Advanced Software Engineering
(CS6401)

Autumn Semester (2024-2025)

Dr. Judhistir Mahapatro
Department of Computer Science and Engineering
National Institute of Technology Rourkela

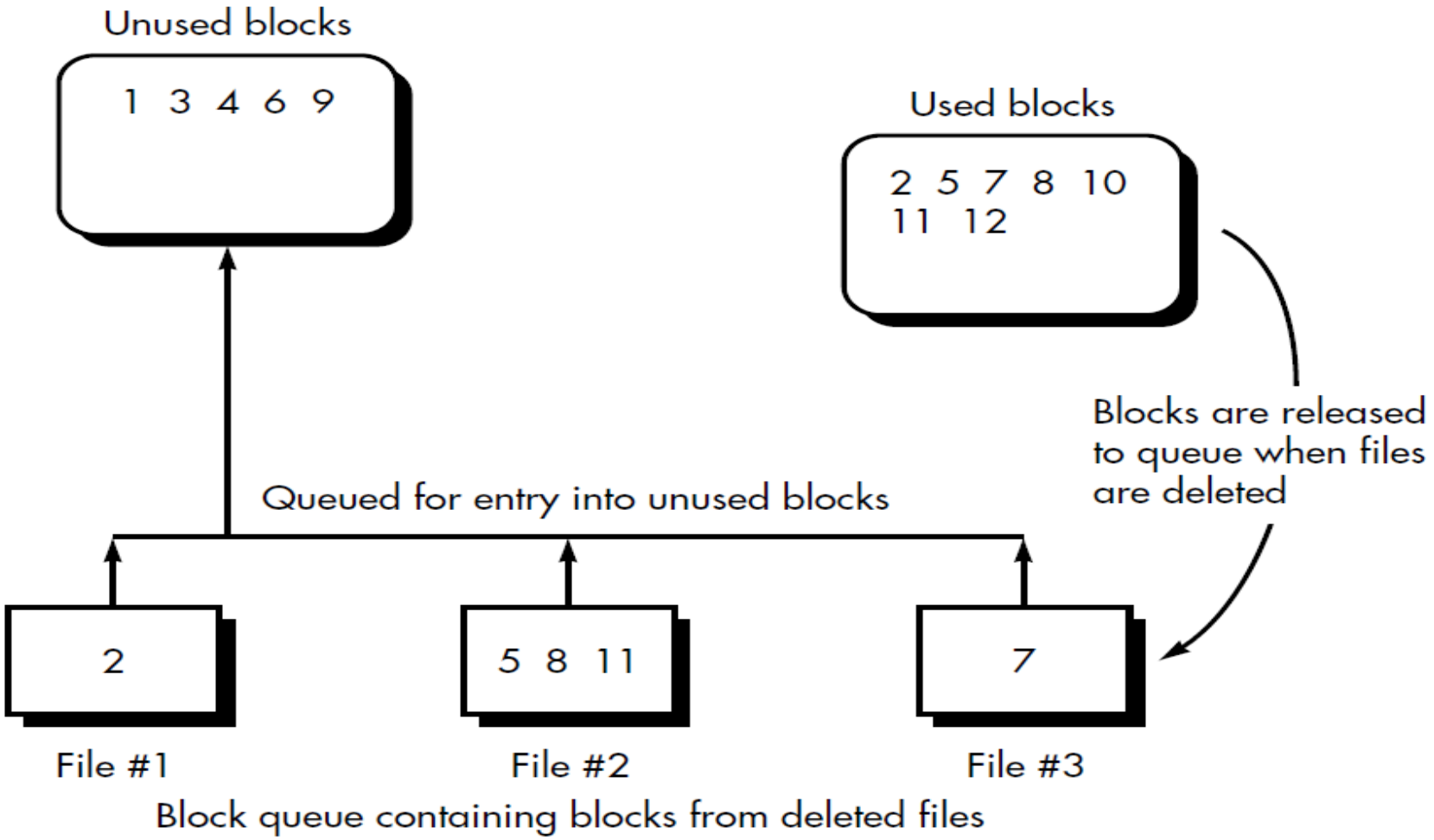
Formal methods

Topics covered in Previous Lecture:

