# Introduction to Logic (1/2)

Moonzoo Kim

CS Division of EECS Dept.
KAIST

moonzoo@cs.kaist.ac.kr
http://pswlab.kaist.ac.kr/courses/cs402-07

# Why is logic important to software engineers?

- Mathematics v.s. logic
- Difficulty of building correct software
- Logic in a software development framework
- Necessity of rigorous requirement specification
- A programming language as a specification language?

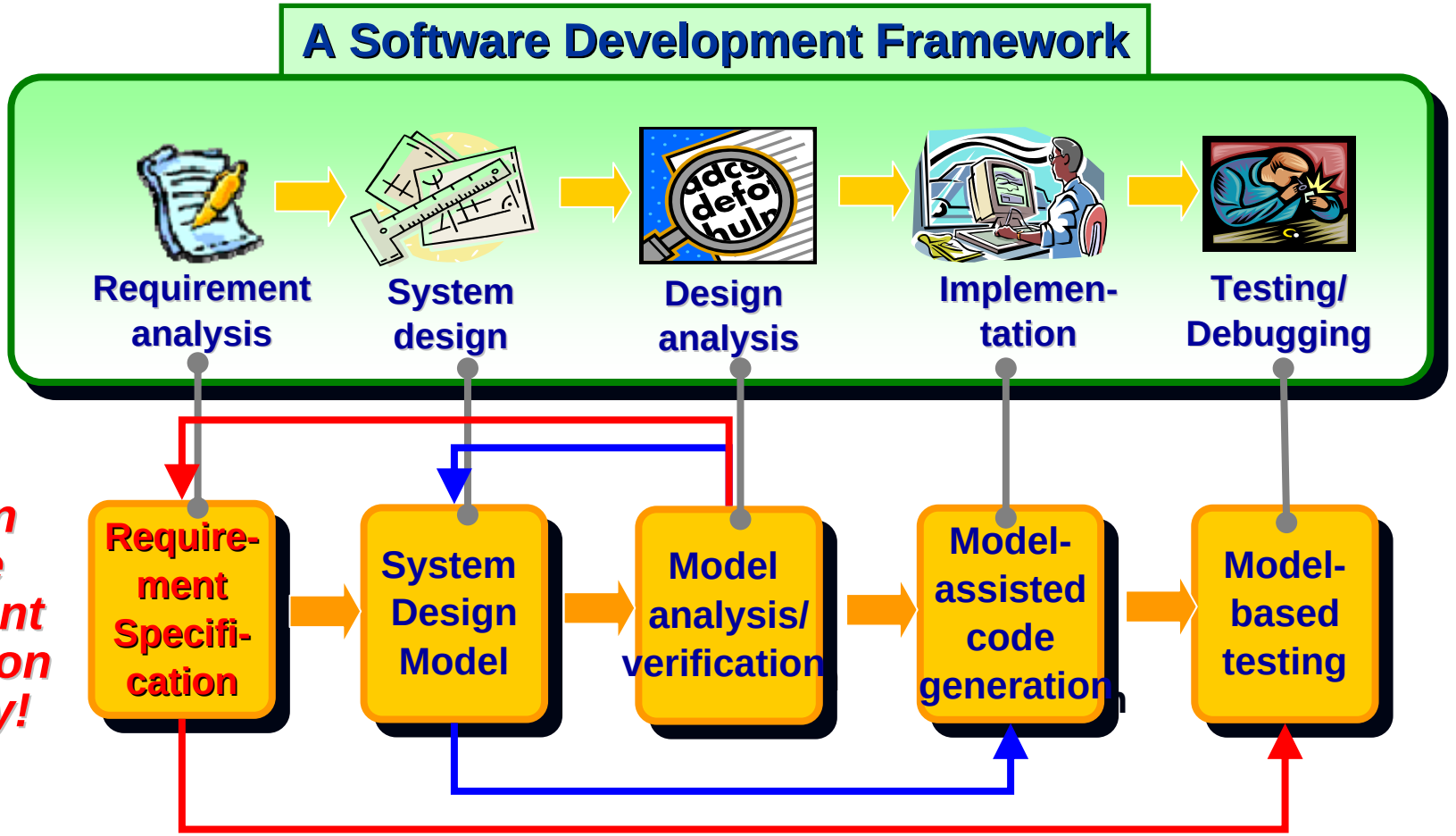
# Mathematics v.s. Logic

- Mathematics is an essential skill for all engineering fields
  - Electrical engineering, mechanical engineering, chemical engineering, etc
  - The kind of mathematics for the above fields studies continuous behaviors
    - Goals are to get a numeric solution for equations.
      - Differential equations, integral calculus, probability, etc
      - $\vec{F} = \frac{d(m\vec{v})}{dt}$ $\int_0^\infty \frac{dv}{dt} m^2$
- Logic is a mathematics for computing systems
  - System engineering, computer science, software engineering, etc
  - Logic studies discrete behaviors
    - Goals are to prove whether a given logic formula is true or not
      - Proposition logic, first order logic, temporal logic, etc
