加速定理与函数分层

王永革
(南开数学研究所)

Speed-up Theorem and Hierarchy of the Recursive Functions

Wang Yongge
(Nankai Institute of Mathematics Nankai University)

Abstract

In this paper, we'll introduce a kind of O(F) -LOOP operator, which will lead to a hierarchy of any recursive functions: O(F)= $O(F)_n$. We've proved that this hierarchy corresponds to the speed-up theorem, i.e. given any r(x) $\in O(F)_i$, there is a language having $O(F)_{i+1}$ complexity which can be speeded up by r(x), and also there is a r(x) in $O(F)_i$, so that any language having $O(F)_i$ complexity cannot be speeded up by r(x). Through this, we can grasp the soul of Speed -up theorem in a more higher level.

摘要

本文引进一种 O(F)-LOOP 算子，通过该算子可对一般递归函数集进行分层，且该算子对应于计算复杂性中的加速定理。由此我们得到加速定理的定量描述。

§1 引 言

Grzegorczyk 在文 [6] 中对原始递归函数进行分层, 这种分层等价于通过递归算子的使用次数而对原始递归函数进行的分层，但这种分层仅仅是对应于原始递归函数的。那么原始递归函数以外呢？本文将引进一种 O(F)-LOOP 算子，由它出发 可对任意的递归函数集进行分层，同时我们证明了 Grzegorczyk 的分层与本文的分层具有非常良好的性质: 它们对应于加速定理。

§2 程序语言 O(F)-LOOP

以下各节的研究基于一种程序语言 O(F)-LOOP，为此我们先来介绍这种语言。本语言中,用一些字符表示 变量. 特别地，将

$$X_1,\ X_2,\ \cdots \text{ 和 } Y$$

分别称为输入变量和输出变量，将

$$Z_1,\ Z_2,\ \cdots$$

称为局部变量。

O(F)-LOOP 的指令共有以下六种：

1. $V \leftarrow 0$ (零指令)
2. $V \leftarrow V+1$ (加 1 指令)
3. $V \leftarrow V'$ (赋值指令)
4. LOOP V (循环指令)
5. END (循环返回指令)
6. $V \leftarrow O(f(v_1,\cdots,V_n))$ ($f \in F$, 本指令称为 Oracle 指令)

定义2.1 设 F 为一给定函数集，则 O(F)-LOOP 程序 P 可计算函数 $g(x_1,\cdots,x_n)$是指对于任意输入 $X_1=x_1$, $X_2=x_2$, …, $X_n=x_n$ 程序 P 的输出 Y 为 $g(x_1,\cdots,x_n)$

例如：对于下述程序 P

$Z_1 \leftarrow X_1+1$
$Z_2 \leftarrow Z_2+1$
LOOP Z_2
  $Z_2 \leftarrow Z_2+1$
END
$Z_3 \leftarrow O(f(Z_2))$ ($f \in F$)
$Y \leftarrow Z_3$

设输入 $x_1=1$, $x_2=2$。程序执行到第三条指令 LOOP Z_2 时，$Z_2=3$，故由第三到第五条指令组成的循环体重复执行三次。注意：循环体内第四条指令使 Z_2 值发生变化，但这不再影响本循环重复执行的次数。程序执行完毕后输出 y=f(6)。事实上，本程序所能计算的函数为 y=f(2x_1+4)。

定义2.2 容许循环指令最多嵌套 n 层的所有 O(F)-LOOP 程序组成的集合记为 $O(F)\text{-LOOP}_n$。

显然，$\cup O(F)\text{-LOOP\_n}$ 就是所有 O(F)-LOOP 程序的集合。

§ 3 E-Ackermann 函数

为了方便起见，以下各节都假定： $F=\{f_1, \cdots, f_m\}$ 是满足下述条件的有穷函数集：

(1). 对于 $1\leqslant i\leqslant m$，$f_i(x)$ 是全定义的；

(2). $f_1(x)=x+2$, $f_2(x,y)=\max\{x,y\}$；

(3). $3\leqslant i\leqslant m$ 时，$f_i(x_1,\cdots,x_n)$ 对于每一分量都是单调不减的。

定义3.1 定义以 F 为基始的 E-Ackermann 函数 $A_F(m,n)$ 为：

$$A_F(0,0)=1, \quad A_F(0,1)=2,$$
$$A_F(0,n)=\max\{f_1(n),f_2(n,n),\cdots,f_m(n,\cdots,n)\} \quad (n\geqslant 2)$$
$$A_F(m+1,n+1)=A_F(m,A_F(m+1,n))$$