- Axiomatic Specification
- Algebraic specification

The Z Specification Language

A Block Handler



Mathematical Notation for Formal Specification

- State is the **collection of free blocks**, **the collection of used blocks**, **the queue of returned blocks**.
- Data invariant, expressed in natural language, is
  - No block will be marked as both unused and used
  - **All the sets of blocks** held in the queue will be **subsets** of the collection of **currently used blocks**.
  - No elements of the queue contain same block#.
  - Used and unused blocks will be the total collection of blocks
  - No duplicate block numbers in unused blocks
  - No duplicate block numbers in used blocks

Mathematical Notation for Formal Specification

- Some operations associated with this data are:
  - **Add()** a collection of blocks to end of the queue
  - **Remove()** a collection of used blocks from front of the queue and place them in the collection of unused blocks
  - **Check()** whether the queue of blocks empty.
  - Precondition of **Add()** is that blocks to be added must in the collection of used block.
  - Precondition of **Remove()** is that the queue must have at least one item in it.

Mathematical Notation for Formal Spec.

- A set *BLOCKS* will consist of every block number
- *AllBlocks* is a set of blocks
- Two sets: *used*, and *free*
- A sequence will contain sets of blocks that are ready to be released from files that have been deleted.
- State can be described as

$$used, free : \mathbb{P}\ BLOCKS$$

$$BlockQueue : \mathrm{seq}\ \mathbb{P}\ BLOCKS$$

- This is very much like declarations of program variables.

Mathematical Notation for Formal Specification

- The data invariant can be written as

$$used \cap free = \emptyset \land$$

$$used \cup free = AllBlocks \land$$

$$\forall i : \mathrm{dom}\ BlockQueue \bullet BlockQueue\ i \subseteq used \land$$

$$\forall i, j : \mathrm{dom}\ BlockQueue \bullet i \neq j \Rightarrow BlockQueue\ i \cap BlockQueue\ j = \emptyset$$

Mathematical Notation for Formal Specification

- First operation to be defined is one that removes an element from the head of the block queue. At least one item in the queue.

$$\#BlockQueue > 0,$$

- The post-condition is that the head of the queue must be removed and placed in the collection of free blocks and the queue adjusted to show the removal:

$$used' = used \setminus head\ BlockQueue \land$$
$$free' = free \cup head\ BlockQueue \land$$
$$BlockQueue' = tail\ BlockQueue$$

Mathematical Notation for Formal Specification

- A second operation adds a collection of blocks, Ablocks, to the block queue. The precondition is that Ablocks is currently a set of used blocks.

$$Ablocks \subseteq used$$

- The post-condition is that the set of blocks is added to the end of block queue and set of used and free blocks remains unchanged:

$$BlockQueue' = BlockQueue \frown \langle Ablocks \rangle \land$$
$$used' = used \land$$
$$free' = free$$

Z Specification Language

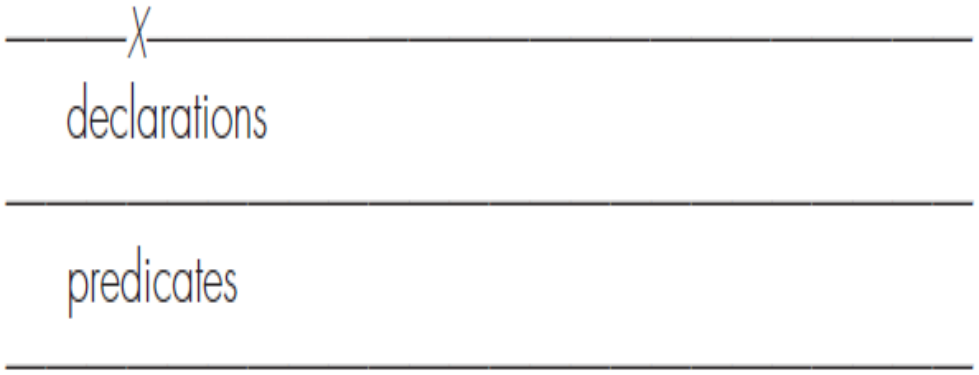
- It applies typed sets, relations, and functions within the context of first-order predicate logics to build schemas