      - $p \wedge q$, $\forall x \exists y\ x<y$ (*there exists a maximal number*), $\Box$ (x>10) (invariance)

# (Relative) Difficulty of Building Correct Software

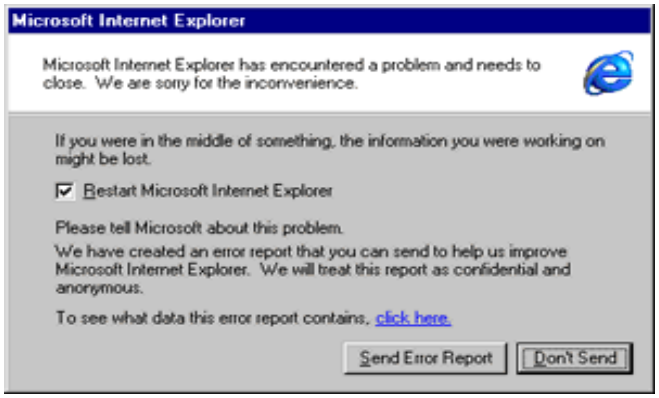
- Most engineering fields can control systems based on feedback because of their continuous behaviors
- Most engineering products handles systems of static structures
- Every engineering field has well-founded scientific disciplines
  - Mechanical Engineering
    - Newton's laws
    - Numerical analysis methods
  - Electrical Engineering
    - Circuit algebra
    - Electronic design automation (EDA)
  - Computer Science
    - Mathematical/Computational logic
    - Formal methods
- Computer system must guarantee their correctness before deployment, because computer system cannot rely on feedback to avoid failure due to its discrete behavior.
- Software handles systems of dynamic structures
- Products of all engineering fields provide warranty *except computer science*
  - Sonata has 10 years engine warranty
  - Pentium processor has 3 years warranty
  - MS Windows does not provide any warranty
    - Use it at your own risk!!!
- Why?
  - Complexity of SW is far greater than products of other disciplines
  - Failure to follow scientific disciplines when developing SW

