2.1. Protection State

The **state** of a system is the collection of the current values of all memory locations, all secondary storage, and all registers and other components of the system. The subset of this collection that deals with protection is the **protection state** of the system. An **access control matrix** is one tool that can describe the current protection state.

Consider the set of possible protection states **P**. Some subset **Q** of **P** consists of exactly those states in which the system is authorized to reside. So, whenever the system state is in **Q**, the system is secure. When the current state is in **P** – **Q**[1] , the system is not secure. Our interest in representing the state is to characterize those states in **Q**, and our interest in enforcing security is to ensure that the system state is always an element of **Q**. Characterizing the states in **Q** is the function of a **security policy**; preventing the system from entering a state in **P** – **Q** is the function of a **security mechanism**. Recall from Definition 1–3 that a mechanism that enforces this restriction is **precise.**

[1] The notation **P** – **Q** means all elements of set **P** not in set **Q**.

The **access control matrix model** is the most precise model used to describe a protection state. It characterizes the rights of each **subject** (active entity, such as a process) with respect to every other entity. The description of elements of **A** form a **specification** against which the current state can be compared. Specifications take many forms, and different specification languages have been created to describe the characteristics of allowable states.

As the system changes, the protection state changes. When a command changes the state of the system, a **state transition** occurs. Very often, constraints on the set of allowed states use these transitions inductively; a set of authorized states is defined, and then a set of operations is allowed on the elements of that set. The result of transforming an authorized state with an operation allowed in that state is an authorized state. By induction, the system will always be in an authorized state. Hence, both states and state transitions are often constrained.

In practice, **any** operation on a real system causes multiple state transitions; the reading, loading, altering, and execution of any datum or instruction causes a transition. We are concerned only with those state transitions that affect the protection state of the system, so only transitions that alter the actions a subject is authorized to take are relevant. For example, a program that changes a variable to 0 does not (usually) alter the protection state. However, if the variable altered is one that affects the privileges of a process, then the program does alter the protection state and needs to be accounted for in the set of transitions.