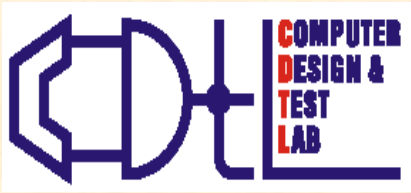
Formal Methods for Software Engineering

Virendra Singh
Computer Design and Test Lab
Indian Institute of Science
Bangalore
Email: virendra@computer.org



Introduction

- Problems in software development
- Formal methods for the problems
- Challenges to formal methods
- Formal engineering methods for the challenges

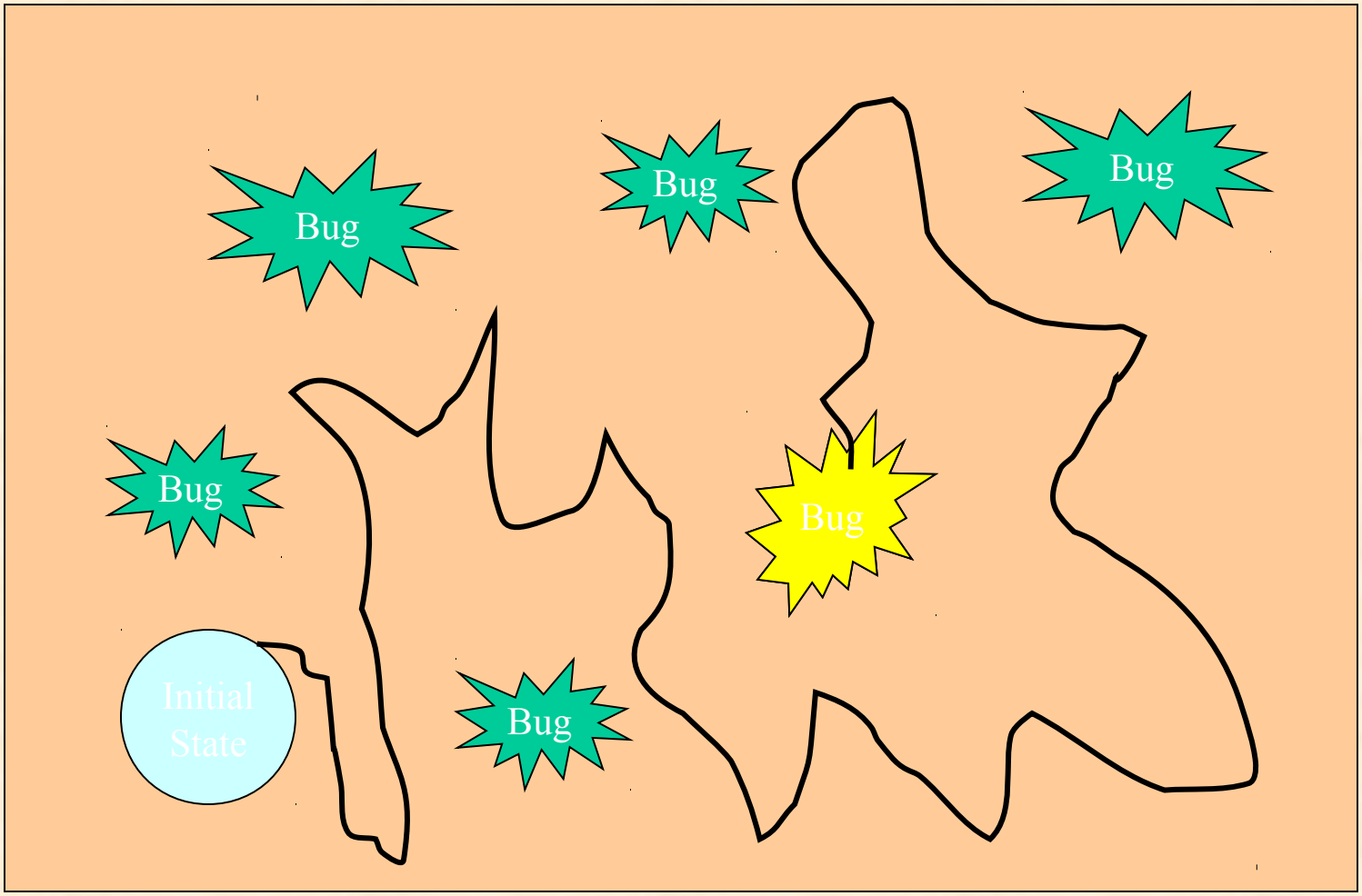
Formal

- Definite
- Orderly
- Methodical
- Some thing which is methodical and done with discipline

Famous Bugs

- History of software is full of notorious bugs
- Long list
  1. 1987: Therac-25
  2. 1990: AT&T long distance breakdown
     - Break statement
  1. 1991: Patriot Missile
     - 28 People killed
- Pentium Bug

Program Testing



Software Testing vs Formal Verification

Program testing can be used to show the presence of the bugs, but never to show the absence!

(E.W. Dijkstra)

Program Testing

