The unary loop-computable functions are recursively enumerable.

(1) The class of all LOOP programs is r.e.

Proof: on the slides → one can find a grammar (context-free) which generates the class of all LOOP programs.

(2) PROBLEM:

- Not every LOOP Program computes a function.
- From the form of a LOOP Program one cannot see if the program computes a function or not.

(and one cannot see if the program computes a unary function: we would need a way of checking whether for every value of register x_1, at the end the values stored in registers $x_3, \dots x_e$ are 0 or not)

(3) IDEA

Change every LOOP program P to obtain a LOOP program $\hat{P}$ which computes a function $f_{\hat{P}}: \mathbb{N} \to \mathbb{N}$.

such that if P computes a function $f_P: \mathbb{N} \to \mathbb{N}$ then $\hat{P}$ computes the same function (i.e $f_P = f_{\hat{P}}$)

Construction: P LOOP program using registers $\{x_1, \dots x_e\}$.
Let x_n be a new register name.

$\hat{P} :=$

```
xn := x1;          % copy value in x1 in xn
P;                 % execute P
x1 := xn; xn := 0; % restore contents of x1
xe := 0; xe-1 := 0; ... ; x3 := 0   % set all registers
                                    % x3...xe, xn to 0
```

(4) Show that the class of unary loop computable functions is r.e.

The i-th LOOP computable function (unary) $f_i: \mathbb{N} \to \mathbb{N}$ is

$$f_i := \begin{cases} f_{\hat{P}} & \text{if } i \text{ is the Gödel number of the LOOP Program } P_i \\ g_0 & \text{otherwise} \end{cases}$$

where $g_0: \mathbb{N} \to \mathbb{N}$ is defined by $g_0(n) = 0$ for all $n \in \mathbb{N}$.

A Turing Machine M_{ra} can recursively enumerate $\{f_1, f_2, \dots\}$ (resp. to LOOP programs)

$i \mapsto$ i encodes correct LOOP Prog? P ? yes → $\hat{P}$. no → P_{NOP}.
$i \in \mathbb{N}$