i(1 \leqq i \leqq n)$ は，その文字が i 番目の素数であることを表している e.g $P_2 = 3$
  ・ $a_i(1 \leqq i \leqq n)$ の数とは，その文字が何番目の数かを表している
- 算術の基本定理より，この符号化は文字列全体から自然数全体への単射であり，素因数分解の一意性に基づき，復号して元の文字列に戻すことが可能である

指標

上記の結果から，プログラムを表す文字列も同様に 1 つの自然数として考えることが可能である．

ところで，漠然とプログラムといっても，現代では様々なプログラム言語があり，具体的なイメージが定まるものではない．そこで，今回は以下のような条件を満たすプログラムについて考えることにする．

1. 自然数上のプログラムであり，入力に対して出力結果が毎回同じである．
2. チューリング完全である

1 番目の条件は，そのプログラムが，自然数を入力にとり，自然数を出力に取る関数としてみなせることを表している．ただし，プログラムは**無限ループをする可能性があるため，部分関数とみなせる**というのが正確である．

2 番目のチューリング完全も最近では有名になっている言葉ではあるが，標準的なプログラミング言語と同等の計算力があるということである．

上記の条件を満たすプログラミング言語を符号化したものを (そのプログラムの) **指標**とよぶ

決定不能述語 halt

上に書いたとおり，プログラミング言語は，一般に停止せずに答えが永久に帰ってこないことがありうる．では，あるプログラムの指標を受け取り，そのプログラムがある自然数の入力に対し，停止するか否かを事前に判定するプログラム（コンパイラに近いかもしれない）を作ることができるだろうか？実は，以下の議論から，そのような停止性を判定するプログラムが作成できないことが判明している．

- $halt(x, y)$ という，指標 x が表すプログラムに y を適用させた場合に，停止（True）か非停止（False) を返すプログラムが存在すると仮定して矛盾を導く
- $X(x)$ というプログラムを考え，このプログラムは，$halt(x, x)$ が True の場合には何かしらの方法で無限ループをし，False の場合には，何かしらの値を返す関数だと考える
- $X(x)$ の指標を e とした場合，$halt(e, e)$ の値を考えると，True と仮定しても，False と仮定しても矛盾する（ともに True かつ False ということになってしまう）
  よってその様なプログラム $halt$ は作成することができない．

このように，述語に相当するプログラム（メンバーシップを判定するプログラム）が作成できない問題を，**決定不能問題**という．