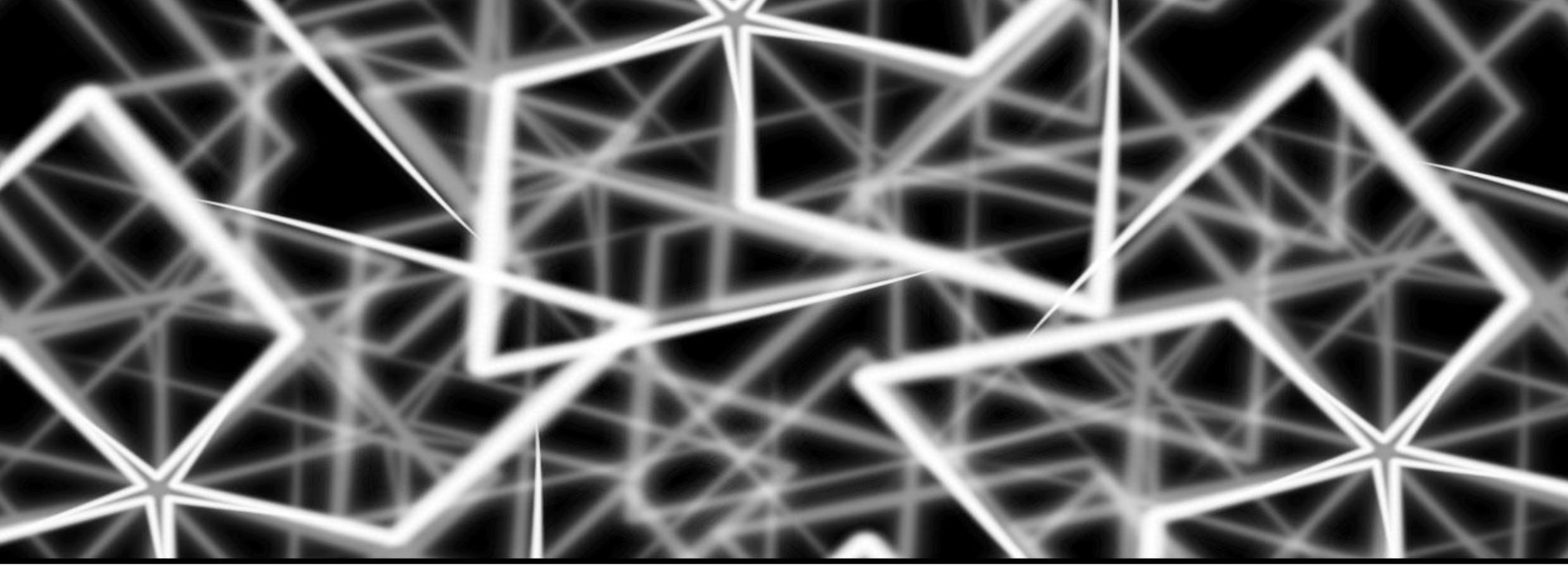


Lecture # 07

Z Specification

Model Based – Z Language

- Z is a *model-based notation*.
- In Z you usually model a system by representing its *state* -- a collection of *state variables* and their values -- and some *operations* that can change its state.
- A model that is characterized by the operations it describes is called an *abstract data type* (ADT).
- Z is also a natural fit to object-oriented programming.
- Z state variables are like instance variables and the operations are like methods;

Model Based – Z Language

- Z is just a notation, it is not a method;
- In general, Z specifications cannot be interpreted or compiled into a running program (or prototype or simulation). *Z is not a programming language.*

Z Language - Schema

Z is a way of decomposing a specification into small pieces called schema. In Z, schemas are used to describe both static and dynamic aspects of a system.

The static aspects include: State Space, Invariants

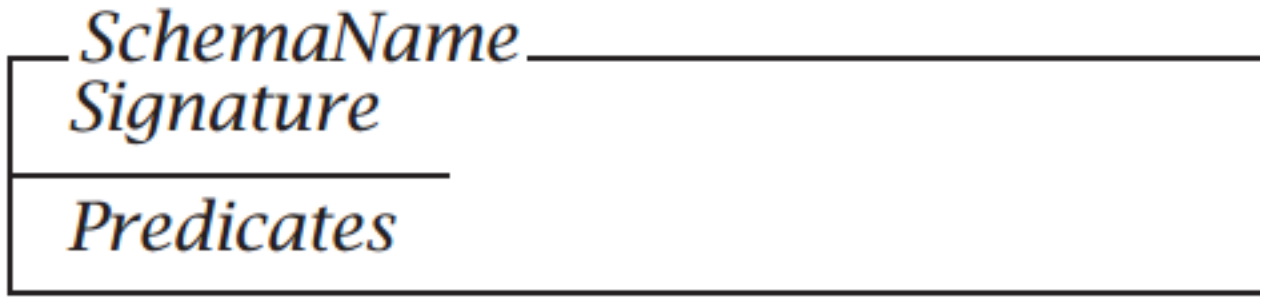
- the states it can occupy;
- the invariant relationships that are maintained as the system moves from state to state.

The dynamic aspects include: Operations, State Changes

- represent the triggers for state changes
- the operations that are possible;
- the relationship between their inputs and outputs;
- the changes of state that happen

Z Schema

- The building-block for structuring specifications
- Graphical notation



- An alternative linear notation

$$SchemaName \mathrel{\widehat=} [Signature \mid Predicates]$$

Z Schema

- Signature introduces variables and assigns them set theoretic types.

- Predicates include expressions that relate the elements of signature:
  - Viewed in terms of invariants, pre- and post-conditions
  - ANDed by default; order is irrelevant

Z Notation

- Identifiers may be composed of upper and lower case letters, digits, and the underscore character; must begin with a letter
- Identifiers may have suffixes:
  - **?** means an input variable
  - **!** means an output variable
  - **′** means a new value (i.e., the after-operation value)
- Schema identifiers may have prefixes:
  - Δ means the state has changed
  - Ξ means no change in the state

SCHEMA FORMAT IN Z

- Set definitions
  - For example:-
    - STUDENT :: = set of students
    - MESSAGE ::= ok|error|not_found
- State Space Schema

System_Name

Variable declarations
(includes declarations of system variables and partial function variables e.g., x:N. means a variable x declared from set of natural numbers. Here, x is a system variable.
A second example: rented: CAR ⇸ CUSTOMER)

Invariants
(one or more than one conditions imposed on system variables that must not be violated)

SCHEMA FORMAT IN Z

INITITAL STATE SCHEMA

Schema_Name
System_Name
Initialization of state variables

OPERATIONAL SCHEMA – UPDATE OPERATIONS

Schema_Name
Δ System_Name **Local variable declarations (if any)**
Pre-Conditions => Post-Conditions

SCHEMA FORMAT IN Z

OPERATIONAL SCHEMA – READ/ ERROR OPERATIONS

Schema_Name

Ξ System_Name
Local variable declarations (if any)

Pre-Conditions (if any) => Post-Conditions
(NOTE: It is not necessary for all the read operations to have post conditions defined. For example, a query may return an empty set (defined in power set))