# A Software Development Framework

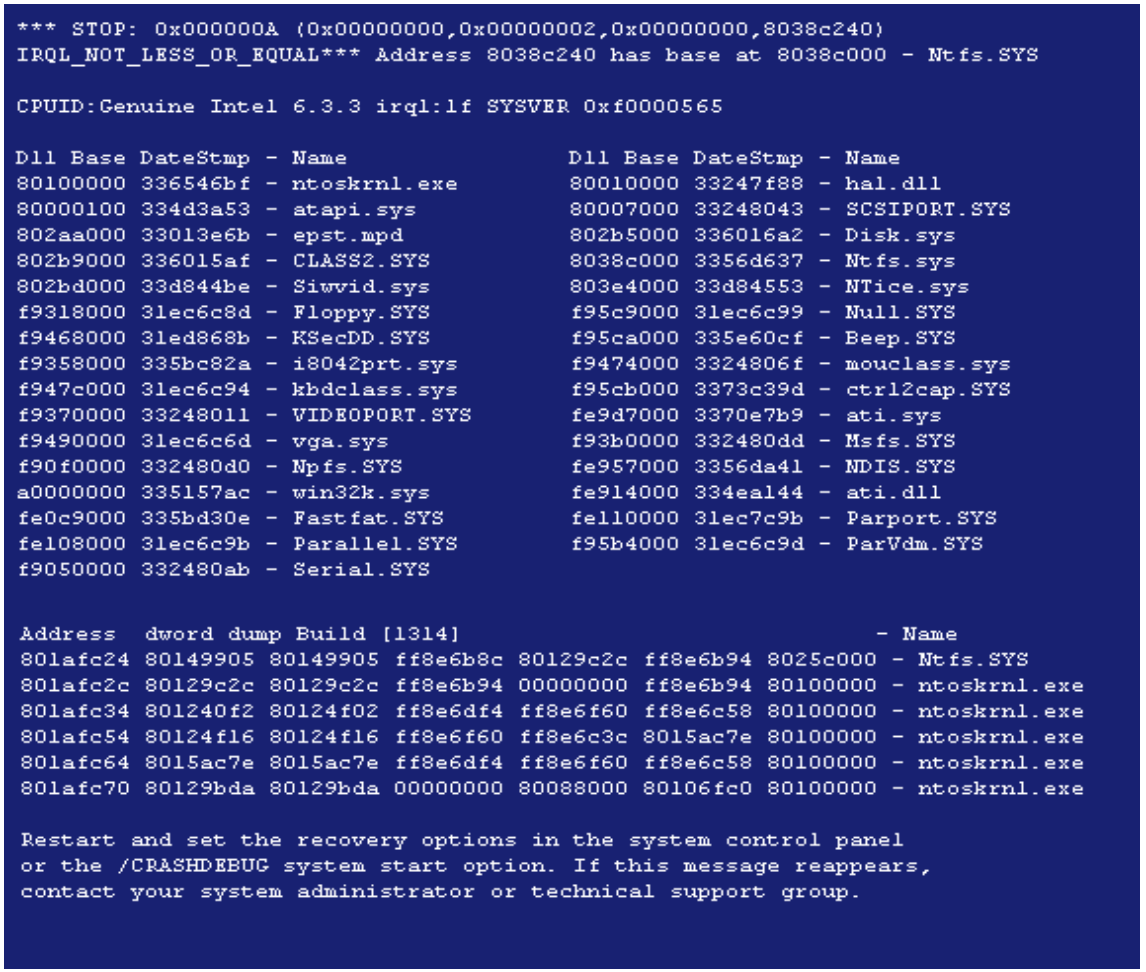



***Logic can describe requirement specification rigorously!***

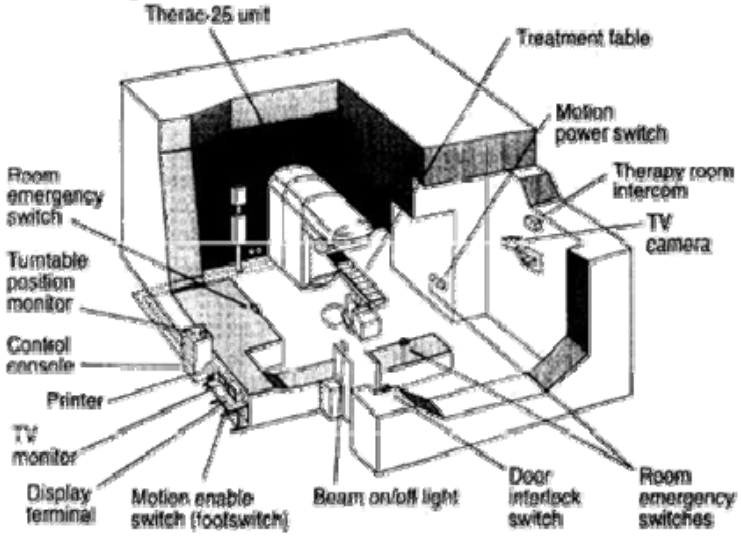
# Notorious “Blue Screen”

