Universality

Then we append the following instruction:

\[ [C] \text{ IF } K = \text{Lt} (Z) + 1 \lor K = 0 \text{ GOTO F} \]

So if the computation has ended, GOTO F, where the proper value will be output. 
Otherwise, the current instruction is decoded and executed:

\[ U \leftarrow \tau((Z)_K) \]
\[ P \leftarrow p_{r(U)+1} \]

Remember that \((Z)_K = (a, b, c)\) is the number of the K-th instruction.
So \(U = (b, c)\) is the code of the statement to be executed.
The variable mentioned in this statement is the \((c + 1)\)-th, i.e., 
the \(r(U) + 1\)-th.

Universality

Then \(l(U)\) contains the type of the instruction to be executed.

\[ \text{IF } l(U) = 0 \text{ GOTO N} \]
\[ \text{IF } l(U) = 1 \text{ GOTO A} \]
\[ \text{IF } \neg (P | S) \text{ GOTO N} \]
\[ \text{IF } l(U) = 2 \text{ GOTO M} \]

If either the instruction is \(V \leftarrow V\), or the instruction is \(V \leftarrow V - 1\) and \(V = 0\) (as indicated by the absence of \(P\) in \(S\)), or the instruction is \(\text{IF } V \neq 0 \text{ GOTO L}\) and \(V = 0\), then nothing is done to \(S\) ("nothing" - GOTO N).
If the instruction is \(V \leftarrow V + 1\), then the exponent of \(P\) in \(S\) needs to be incremented ("add" – GOTO A).
If the instruction is \(V \leftarrow V - 1\) with \(V > 0\), then the exponent of \(P\) in \(S\) needs to be decremented ("minus" – GOTO M).

Universality

If none of the four previous predicates were true, a GOTO command has to be executed:

\[ K \leftarrow \min_{l \leq \text{Lt}(Z)} [l((Z)_k) + 2 = l(U)] \]
\[ \text{GOTO C} \]

So if the label \(l(U) – 2\) exists in the program, the number \(K\) of the next instruction to be executed will be set to the first instruction with that label. 
Otherwise, \(K\) will be set to 0.

As you remember, if \(K = 0\) or \(K = \text{Lt}(Z) + 1\), then the computation stops.
In any case, our interpreter program executes a GOTO C to execute the next instruction.

Universality

The program continues as follows:

\[ [M] \quad S \leftarrow \lfloor S / P \rfloor \quad \text{GOTO N} \]
\[ [A] \quad S \leftarrow S \cdot P \quad \text{GOTO N} \]
\[ [N] \quad K \leftarrow K + 1 \quad \text{GOTO C} \]

The value of the variable in the current instruction is decremented or incremented by 1 by dividing or multiplying \(S\) by \(P\), respectively.
Then \(K\) is incremented and the next instruction executed.

Universality

The program concludes with the following line:

\[ [F] \quad Y \leftarrow \lfloor S \rfloor \]

This way, after termination of the interpreted program, its output value becomes the output value of the interpreter.

On the next slide, we will list the entire interpreter program.
Universality

For each $n > 0$, the sequence

$$\Phi^n(0), \Phi^n(1), \ldots$$

enumerates all partially computable functions of $n$ variables. We can also write:

$$\Phi^y(n)(x_1, \ldots, x_n) = \Phi^n(x_1, \ldots, x_n, y).$$

We can omit the superscript $(n)$ when $n = 1$:

$$\Phi^y(x) = \Phi(x, y) = \Phi^1(x, y).$$

Universality

Consider the following predicates:

$$\text{STP}^y(x_1, \ldots, x_n, y, t)$$

$\Leftrightarrow$ Program number $y$ halts after $t$ or fewer steps on inputs $x_1, \ldots, x_n$

$\Leftrightarrow$ There is a computation of program $y$ of length $\leq t + 1$, beginning with inputs $x_1, \ldots, x_n$

These predicates are computable, which we can easily prove:

We can simply add a counter to our universal programs to determine when we have simulated $t$ steps.