TCS **作业一**：Turing Machine and Halting

计试 2101 仲星焱

2023 年 7 月 4 日

1.1 一个比较显然的想法是以一个 $*$ 隔开输入和输出的部分，在输出部分实现二进制的 +1 功能。以下假设图灵机读写头初始位于输入串任意位置，输出在输入左侧空一格的位置。

算法具体流程为，每次找到输入串当前存活的最右字符，删去之后对输出进行 +1，具体可以参考下表：

表 1: Problem 1.1 图灵机构造

含义	状态	读到	S	W	M
初始状态；扫描输入，向右直到遇到结尾空位置	q0	1/0	q0	1/0	+1
		*	q1	*	-1
处理输入，读到 * 位置并刚返回，置当前位为 *	q1	1/0	q2	*	-1
		*	end	/	/
扫描输入，前往计数	q2	1/0	q2	1/0	-1
		*	q3	*	-1
执行计数，找连续进位到达的最高位	q3	1	q3	0	-1
		0/*	q4	1	+1
计数器更新完毕，前往扫描输入	q4	1	illegal	/	/
		0	q4	0	+1
		*	q0	*	+1
结束状态，直接停机	end	/	/	/	/
非法结束状态，磁带状态异常，直接停机	illegal	/	/	/	/

1.2　由于没有详细说明图灵机读写移动和状态转移的顺序，本例中假设一般图灵机的执行顺序为：读状态、写、按照读取的数据进行移动、按照读取的数据进行状态转移。

此解法下，转化后的二进制图灵机状态数是等价三进制图灵机的 15 倍。

输入转化方式即为题设方式，二进制图灵机两位解码三进制图灵机的一位，具体为 $0 \to 00, 1 \to 01, 2 \to 10, * \to **$，接下来用 $hi(sign)$ 表示这些编码的高位，用 $lo(sign)$ 表示编码的低位。

对于三进制图灵机 $T = (\Gamma_T, S_T, W_T, M_T)$，构造二进制图灵机 $S = (\Gamma_S, S_S, W_S, M_S)$，其中 $\Gamma_S = \Gamma_T \times \{\#, r_0, r_1, r_*, 0++, 1++, *++, 0--, 1--, *--, +, -, 0, 1, *\}$，共 15 个附加符号，接下来解释这些符号的含义。

$\#$ 表示当前位于某个编码的高位，读取高位之后前进到低位，设当前状态为 $(x, \#)$，有 $S_S(x, \#, cur) = (x, r_{cur}), W_S(x, \#, cur) = cur, M_S(x, \#, cur) = +1$。其中 cur 表示当前磁头读取到的数据，即 $0, 1$ 或 $*$。

r_0, r_1, r_* 表示刚才读取了某个编码的高位，目前位于马上读取的低位，之后回到高位进行写，低位的写移动和状态转移操作被压缩在剩余的附加符号中，用 now 表示 r 附带的数据，$W_S(x, r_{now}, cur) = lo(W_T(x, < now, cur >)), M_S(x, r_{now}, cur) = -1$，$S_S(x, r_{now}, cur) = (S_T(x, < now, cur >), extra), extra$ 具体的取值根据 W_T, M_T 来确定，详细描述为 $extra$ 开头符号为 $hi(W_T(x, < now, cur))$，如果 $M_T(x, < now, cur >) \neq 0$, 则携带两个对应的符号，表示两次位移。

$0++, 1++, *++, 0--, 1--, *--$ 表示当前位于某位的高位，此时磁针探头需要写的数据为符号中数据，在此之后需要向左/右移动进行下一次操作，开始下一次操作时，图灵机 T 对应的状态即为此时 S 携带的状态，$S_S(x, now++, cur) = S_S(x, +), W_S(x, now++, cur) = now, M_S(x, now++, cur) = +1$, 此处 $--$ 同理。

$+, -$，表示开始下一次操作之前还需要向指定方向再进行一次移动。$S_S(x, +, cur) = (x, \#), W_S(x, +, cur) = cur, M_S(x, +, cur) = +1$, 此处减号同理。

$0, 1, *$ 表示填完该位之后直接以下个状态运行，$S_S(x, now, cur) = (x, \#)$，$W_S(x, now, cur) = now, M_S(x, now, cur) = 0$。

综上，即为三进制-二进制图灵机转化策略，由此可以推广说明任意字

符集的图灵机计算能力等价。

1.3　首先，利用数轴上的整数对 two-way tape 的所有单元进行编号，即编号为 $\ldots, -3, -2, -1, 0, 1, 2, 3\ldots$，将 one-way tape 每两个单元与 two-way tape 每一个单元进行一一对应，从左到右编号为 $0, 1, -1, 2, -2, 3, -3, \ldots$。