- Program: Check the equality
- isEqual (“House”, Mouse”)
- Is Equal (“Testing”, “Testing”)
- isEqual(“house”, “home”

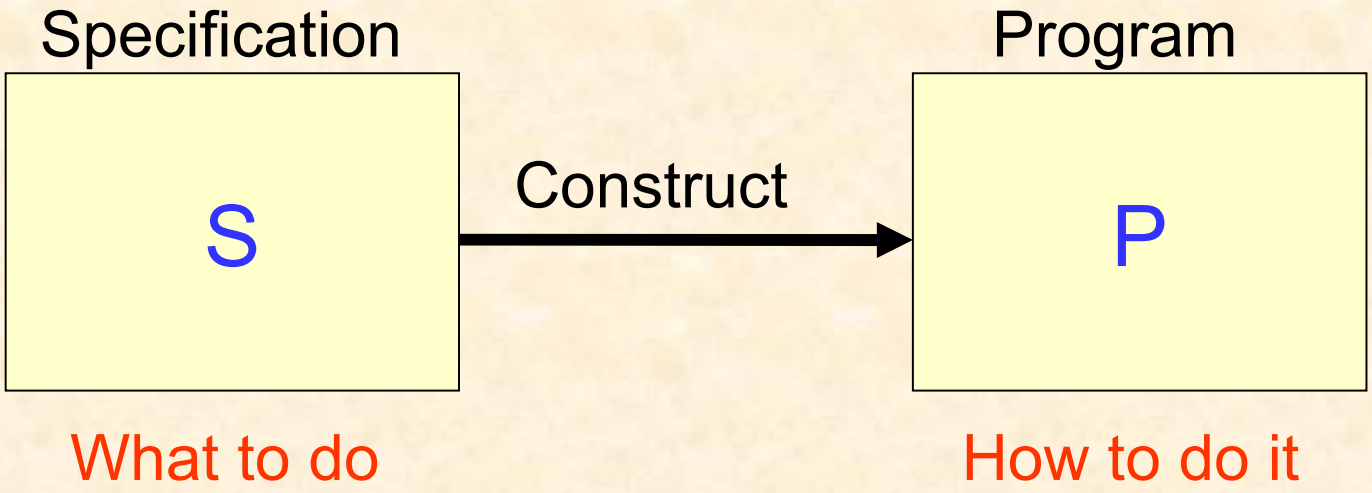
Program Testing

- Program: Check the equality
- equal = strlen(string1) == strlen (String2);
- If (equal)
  - for (i=0; i < strlen(String1); i++)
    - Equal = string1[i] == string 2[i];
- Return equal;

Program Testing

- Sorted = false / /1
- While (!sorted){ //2
- Sorted = true;
- For (int i = 0; i < SIZE-1; i++){ //3
  - if (a[i] > a[i+1]{ //4
    - Swap (a[i], a[i+1]) //5
    - Sorted = false;
      } //6
  - }//7
- } //8

Problems in software development



- How to ensure that S is not ambiguous so that it can be correctly understood by all the people involved?
- How can S be effectively used for inspecting and testing P?
- How can software tools effectively support the analysis of S, transformation from S to P, and verification of P against S?

