Subject Name: Information System

Unit No:02 Unit Name: Access Control Models

Faculty: Mrs. Bhavana Alte

Mr. Prathmesh Gunjgur

Unit No: 2

Unit Name: Access Control Models

Access Control Algebra

Access Control Algebra

- **Access Control Algebra** is a formal approach to modeling and managing access control policies. It involves using **logical operations** and **set theory** to combine and define access control rules, policies, and permissions.
- The goal is to provide a **structured** and **mathematical** way to combine different access control decisions to ensure **secure access** to resources.
- Access control algebra allows for the **combination of multiple policies** in a logical and consistent way to define how permissions are granted or denied.

Key Concepts in Access Control Algebra:

•**Permissions**: These are the actions users can perform on resources (e.g., **read**, **write**, **execute**).

•**Subjects**: The entities trying to access the resources (e.g., **users**, **processes**, **devices**).

•**Objects**: The resources to be protected (e.g., **files**, **databases**, **documents**).

•**Access Control Policies**: Rules that define what permissions are granted to subjects for accessing objects (e.g., **who can access what resources and how**).

•**Logical Operations**: Set-based operations such as **union**, **intersection**, and **complement** are used to combine access control policies.

Basic Operations in Access Control Algebra:

- Access Control Algebra uses logical set operations to manipulate access policies.
- The key operations include:
- **Union (OR):** Combining two policies to grant access if either of the policies allow it. This operation means that a subject can access a resource if at least one policy allows the access.
- **Symbol:** $A \cup B$
- **A∪BMeaning**: Either policy A or policy B (or both) allow the access.
- Example: A user can read a file if they are either a Manager or have read permission granted by the resource owner.

Basic Operations in Access Control Algebra….:

- **Intersection (AND)**: Combining two policies to grant access only if **both policies** allow it. This operation means that a subject can access a resource only if **both policies** allow the access.
- **Symbol**: A∩B
- **Meaning**: Access is granted only if both policies A and B allow the access.
- **Example**: A user can read a document only if they are part of the **Admin role** and the document's **sensitive level** is **public**.

- **Complement (NOT)**: This operation denies access if a policy **does not allow** it.

•It represents the negation of a policy.

•**Symbol**: $\overline{A}$

•**Meaning**: Access is denied if policy $\overline{A}$ does not allow access.

•**Example**: A user cannot access a resource if they **do not** have the **Admin** role.

- **Difference (Subtract)**: This operation allows access to a resource by removing permissions granted by one policy from the permissions granted by another policy.

•**Symbol**: A−B

•**Meaning**: Access is granted based on policy A, but **denied** for permissions in policy B.

•**Example**: A user has **read** permission (policy A) but is denied the ability to **edit** the document (policy B).

Access Control Algebra with Example:

- **Scenario:**
- **User A** is a **Manager** and wants access to a **Sensitive File**.
- The **Sensitive File** has the following access policies:
  - Policy 1: Only **Managers** can access it (Role-based access).
  - Policy 2: Only users with a **Top Secret** clearance can read the file (Clearance-based access).
  - Policy 3: The file should only be accessed during **business hours** (Time-based access).

Step 1: Defining the Policies

- **Policy 1** (Role-based access):
  - If the user is a **Manager**, they are granted **read** permission.
  - Let's denote this as:
    - P1=Role(Manager)
- **Policy 2** (Clearance-based access):
  - If the user has **Top Secret** clearance, they are granted **read** permission.
  - Let's denote this as:
    - P2=Clearance(Top Secret)
- **Policy 3** (Time-based access):
  - The file can only be accessed during **business hours** (9 AM to 5 PM).
  - Let's denote this as:
    - P3=Time(Business Hours)

Step 2: Combining Policies Using Algebra

- Combine these policies using **set operations** to determine if **User A** can access the **Sensitive File**.
- **Union of Policy 1 and Policy 2** (Role OR Clearance):
  - User A is a **Manager**, so they meet Policy 1. User A also has **Top Secret** clearance, so they meet Policy 2.
  - Combined policies: $P1 \cup P2$
  - This means **User A** can access the file if they are a **Manager** or have **Top Secret clearance**.

- **Intersection with Policy 3** (Time AND):
  - The access is allowed only during **business hours**. So, the combined policy from Step 1 will be further restricted by Policy 3.
  - Final access decision: $(P1 \cup P2) \cap P3$
  - This means **User A** can access the file only if they are either a **Manager** or have **Top Secret clearance** AND it is **business hours**.

Final Access Decision

- **User A** is a **Manager** with **Top Secret clearance**, and it's **business hours**.
- According to the combined policy (P1∪P2)∩P3, User A meets the access criteria, so they are **granted access** to the **Sensitive File**.

Benefits of Access Control Algebra

- **Formal and Precise**: Access Control Algebra uses **logical operations** and **set theory**, which provides a formal and precise way to define access control policies.
- **Flexibility**: By combining policies using union, intersection, and complement, you can easily adapt access control systems to complex scenarios.
- **Scalable**: It's easy to scale up by adding more policies and combining them using algebraic operations.

Use Cases for Access Control Algebra:

- **Complex Organizational Systems**: In large organizations with multiple policies, Access Control Algebra allows for **fine-grained access control** by combining various conditions (e.g., roles, clearance levels, time restrictions).

- **Security for Sensitive Data**: For systems that handle sensitive or classified data, Access Control Algebra allows for **dynamic policies** based on different conditions.

- **Contextual Access Control**: For systems where access should depend on factors like the **time of day**, **user role**, and **resource sensitivity**, Access Control Algebra provides a flexible framework.

Thank You