并称如下定义的函数序列 {u_n(x)} 为 A_F(m,n) 导出的层次函数：

$$u_n(x)=A_F(n,x)$$

易知

$$u_0(0)=1, \quad u\_0(1)=2,$$
$$u_0(x)=\max\{x+2,x,f_3(x,\cdots,x),\cdots,f_m(x,\cdots,x)\} \quad (x\geqslant 2)$$
$$u_{n+1}(x)=u_n^{(x)}(1)$$

这里 u_nˆ{(x)}=u_n⋯u_n(1)

下边我们给出 $\{u_n(x)\}$ 的一些基本性质：

性质3.1 $u_{n+1}(x+1)=u_n(u_{n+1}(x))$

性质3.2 $u_n^{(k)}(x)\geqslant k$

性质3.3 $u_n(x)>x$

性质3.4 $u_n(x+1)>u_n(x)$

性质3.5 $u_{n+1}(x)\geqslant u_n(x)$

性质3.6 $u_n^{(k+1)}(x)>u_n^{(k)}(x)$

性质3.7 $u_n^{(k+1)}(x)\geqslant 2*u_n^{(k)}(x) \quad (n\geqslant 1)$

性质3.8 $u_n^{(k+1)}\geqslant u_n^{(k)}(x)+x$

性质3.9 $u_n^{(k)}(x)\geqslant 2^k*x$

以上各性质的证明完全同于 [1] 中 Ackermann 函数类似性质的证明，这里从略。

性质3.10 对于是一切 n, k, $u_{n+1}(x)>u_n^{(k)}(x)$ a.e.

证明 n=0 时，因为对于 x>1, $u_0(x)>x+2$, $u_1(m)=u_0^{(m)}(1)>2m-1$, 所以对于任意 k，当 $x>2k+1$时,$u_0^{(x-k)}(1)>2(x-k)-1=2x-2k-1>x$, 即 $x>2k+1$ 时 $u\_1(x)=u_0^{(k)}(u_0^{(x-k)}(1))>u\_0\hat{}\{(k)\}(x)$.

$n\neq 0$ 时，施归纳于 k，k=0 则 $u_{n+1}(x)>x=u_n^{(0)}(x)$

设结论对 k 成立，则

$$u_n^{(k+1)}(x)<u_n^{(k+1)}(2x-4) \quad \text{a.e}$$
$$\leqslant u_n^{(k+1)}(u_1(x-2))\leqslant u_n^{(k+1)})(u_n(x-2))=u_n^{(2)}(u_n^{(k)}(x-2))$$
$$\leqslant u_n^{(2)}(u_{n+1}(x-2)) \quad \text{(a.e.)}$$
$$=u_n^{(2)}(u_n^{(x-2)}(1))=u_{n+1}(x)$$

性质3.11 当 $n>2$ 时，$u_n(x)>x^2$ a.e.

证明 $u_1(x)\geqslant 2x$ a.e.,

$u_2(x)\geqslant 2^x>x^2$ a.e.,

$u_n(x)>u_2(x)>x^2$ a.e. (n>2).

§ 4 O(F)-LOOP 函数

定义4.1 设 F 为上节定义的函数集，则将 0(F)-LOOP 程序所能计算的函数全体称为 0(F)-LOOP 函数集，记为 0(F)。将 0(F)-LOOP$_n$ 程序所能计算的函数全体记为 0(F)$_n$.

显然，有 0(F)=∪0(F)$_n$

对于 f(x)∈0(F)，我们称 f(x) 可由 F 出发经 0(F)-LOOP 算子运算而得。

定义4.2 设 P 是一 0(F)-LOOP 程序，则我们以如下方式定义P的时间、空间和时空复杂度。时间复杂度 $T_P(x_1,\cdots,x_n)$ 指对于输入 $(x_1,\cdots,x_n)$，程序P在执行过程中所执行的零指令、加1指令、赋值指令和Oracle 指令数的总和。空间复杂度 $S_P(x_1,\cdots,x_n)$ 指对于输入 $(x_1,\cdots,x_n)$，程序P在执行过程中各个输入变量、中间变量和输出变量所曾达到的最大值。时空复杂度定义为：$ST_P(x_1,\cdots,x_n)=\max\{S_P(x_1,\cdots,x_n), T_P(x_1,\cdots x_n)\}$

定理4.1 设 P∈0(F)-LOOP$_n$，则 $ST\_P(x_1,\cdots,x_n)\in 0(F)_n$.