An example of informal specification:

"A software system for an Automated Teller Machine (ATM) needs to provide services on various accounts. The services include operations on current account, operations on savings account, transferring money between accounts, managing foreign currency account, and change password. The operations on a current or savings account include deposit, withdraw, show balance, and print out transaction records."

A better way to write the same specification:

"A software system for an automated teller machine (ATM) needs to provide services on various accounts.

The services include

a. operations on current account
b. operations on savings account
c. transferring money between accounts
d. managing foreign currency account,
e. change password.

The operations on a current or savings account include

a. deposit
b. withdraw
c. show balance
d. print out transaction records."

The major problems with informal specifications:

- Informal specifications are likely to be ambiguous, which is likely to cause misinterpretations.
- Informal specifications are difficult to be used for inspection and testing of programs because of the big gap between the functional descriptions in the specifications and the program structures.
- Informal specifications are difficult to be analyzed for their consistency and validity.
- Information specifications are difficult to be supported by software tools in their analysis, transformation, and management (e.g., search, change, reuse).

A possible solution to these problems:

Formal Methods!!!

Formal methods for the problems

What is formal methods?

Formal methods = Formal Specification

+

Refinement

+

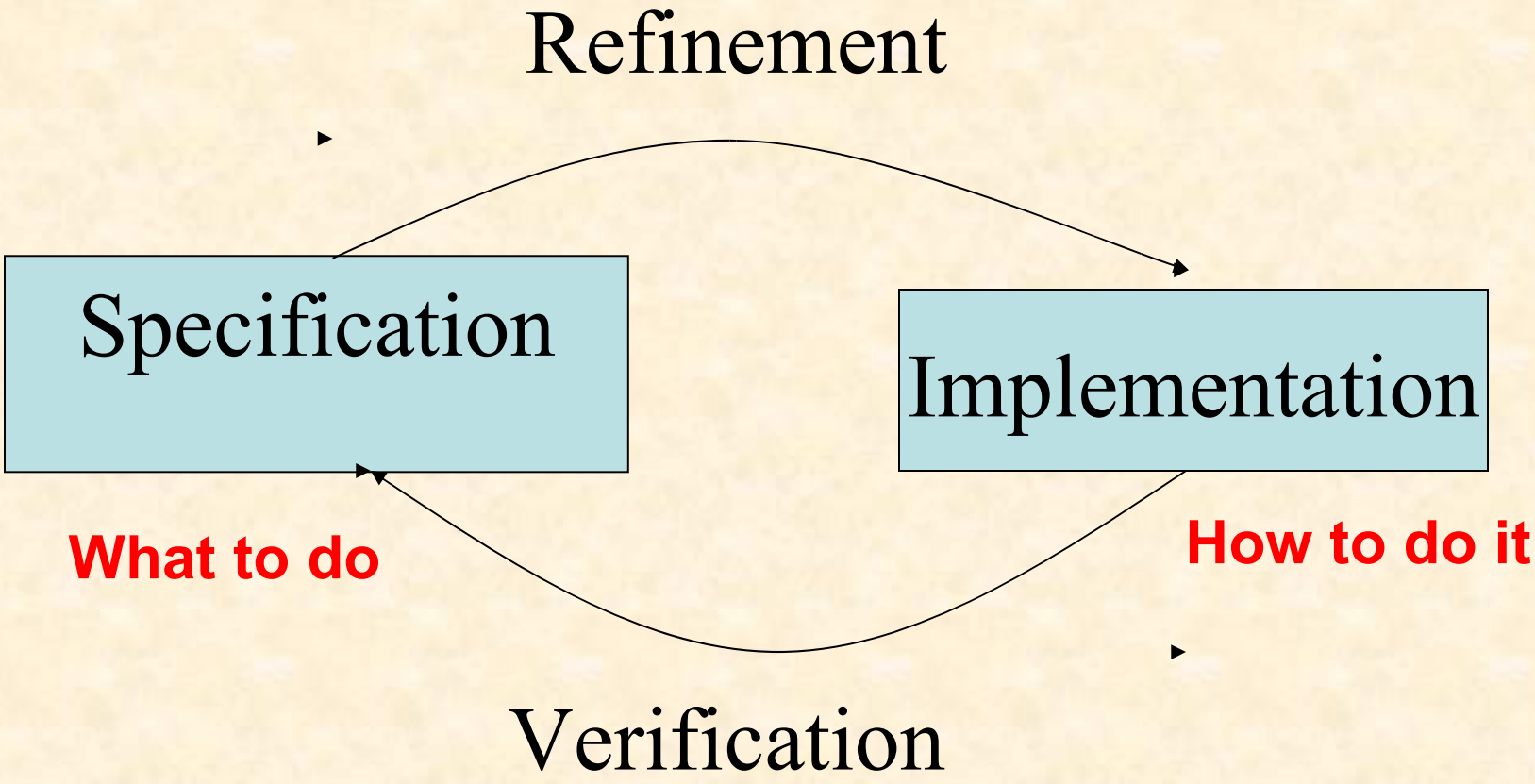
Formal Verification

Set theory, logics, algebra, etc.

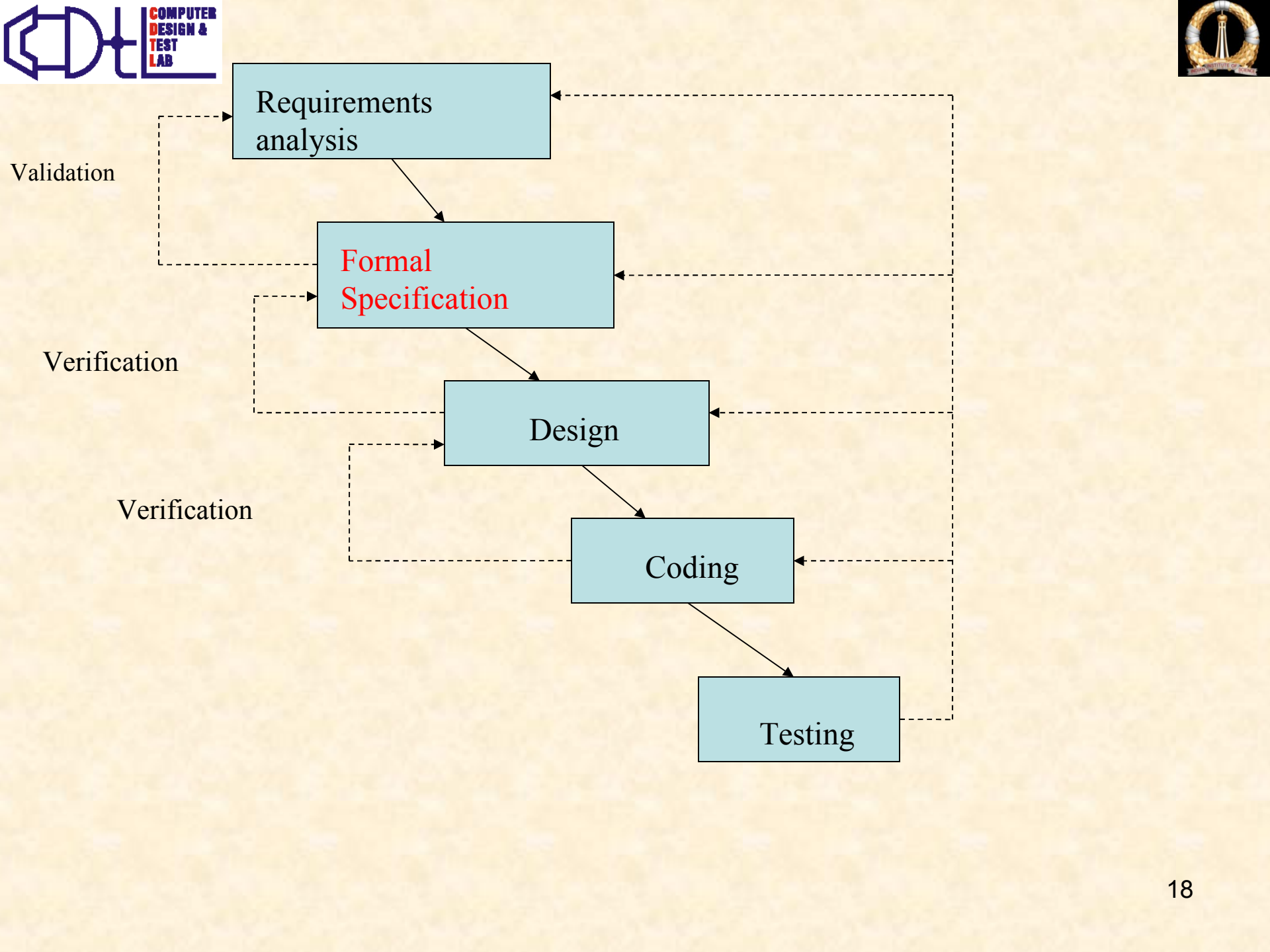
Formal methods can also be understood as the following three components:

- Formal notation (or language) for writing specifications.
- Logical calculus for formal verification (or proof)
- Method for developing software systems.

From the abstract to the concrete



Check the correctness


Requirements analysis
Validation
Formal Specification
Verification
Design
Verification
Coding
Testing

The question of our interest is:

How to write a formal specification?

Many formal notations have been developed for writing formal specifications and the most commonly used ones include VDM, Z, and B.

The most commonly used formal methods

(1) VDM-SL (Vienna Development Method – Specification Language),
IBM Research Laboratory in Vienna

References:

(1) "Systematic Software Development Using VDM'', by Cliff B. Jones, 2nd edition, Prentice Hall,1990.

(2) "Modelling Systems", by John Fitzgerald and Peter Gorm Larsen, Cambridge University Press,1998.

Operation specification

OperationName(input)output

ext State variables

pre preconiditon

post postcondition

Example:

Add(x : nat) y : nat

ext rd z : nat /*z is an external variable */

pre true

post y > x + z

Operations are organized into modules:

```
module A
 local variables declarations
 invariant declarations

 operation specification1;
 operation specification2;
 …
 operation specificationn;
end
```

(2) Z, PRG (Programming Research Group), the University of Oxford, UK

References:

(1) "The Z Notation", by J.M. Spivey, Prentice Hall, 1989.

(2) "Using Z: Specification, Refinement, and Proof", by Jim Woodcock and Jim Davies, Prentice Hall, 1996.

A Z specification is composed of a set of schemas and possibly their sequential compositions.

A schema can be used to define global variables, state variables, and operations.

