Informal Specifications

- Natural language specifications
- Natural language is wrought with ambiguity
  - Syntax and semantics open to interpretation
- Implementation is often difficult
  - One is never sure of completeness/correctness
- Easy to write
  - Not too much special training required

Formal Specifications

- Syntax and semantics formally defined
  - Some form of math
- Easy to show completeness and correctness
  - Formal proofs
- Difficult to write
  - Requires training and math competence
- Lightweight tools are beginning to emerge

Formal Specs: Roadmap

- Graphical Languages
  - Finite State Machines
  - Statecharts
  - Petri Nets
- Textual
  - Algebraic specs
    - Larch
  - Model-based specs
    - Z
    - Eiffel
    - RESOLVE
Finite State Machines

- Describe system in terms of **states** it can be in
- Define **transitions** between states
- State Transition Diagram
  - \( J \): set of states
  - \( K \): set of inputs
  - \( T \): transition function
  - \( IS \): initial state
  - \( FS \): final state

FSM: Example

- Safe combination lock
  - 3 positions: 1, 2, 3
  - Dial can be turned left or right: L, R
  - Correct combination: 1L, 3R, 2L
  - Any other combination sets off alarm
- STD
  - \( J \) = \{LOCKED, A, B, ALARM, UNLOCKED\}
  - \( K \) = \{1L, 1R, 2L, 2R, 3L, 3R\}
  - \( IS \) = \{LOCKED\}
  - \( FS \) = \{UNLOCKED, ALARM\}

Algebraic Specification

- Defines a type in terms of relationships between its operations
  - Introduction: declares type name
  - Description: Informal description
  - Signature: Syntax of operations
  - Axioms: Semantics of operations
Algebraic Spec Process

- Specification structuring
- Specification naming
- Operation selection
- Informal operation specification
- Syntax definition
- Axiom definition

Spec example: Stack

```
sort Stack
imports INTEGER

Defines a stack component. Elements are accessed in a last-in-first-out order.

New -> Stack
Push(Stack, Elem) -> Stack
Pop(Stack) -> Stack
Top(Stack) -> Elem
Length(Stack) -> Integer

Pop(New) = Undefined
Pop(Push(s, e)) = s
Top(New) = Undefined
Top(Push(s, e)) = e
Length(New) = 0
Length(Push(s, e)) = Length(s) + 1
```

Example: Stack

- Stack: Last-In-First-Out
- Operations
  - New (creates new stack)
  - Push (pushes element into stack)
  - Pop (removes top element from stack)
  - Top (peeks at the top element in stack)
  - Length (queries number of elements in stack)

Model-Based Specification

- Specification of a component based on a mathematical model that represents it
- Model should be representative of all operations
- Model helps visualize operations
Z (pronounced “zed”)• Specification language to enable model-based specification• Defines state of a component and operations• Each operation specifies how state is altered

Example: Birthday Book• A small database for recording people’s names and birthdays.
  • The system should allow us to:
    – add new people to database
    – look up birthday of a person
    – find names of people born on given day

Structure of Z specs• Schemas describe:
  – what states can a system occupy
  – what operations can happen
  – relationships between parts of complex system• Informal text:
  – motivates formal descriptions
  – relates model to reality
  – documents requirements

Schema: Birthday Book

BirthdayBook
known : P NAME
birthday : NAME → DATE
known = dom birthday
**Initialization**

```
InitBirthdayBook
BirthdayBook'
birthday' = ∅
```

**Benefits of Z Specs**

- Easy to find faults (accompanying proof system)
- Requires exactness (no ambiguity, contradictions, etc)
- Reduces overall cost of development
- CASE tools for paraphrasing Z specs

**Operation: AddBirthday**

```
AddBirthday
△BirthdayBook
n? : NAME
d? : DATE

n? /∈ known
birthday' = birthday ∪ {n? → d?}
```

**Contract-Based Specs**

- Each operation is defined in terms of a contract
  - Pre-condition: What must be true before operation is called
  - Post-condition: What must be true when operation completes
- If pre-condition is satisfied, operation guarantees post-condition
Example: Stack

- Mathematical Model
  - String of Element
- Operations
  - New
  - Push(s, e)
  - Pop(s)
  - Top(s)
  - Length(s)

Stack Pop: Spec

\[
\begin{align*}
\text{Elem Pop(Stack } s) \text{) requires:} \\
& (|s| \neq 0) \\
\text{ensures:} \\
& \text{there exists } a: \text{String of Object, } \\
& b: \text{Object} \\
& (#s = \langle b \rangle \ast a) \text{ and} \\
& (s = a) \text{ and} \\
& (\text{Pop()} = b)
\end{align*}
\]