对于 two-way tape 的输入，映射到对应的 one-way tape 的第二单元中，另外的第一单元做如下处理，0 的第一单元置 0，正数的第一单元置 1，负数的第一单元置 $*$。

于是每次需要对磁头进行移动时，只需要向前移动一单元即可知道实际的移动方向，只需要对 S 中的有限个状态构造 $\Gamma_T = \Gamma_S \times \{-1, 0, +1, \} \times \{IO, MV\}$，其中 IO, MV 表示当前状态应该处理磁头为输入/输出信息还是移动信息，前面的数字表示当前需要执行移动的方向，同样为有限个状态。最后，按照上述描述构造对应的移动和输入输出即可。

反过来，对于任意 one way 图灵机，显然可以直接用 two-way 图灵机进行模拟。

综上，对于 one-way 图灵机 T，总是存在一个 two-way 图灵机 S 与其等价。

1.4　由题意，考虑证明必要性，若 L 是递归可枚举的，根据定义，$\exists T, \forall x \in L, T$ 停机并且输出 1，否则停机情况下输出 0，采用通用图灵机 U 对 T 进行模拟，若 T 停机，U 读取其输出，若输出为 1，U 停机，否则 U 自动进入无穷循环即可。

考虑证明充分性，若 $\exists T, \forall x \in L, T$ 停机 $, \forall x \notin L, T$ 不停机，则采用通用图灵机 U 对 T 进行模拟，T 停机时 U 输出 1 即可，根据定义，L 是递归可枚举的。

综上，原命题成立，L 是递归可枚举的当且仅当存在图灵机 T 当且仅当 $x \in L$ 时停机。

1.5　命题不成立，证明如下：

假设命题成立。构造平凡图灵机 T，其接受输入 (t, x) 并模拟图灵机 t 在输入 x 上的运行，显然 T 停机当且仅当 t 在 x 上停机。

由命题成立知道 $\exists T'$，在所有输入 (t,x) 上停机当且仅当 t 在 x 上不停机。

采用通用图灵机 U 对 T 和 T' 进行同步模拟，则我们得到一个停机问题的判定算法，与已知矛盾。

故命题不成立。

1.6　该语言不递归，证明如下：

假设该语言递归，则 $\exists S, f_S(T,x,y) = [f_T(x) = f_T(y)]$，其中 [] 表示对其中命题判断真假，真命题则返回 1，假命题则返回 0，根据定义，上述 S 对于任意输入必停机，且 (T,x,y) 作为输入的时候，输出为 $f_S(T,x,y)$ 的值。

现在，利用一个通用图灵机 U, 然后找到一个平凡图灵机 T ，使得 T 直接停机并输出 1，使用 U 模拟 T, 并且 U 停机并输出 1 当且仅当 T 停机并输出 1。有 $f_U(T,x) = 1$。

现在，对于任意图灵机 T' 和任意输入 x'，有 $f_S(U,(T,x),(T',x')) = [f_U(T,x) = f_U(T',x')] = f_U(T',x')$，即可利用 S 计算 T' 在 x' 上是否停机，与停机问题结论互斥，故假设不成立。

综上，该语言不递归。

1.7　(a) 是，考虑如下证明:

设该语言为 L_1，使用通用图灵机 U 对 T,S 进行输入为 x 的同步模拟，在 T,S 出现停机时 U 输出结果并停机，对于 $\forall(T,S,x) \in L_1$，T 在 x 上必然停机，故 U 在 (T,S,x) 上必然停机。另取一个通用图灵机 D 模拟 U 即可证明语言 L_1 是递归可枚举的。

(b) 不是，考虑反证：

设该语言为 L_2，由于其是递归可枚举的，$\exists B$，使得 $\forall(T,S,x) \in L_2$，B 都停机并做出输出。另外设 (a) 中所得图灵机为 A。

构造平凡图灵机 S，使得 S 对任意输入都不停机。对于任意图灵机 T 和输入 x，要么 T 比 S 快，要么 S 快于或等于 T, 两种情况分别对应 T 在 x 上停机或不停机。

对于停机的情况，在 A 上运行 $(T,S,x) \in L_1$ 必然让 A 停机。

对于不停机的情况，在 B 上运行 $(S,T,x) \in L_2$ 必然让 B 停机。

采用通用图灵机 U ，对 A 运行输入为 (T,S,x) 和对 B 运行输入为 (S,T,x) 进行同步模拟，直到 A 和 B 其中一个停机（由前述可知两者必然有一个停机），停机的一方即暗示了 T 在 x 上是否停机，由此我们找到一个停机问题的判定算法，与已知停机问题无图灵判定算法矛盾，故假设不成立。

综上，该语言不是递归可枚举的。