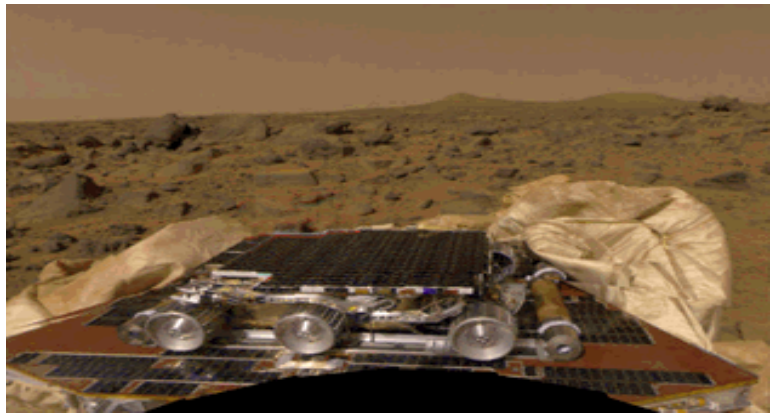






# Tragic Accidents Caused by SW Bug




June 2002

"Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated $59.5 billion (60조원) annually, or about 0.6 percent of the gross domestic product

…

At the national level, over half of the costs are borne by software users and the remainder by software developers/vendors."

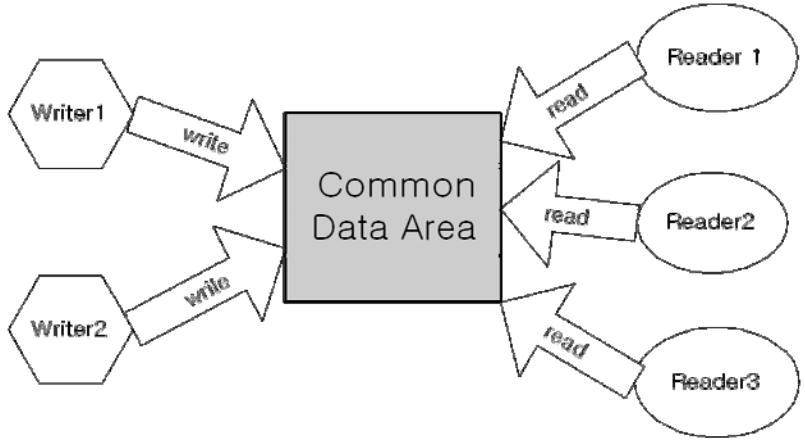
…

The study also found that, although all errors cannot be removed, more than a third of these costs, or an estimated $22.2 billion, could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects."

