Pumping lemma for CFG:

\[ \forall \text{CFG} \exists p \forall w \left( \text{len}(w) \geq p \rightarrow \right. \]

\[ \exists u, v, x, y, z \left( w = uvxyz \land \text{len}(vy) > 0 \land \text{len}(vx) \leq p \land \forall \epsilon \left( uv^\epsilon xy^\epsilon z \in L \right) \right) \]
We select 2 lowest repetitions of the same variable on the same branch.

\[ u = 1 \]
\[ v = 0 \]
\[ x = \wedge \]
\[ y = 1 \]
\[ z = 0 \]
Why?

Who was Turing?

What are Turing machines?

1932

1934

1936

1939

Read (TA)

unary code: 0 - 0, 1 - 1, 2 - 11, 3 - 111, 4 - 1111,

Add 1 in unary code

we have

we want.
start, \_ \rightarrow R, \text{ work}
work, 1 \rightarrow R
work, \_ \rightarrow 1, \text{ back, L}
back, 1 \rightarrow L
back, \_ \rightarrow \text{ halt}
Subtract in unary code:

we go right until we see blank,
then we go back one step
and then replace 1 at blank & start going back

\[
\text{Ready to delete}
\]
start, \( \rightarrow \) R, working
working, \( \rightarrow \) R
working, \( \rightarrow \) L, readyToDelete
readyToDelete, \( \rightarrow \) w, back, L
back, \( \rightarrow \) L \( \rightarrow \) halt

Binary numbers: \[6_{10} = 110_2\]

We write starting with most significant digit.
Most computers start with least significant digit.

\[011\]

Because all arithmetic operations start with least significant digits.

How do you add 1 in binary code?

\[\begin{array}{c}
10111 \\
\hline
11111 \\
\hline
11000 \\
\end{array}\]

\[24_{10} \quad 1000\]

Algorithm: you add 1. You start at least significant digit, replace 1s with 0s until you see 0 or \( \bar{1} \), then you replace them with 1, and stop.

Hardware supported
(start, \_ \rightarrow R, \text{working}
working, 1 \rightarrow 0, R
working, 0 \rightarrow 1, \text{back, L}
back, 0 \rightarrow L
back, \_ \rightarrow \text{halt}

\begin{align*}
100011 & \\
101011 & \\
101001 & \\
101001 & \\
\end{align*}

\begin{align*}
& \begin{array}{c}
1001 \quad 1000 \quad 8 \quad 8 \quad 1000_2 = 8
\end{array}
\end{align*}