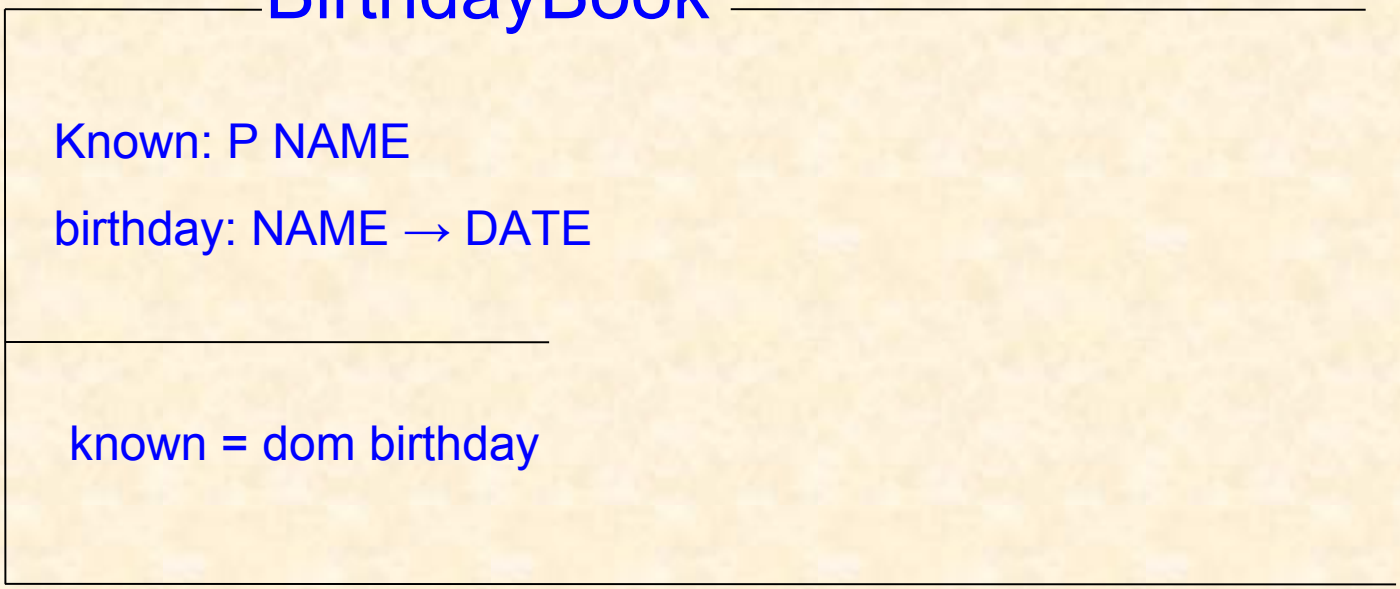
Axiomatic schema for defining global variables:

$$\begin{array}{|l} age: \mathbb{N} \\ \hline age > 0 \end{array}$$

declaration

predicate

A schema for defining state variables:



A schema for defining an operation:

AddBirthday

$\triangle$BirthdayBook

name?: NAME

date?: DATE

---

name? $\notin$ known

birthday' = birthday $\cup$ {name? $\rightarrow$ date?}

(3) B-Method,

Jean-Raymond Abrial, France

References:

(1) "The B-Book: Assigning Programs to Meanings", by J-R Abrial,
Cambridge University Press,1996,

A B specification is composed of a set of related abstract machines. Each abstract machine is a module that contains a set of operation definitions. Each operation is defined using pre- and postconditions.

1.3 Challenges to formal methods

- Formal specifications of large-scale and complex software systems can be difficult to write, to read, and to understand for many engineers in industry.
- Communications between clients and developers via formal specifications can be difficult.
- Modifications of formal specifications for consistency during a project can be time-consuming and costly.
- Formal verification is difficult to perform and is not cost-effective for the assurance of program correctness.
- The tool support does not necessarily reduce the difficulty of using formal methods.

Formal engineering methods for the challenges

Formal Engineering Methods (FEM) provide a way to integrate Formal Methods into the entire software development process to achieve rigor (methodology), comprehensibility (human), and tool supportability (tool) of software process.

Formal Methods

Application of Formal Methods in Software Engineering

Formal Engineering Methods

The difference between FM and FEM

FM answers the question:

what should we do and why?

FEM answers the question:

what can we do and how?

Component

Architecture