证明 在程序 P 前边添加两条指令：S←0, T←0, 其中 T、S 为 P 中未出现过的二中间变量。在 P 的每一条零指令、加1指令和 Oracle 指令后边添加两条指令 T←T+1, S←0(max(S,V))，其中 V 为前边一条指令中值发生变化的那个变量。在P的最后加指令 Y ←0(max(T,S))。从而得到一新程序 P’ 。显然， P’ ∈ 0(F)-LOOP$_n$，且 P’能够计算 $ST_P(x_1,\cdots,x_n)$. 故 $ST_P\in 0(F)_n$.

定理4.2 设 P∈0(F)-LOOP$_n$，则存在 k，使得

$$ST_P(x_1,\cdots,x_n)\leqslant u_n^{(k)}(\max(x_1,\cdots,x_m))$$

证明 以下记 $v=\max(x_1,\cdots,x_m)$，对 n 施归纳法. n=0 时，P 中无 LOOP-END 指令，不仿设 P 共有 k 条指令。则有

$$ST_P(x_1,\cdots,x_m)\leqslant u_0^{(k)}(v)$$

设对于 n-1 结论已成立。则

(1). 当 P∈0(F)-LOOP$_{n-1}$ 时，由归纳假定，存在 k，使得 $ST_P(x_1,\cdots,x_m)\leqslant u_{n-1}^{(k)}(v)\leqslant u_n^{(k)}(v)$ 成立。

(2). 当 P 具有如下形式

LOOP V
Q
END

时，其中 Q∈0(F)-LOOP$_{n-1}$。由归纳假定，存在 k 使得

$$S_Q(X_1,\cdots,x_m)\leqslant ST_Q(x_1,\cdots,x_m)\leqslant u_{n-1}^{(k)}(v)$$
$$T_Q(X_1,\cdots,x_m)\leqslant ST_Q(x_1,\cdots,x_m)\leqslant u_{n-1}^{(k)}(v)$$

故，当循环体Q执行一次后，各变量的最大可能值为 $u_{n-1}^{(k)}(v)$；当循环体Q执行两次后，各变量的最大可能值为 $u_{n-1}^{(2k)}(v)$； 依次类推可得：$S_P(x_1,\cdots,x_m)\leqslant u_{n-1}^{(vk)}(v)\leqslant u_{n-1}^{(vk+1)}(v)\leqslant u_{n-1}^{(v(k+1))}(1)\leqslant u_{n-1}^{(v^{\ })}(1)\leqslant u_{n-1}^{(\ \ )}(1)\leqslant u_{n-1}^{(\ \ )}(1)=u_n(u_n^{(k+1)}(v))=u_n^{(k+2)}(v)$.

反复利用性质3.8，可得：$T_P(x_1,\cdots,x_m)\leqslant u_{n-1}^{(k)}(v)+u_{n-1}^{(2k)}(v)+\cdots u_{n-1}^{(vk)}(v)\leqslant u_{n-1}^{(k+1+k)}(v)+u_{n-1}^{(3k)}(v)+\cdots u_{n-1}^{(vk)}(v)\leqslant\cdots\leqslant u_{n-1}^{(vk+1)}(v)\leqslant\cdots\leqslant u_n^{(k+2)}(v)$.

所以 $ST_P(x_1,\cdots,x_m)\leqslant u_n^{(k+2)}(v)$

(3). 当 P 具有如下形式

Q_0
LOOP V_1
P_1
END
Q_1
…
LOOP V_r
P_r
END
Q_r

时，其中， P_i, $Q_i\in 0(F)$-LOOP$_{n-1}$.

反复利用性质3.8，可得 $ST_P(x_1,\cdots,x_m)\leqslant u_n^{(\ )}(v)+u_n^{(\ )}(u_n^{(\ )}(v)+\cdots+u_n^{(\ )}(u_n^{(\ )}(\cdots u_n^{(\ )}(v))\cdots)\leqslant\cdots\leqslant u_n^{(+++s-1)}(v)$，至此归纳步骤完成。

推论　设 $g\in O(F)_n$，则有常数 k，使得 $g(x_1,\cdots,x_m)\leqslant u_n^{(k)}(\max(x_1,\cdots,x_m))$

定理4.3　$u_{n+1}(x)\in O(F)_{n+1}\backslash O(F)_n$，$u_0(x)\in O(F)_0$.