***NIST Planning Report 02-3***
***The Economic Impacts of Inadequate Infrastructure for Software Testing***

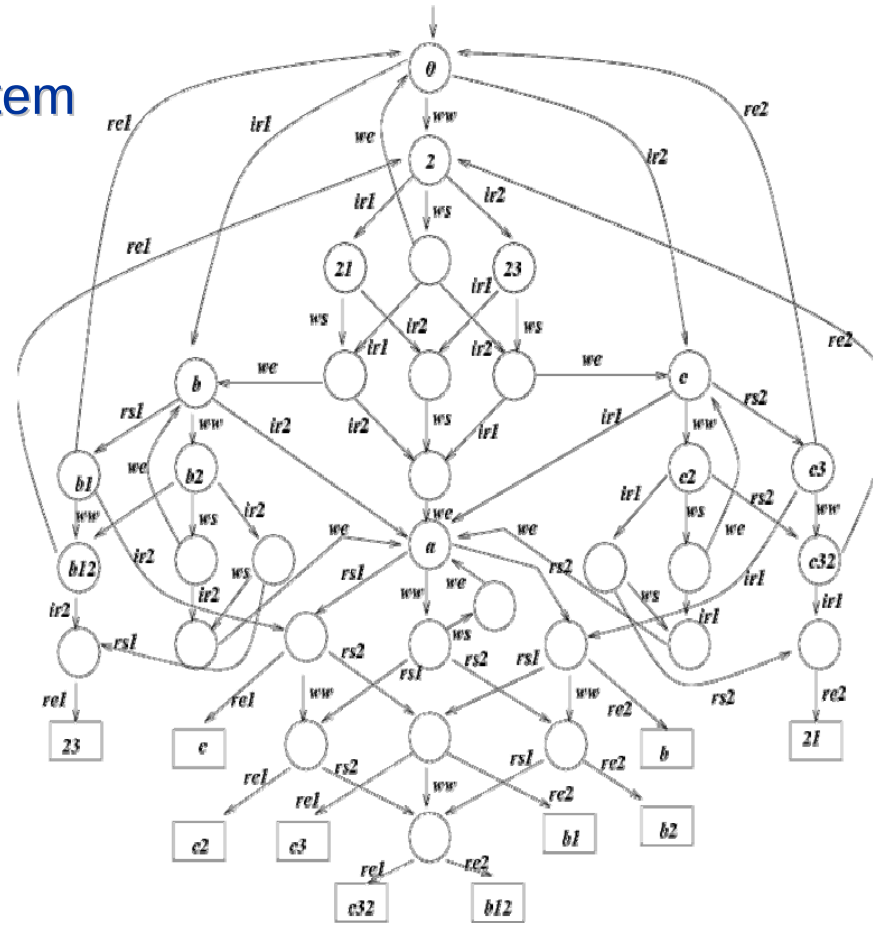
# Requirement Specification

- Requirement specifications are the goals that a target software must satisfy
- Ex> For a system containing 3 readers, 2 writers, and common common data area, the system should satisfy the following three requirement properties
  - Concurrency (CON)
    - Multiple readers can read data concurrently
  - Exclusive writing (EW)
    - A writer can write into the data area at an instant with no readers
  - High priority of a writer (HPW)
    - A writer's request should have a higher priority than that of a reader



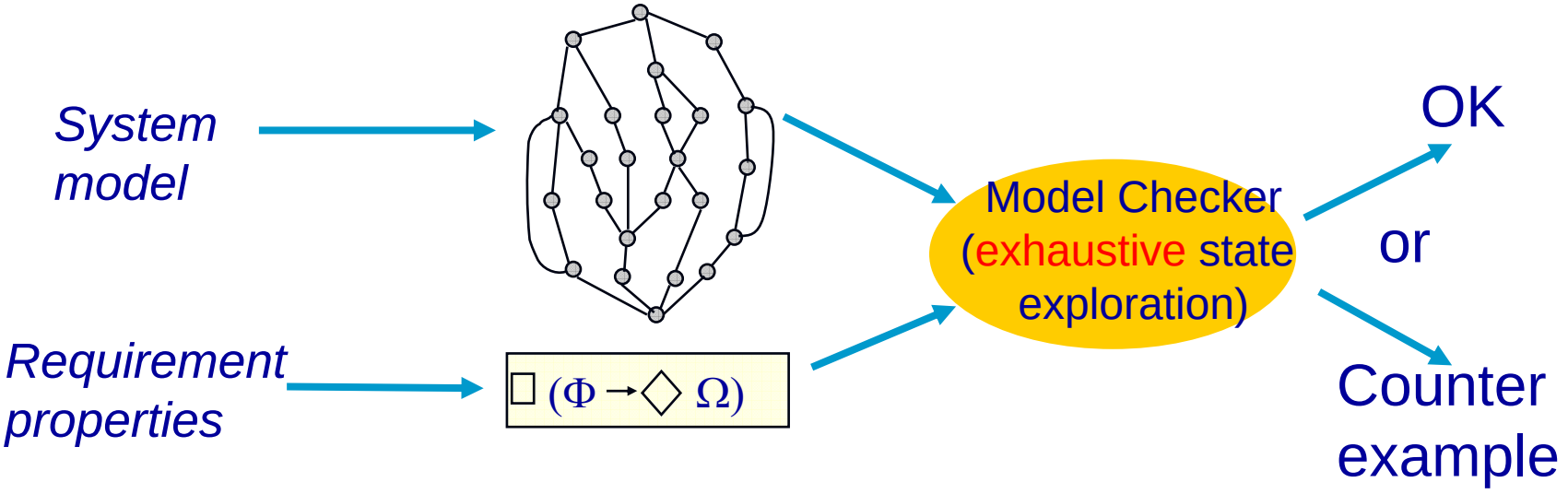
# System Design Model

- Abstract description of a target system
- Model must have clear meaning/semantics
- Ex. A system design model for 2 readers and one writer in process algebra
  - System = ($R_1$| $R_2$| W|D)\{…}
    - $R_i$=check_lock. read_request$_i$. start_read$_i$. end_read$_i$.$R_i$
    - $W_i$= check_lock. write_request. start_write. end_write.W
    - D = ‘read_request$_1$. …



