

## Notation of Algorithm

## Closure Properties of Decidable Languages

- Theorem: Turing-decidable (recursive) languages are closed under complement,  $\cap$ ,  $\cup$

Proof: 1. complement

2.  $\cap$

3.  $\cup$

## Closure Properties of Turing-recognizable Languages

- Theorem: Turing-recognizable (recursively enumerable) languages are closed under  $\cap$ ,  $\cup$

Proof: 1.  $\cap$

2.  $\cup$

- Theorem: Turing-recognizable languages are NOT closed under complement.

## The Big Picture

- The relationship among classes of languages

## Definition of Algorithm

- An algorithm — informally, a collection of simple instructions for carrying out some task.
- *Church-Turing Thesis*:  
Intuitive notation of algorithms  $\equiv$  Turing machine algorithms
- Formal definition of the notion of algorithm
  - Turing machine
  - $\lambda$ -calculus

## Levels of Description of TM Algorithms

- Formal description — state diagram; lowest, most detailed.
- Implementation description — use English to describe the way that Turing machine moves its head and the way it stores data on its tape.
- High-level description — use English to describe an algorithm, ignoring the implementation model.
- Examples

## Algorithmic Solvability

- What problems can be solved algorithmically and what cannot.
- Why should we study unsolvability?
  - Make unsolvable problems solvable.
  - Gain an important perspective on computation.

## Computational Complexity

- A decidable problem (computationally solvable in principle) may not be solvable in practice.
- Computational complexity: the requirement of time, memory, or other resources for solving a problem.
  - Measuring the time used to solve a problem
  - Classifying problems according to the amount of time required