定理4.4 设 P 是计算 $g(x_1,\cdots,x_m)$ 的 O(F)-LOOP 程序，如对于 $n\geqslant 2$，存在常数 k，使得 $ST\_P(x_1,\cdots,x_m)\leqslant u_n^{(k)}(\max(x_1,\cdots,x_m))$，则有 $g\in O(F)_n$.

以上二定理证明完全类似于[1]中 Ackermann 函数及 LOOP 程序相应性质的证明，在此从略。

§ 5　加速分层与加速定理

定义5.1 离线图灵机 M 是具有一条带边界标志的只读输入带和 k 条单向无穷工作带的图灵机。对于每个长为 n 的输入，如果M在每条工作带上最多扫描 S(n) 个带单元，则称M是 S(n) 空间有界的，或者说 M 具有空间复杂度 S(n)。被M接收的语言L称为具有空间复杂度 S(n).

定义5.2 函数 S(n) 空间可构造，如果存在 S(n) 空间有界的离线图灵机M，使得对每一个 n，都有长为 n 的输入α，M对α的动作恰好用到 S(n) 个工作带单元。特别地，如果对一切长为 n 的输入M都恰好用到 S(n)　个工作带单元，则称 S(n) 是完全空间可构造的。

性质5.1 对于任一全定义的递归函数 f(x)，总存在一函数 $g(x)\geqslant f(x)$，使得 g(x) 是完全空间可构造的。

性质5.2 log n，n^k，2^n，和 n! 都是空间可构造的。

性质5.3 如果 $S_1(n)$，$S_2(n)$ 是空间可构造的，则 $S_1(n)*S_2(n)$，$2^{S(n)}$，S_1(n)^{S(n)} 与 $f(n)=S_1^{(n)}(1)=S_1\cdots S_1(1)$ 都是空间可构造的。

性质5.4 如果 $S(n)\geqslant n$ 是空间可构造的，则 S(n) 也是完全空间可构造的。

以上性质证明参阅[2]。

定义5.3 称函数集序列　，　，… 为函数集　的一个加速分层，如果以下条件成立：

(1).

(2).

(3). 存在完全空间可构造的函数 $u_0(x)$，使得对于任意 k，$u_0^{(k)}(x)\in$　，$u_n^{(k)(x)}\in$　，其中 $u_n(x)=u_{n-1}^{(x)}(1)$. 并称 $\{u_n(x)\}$ 为该加速分层的层次函数。

(4). 任给 f(x)　　，存在 k，使得 $f(x)\leqslant u_n^{(k)}(x)$.

例5.1 原始递归函数集的 Grzegorczyk 分层是原始递归函数的一个加速分层。其中层次函数就是由 Ackermann 函数导出的层次函数。

例5.2 如果取适当的函数集 F 使得 $u_0(x)=\max\{f_1(x),f_2(x,x),\cdots,f_n(x,\cdots,x)\}$是完全空间可构造的。则 $\{O(F)_n:\ n=0,\cdots\}$ 是 O(F) 的一个加速分层。

定义5.4　设 r(x)　是一全定义的递归函数，L　为$\Sigma=\{0,1\}$　上的语言，则我们称L可被 r(x)　加速，如果，任给接受L的图灵机M_i，都存在接受L的另一图灵机 M_j，使得 $r(S_j(x))\leqslant S_i(x)$,其中$S_i(x)$，$S_j(x)$ 分别为 M_i，M_j 的空间复杂度。

定义5.5 称语言 L 具有　复杂度，如果存在接受L的图灵机 M_i 及 $f(x)\in$　,使得 $S_i(x)\leqslant f(x)$.

引理5.1 设 $\{u_n(x)\}$ 是函数集　的某个加速分层的层次函数，对于任一给定 m，如果 $u_m(x)\geqslant x^2$ a.e.，则存在语言 L 满足下述条件

(1). 如果 M_i 接受 L，则 $S_i(x)>u_{m+1}(x-i)$　a.e.

(2). 任给 k，存在接受 L 的 M_j，使得对于一切 $x\geqslant k$，$S_j\leqslant u_{m+1}(x-k)$.