# Design Analysis

- We definitely want to guarantee/prove that a system design model *M* satisfies a requirement property $\phi$
  - $M \models \phi$



# Necessity of Rigorous Requirement Specification 1/4

- Specifications in natural language (ex. 한국어, English, 日本語, etc) can be easily ambiguous, inconsistent, and incomplete
- Connectives in natural languages can have several meaning
  - John eats an apple and Mary eats an orange
  - John drove on and hit Mary
  - The sun is shining and John feels happy
  - John is a Korean or a computer scientist
  - John is working or he is at home

***"and" may have an ordering information in time***

***"and" may indicate a cause and result***

***"or" may have an exclusive meaning***

# Necessity of Rigorous Requirement Specification 2/4

- Specification in natural languages can be easily incomplete due to assumption of implicit “common senses” which actually do not exist
- Ex. For the 3-readers and 2-writers system in the page 8
  - Concurrency (CON): “Multiple readers can read data concurrently.”
    - What if only 2 readers can read concurrently?
  - Exclusive writing (EW): “A writer can write into the data area at an instant with no readers”
    - What if two writers write into the data area at the same time?
  - High priority of a writer (HPW): “A writer’s request should have a higher priority than that of a reader”
    - What if a writer requests one second later than a reader? Should the system wait for handling readers request? If so, how long?

# Necessity of Rigorous Requirement Specification 3/4

- Informal specification can easily result in erroneous systems
- Example (retail chain management software)
  - If the sales for the current month are below the target sales, then a report is to be printed, unless
    - the difference between target sales and actual sales is less than half of the difference between target sales and actual sales in the previous month
    - or if the difference between target sales and actual sales for the current month is under 5 percent.
  - Find two ambiguous points

# Necessity of Rigorous Requirement Specification 4/4

- Suppose that $R_i$ means i th reader is reading.
- The requirement $\phi_{CON}$ for concurrency can be written in propositional logic
  - If it is ok for two readers to read concurrently, $\phi_{CON}$ should be
    - $(R_1 \wedge R_2) \vee (R_2 \wedge R_3) \vee (R_3 \wedge R_1)$ for some time instant t
    - Note that if it is ok for only two readers to read concurrently, $\phi_{CON} = (R_1 \wedge R_2 \wedge \neg R_3) \vee (\neg R_1 \wedge R_2 \wedge R_3) \vee (R_1 \wedge \neg R_2 \wedge R_3)$
  - If all three readers should be able to read concurrently, $\phi_{CON}$ should be
    - $(R_1 \wedge R_2 \wedge R_3)$ for some time instant t

# A Programming Language as a Specification Language?

- It is not impossible to describe a requirement specification in a conventional programming language such as C/C++ or Java
- But this approach has several disadvantages
  - Conventional programming languages were designed to specify what to do in a low-level detail
    - Complex to describe requirement specifications
  - Conventional PL focuses on describing step-by-step execution (i.e., imperatively), not goals of executions.
    - Inconvenient to describe requirement specification
- Logics have been designed to specify/declare req. specifications. So use what is designed for its original purpose!!!
- Logic has formal syntax and formal semantics
- Ex. For propositional logic
  - Syntax: $\phi = p \mid \neg \phi \mid \phi \wedge \phi \mid (\phi)$
  - Semantics:
    - $[p]_s$= true if a proposition $p$ is true; false otherwise
    - $[\phi_1 \wedge \phi_2]_s$ = true if $[\phi_1]_s = [\phi_2]_s$ = true; false otherwise
    - $[\neg \phi]_s$ = true if $[\phi]_s$ = false; false otherwise