- Z specifications are organized as a set of schemas
  - A language structure that introduces variables and specifies the relationship between these variables.

- A schema describes the stored data that a system accesses and alters.
  - In context of Z, this is called as the “State”

Z Specification Language

- A schema identifies the operations that are applied to change state and the relationships that occur within the system.
- A schema X is defined by the form



Z Symbols

Sets:

$S : \mathbb{P}\, X$	S is declared as a set of Xs.
$x \in S$	x is a member of S.
$x \notin S$	x is not a member of S.
$S \subseteq T$	S is a subset of T: Every member of S is also in T.
$S \cup T$	The union of S and T: It contains every member of S or T or both.
$S \cap T$	The intersection of S and T: It contains every member of both S and T.
$S \setminus T$	The difference of S and T: It contains every member of S except those also in T.
$\varnothing$	Empty set: It contains no members.
$\{x\}$	Singleton set: It contains just x.
$\mathbb{N}$	The set of natural numbers 0, 1, 2, ...
$S : \mathbb{F}\, X$	S is declared as a finite set of Xs.
max (S)	The maximum of the nonempty set of numbers S.

Z Symbols (Cont..)

Functions:

$f: X \rightarrowtail\!\!\!\!\!\!+ \; Y$	f is declared as a partial injection from X to Y
$\text{dom } f$	The domain of f: the set of values x for which $f(x)$ is defined.
$\text{ran } f$	The range of f: the set of values taken by $f(x)$ as x varies over the domain of f.
$f \oplus \{x \mapsto y\}$	A function that agrees with f except that x is mapped to y.
$\{x\} \trianglelefteq f$	A function like f, except that x is removed from its domain.

Logic:

$P \wedge Q$	P and Q: It is true if both P and Q are true.
$P \Rightarrow Q$	P implies Q: It is true if either Q is true or P is false.
$\theta S' = \theta S$	No components of schema S change in an operation.

Block Handler Schema

BlockHandler

$$used, free : \mathbb{P}\ BLOCKS$$

$$BlockQueue : \text{seq}\ \mathbb{P}\ BLOCKS$$

$$used \cap free = \varnothing \land$$

$$used \cup free = AllBlocks \land$$

$$\forall i : \text{dom}\ BlockQueue \bullet BlockQueue\ i \subseteq \text{used} \land$$

$$\forall i, j : \text{dom}\ BlockQueue \bullet i \neq j \Rightarrow$$

$$BlockQueue\ i \cap BlockQueue\ j = \varnothing$$

Block Handler Schema

- Whenever a schema representing the data invariant and state is used in another schema it is preceded by the Δ symbol

RemoveBlock

$\Delta BlockHandler$

$$\#BlockQueue > 0,$$
$$used' = used \setminus head\ BlockQueue \land$$
$$free' = free \cup head\ BlockQueue \land$$
$$BlockQueue' = tail\ BlockQueue$$

Block Handler Schema

- Adds a collection of blocks to the end of queue.
- Ablocks? –which acts as an input parameter, they are not part of the state.

AddBlock

$\Delta BlockHandler$

$Ablocks? : \mathrm{BLOCKS}$

$Ablocks? \subseteq used$

$BlockQueue' = BlockQueue \frown \langle Ablocks? \rangle$

$used' = used \land$

$free' = free$