证明完全类似于 [2] 中加速定理的证明，这里从略。

定理5.1 设　=　为　的一个加速分层，$\{u_n(x)\}$为其层次函数，$u_m(x)\geqslant x^2$　a.e.，则

1. 存在 r(x)∈　，使得任给具有　复杂度的非平凡语言 L(即最少具有线性复杂度)，L不能被 r(x) 加速。
2. 任给递增函数 r(x)∈　，存在具有　复杂度的语言 L，L 可被r(x)加速。

在证明本定理之前，我们先对本定理作几句说明：定理中 (1) 说明加速定理并非任意加速，对于任一给定的加速因子 r(x)，能被 r(x) 加速的语言至少要具有比 r(x) 高一级的复杂度，即在　中。(2) 说明了加速现象的存在性。

(1)、(2) 综合起来说明对于 r(x)∈　，存在具有　复杂度的语言 L 可被r(x) 加速，但事实上这种加速并未能“真正”地对L加速，因为被加速后的L的复杂度仍在　中，而未降到　中。直观上讲，就是说对于给定的加速因子r(x)，存在一个复杂度非常大的 S(x)，使得 r(S(x)) 与 S(x) 具有同一数量级的复杂度。

因此加速定理似乎没有多少“实际”意义，因为取 r(x) 为多项式时，可被 r(x) 加速的语言最少具有指数复杂度(由 Grzegorczyk 分层)，且存在一个指数复杂度的语言可被该 r(x) 加速，而加速后的语言仍具有指数复杂度。

定理5.1的证明：(1) 取 $r(x)=u_m(x)$，则任给具有 复杂度的语言 L，存在 $r(x)\in$ 及图灵机 M_i，M_i 接受 L 且 $S_i(x)\leqslant r_m(x)$. 由于 $r_m(x)\in$ ，故存在 k，使得 $r_m(x)\leqslant u_m^{(k)}(x)$. 假设 L 可被 r(x) 加速，则有接受 L 的另一图灵机M_j，使得$r^{(k+1)}(S_j(x))\leqslant S_i(x)\leqslant u_m^{(k)}(x)$. 而 $S_j(x)\geqslant x$，所以$u_m^{(k+1)}(x)\leqslant u_m^{(k)}(x)$，这与性质 3.6 矛盾。故 L 不能被 r(x) 加速。

(2). 由引理5.1，存在语言 L，满足

(a). 如果 M_i 接受 L，则 $S_i(x)>u_{m+1}(x-i)$ a.e.

(b). 任给 k，存在接受 L 的图灵机 M_j，使得 $S_j(x)\leqslant u_{m+1}(x-k)$

又因 $r(x)\in$ ，故存在 k_0，使得 $r(x)\leqslant u_m\hat{}\{(\ )\}(x)$.

现在对任给的接受 L 的图灵机 M_i，由(a)可知 $S_i(x)>u_{m+1}(x-i)$ a.e.，取$k=i+k_0+1$，则由(b)可知，存在接受 L 的图灵机 M_j，使得 $S_j(x)\leqslant u_{m+1}(x-k)$,所以 $r(S_j(x))\leqslant u_m^{(\ )}(u_{m+1}(x-k))=u_{m+1}(x-k+k\_0)<u_{m+1}(x-i)<S_i(x)$.

定理5.2（内部位移加速定理）设 $=\cup$ 为 的一加速分层。则对任意的 m 和 k，存在具有复杂度的语言 L，使得任给接受 L 的图灵机 M_i，都存在满足下述条件的图灵机 M_j

(1). M_j 接受 L;

(2). $S_j(n)\leqslant S_i(n-c)$ a.e.

证明 由引理5.1，对于 m，存在具有 复杂度的语言 L，使得

(a). 任给 M_i 接受 L，都有 $S_i(n)>u_{m+1}(n-i)$ a.e.

(b). 对于 k=i+c，存在 M_j 接受 L，且 $S_j(n)<u_{m+1}(n-i-c)$所以 $S_i(n-c)>u_{m+1}(n-i-c)>S_j(n)$ a.e.

在结束本文之前，作者对徐书润老师精心的指导表示衷心的感谢，同时对编辑及本文审稿人提出的建设性修改意见表示衷心感谢!

参考文献

[1] M.D.Davis, E.J.Weyuker, Computability, Complexity, and Languages, Academic Press. 1983.
[2] J.E.Hopcroft, J.D.Ullman, Introduction to Automata Theory, Languages, and computation, Addison-wesley Publishing Company. 1979.
[3] Cutland, Computability, Cambridge University Press, 1980.
[4] E.Borger, computability, Complexity, Logic, North-Horland Press. 1989.
[5] H.E.Rose, Subrecursion: Functions and Hierarchy, Oxford University Press, 1984.
[6] A.Grzegorczyk, Some classes of recursive functions, Rozprawy matematycene no.4, Instytut Matematyczny Polskiej Akad. Nauk. Warsaw, 1953