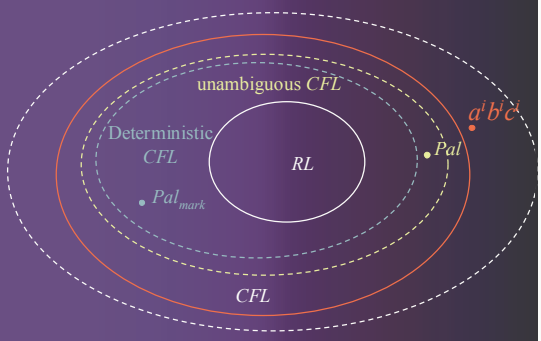
# Session 25

Turing Machines
and
Chomsky Hierarchy

## Non-CFL



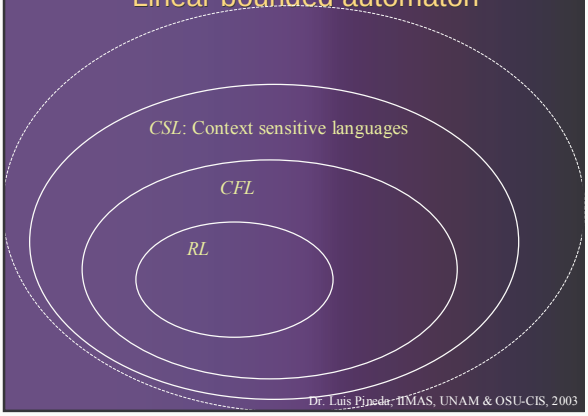



## Two Stacks PDA

- Processing $a^i b^i c^i$:
  - Push all $a$'s in stack 1
  - For each $b$ pop one $a$ and push a $b$ in stack 2
  - eventually $a^i = b^i$
  - For each $c$ pop one $b$ from stack 2
  - eventually $b^i = c^i$
- But the language is not a CFL
- The 2 stacks PDA is not a PDA



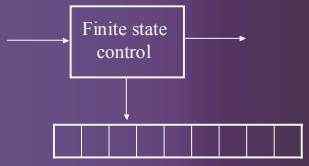
## Linear bounded automaton






## Turing Machine

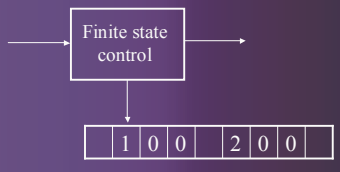
- Moves: Depending on current state and symbol in the tape:
  - Select next state
  - Write symbol or move left or move right





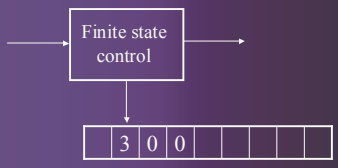

## Turing Machine

- Computing a function:
  - Initial state: the arguments



## Turing Machine

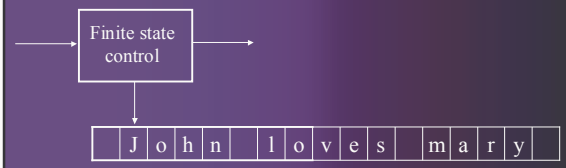
- Computing a function:
  - Final state: the value






## Turing Machine

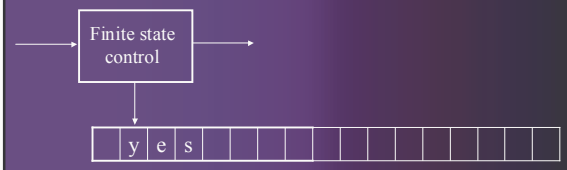
- Interpreting language is computing a function:






## Turing Machine

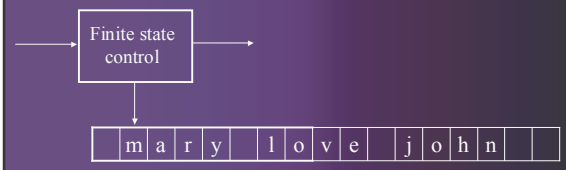
- Interpreting language is computing a function:






## Turing Machine

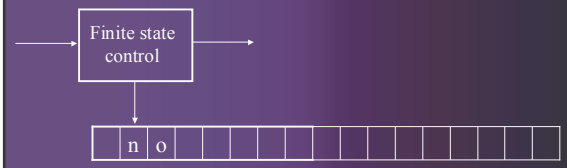
- Interpreting language is computing a function:






## Turing Machine

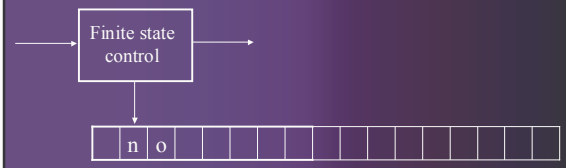
- Interpreting language is computing a function:





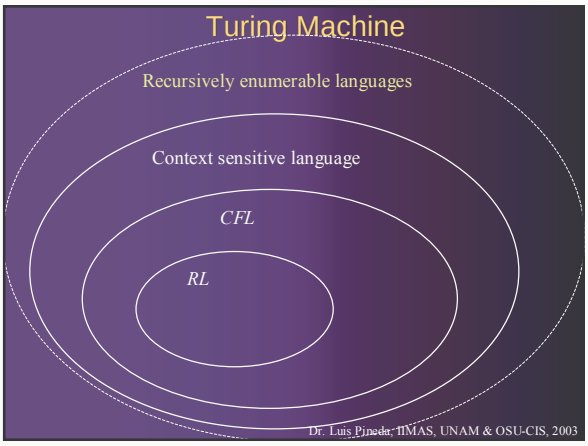

## Turing Machine

- Interpreting language is computing a function:



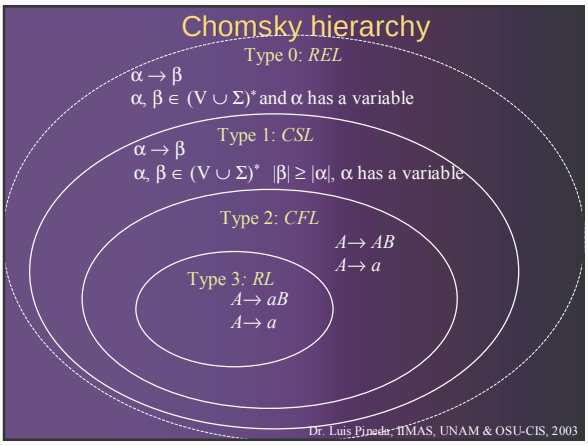

- There is a TM for every function
- Knowing a language is knowing a function!

## Turing Machine






## Chomsky hierarchy






## The halting problem

- If the tape is rewritten we cannot tell whether the computation will ever stop!
- The halting problem cannot be solved by a Turing machine



## Church thesis

- The set of function computed by Turing Machines correspond to the set of functions that can be computed intuitively by people
- If a computational device is general enough to compute this set of functions, it is equivalent (can be reduced) to a Turing Machines



## Church thesis

- Equivalent formalisms:
  - Turing Machines
  - Theory of Recursive Functions
  - Abacus computation
  - Lambda Calculus