$= \bigcap_{i \in I}$

**Definition 0.1.** A requirement, $\rho$, is said to be *separable* if and only if it can be written as $\rho(x) = \prod_{i \in I} \rho_i(x)$, where $\forall i \in I$, $\rho_i(x)$, is a requirement with required set $X$.

**Definition 0.2.** A *requirement operator* is a mapping, $\mathbf{Req} : \mathbf{Set} \to \mathbf{Set}$, such that,

$$\mathbf{Req}_\rho(X) := \{ \; x \in X \mid \rho(x) = 1 \; \}.$$

**Proposition 0.1.** Given requirement $\rho$, if $\rho = \rho_1 \cdot \rho_2$, where $\rho_1$ and $\rho_2$ are requirements with required set X, then

$$\mathbf{Req}_\rho(X) = \mathbf{Req}_{\rho_1}(X) \cdot \mathbf{Req}_{\rho_2}(X)$$

*Proof.* The proof of this proposition is very straightforward. Let $X$ be a set and $\rho$ be a requirement with required set $X$. Then,

$$\begin{aligned}
\mathbf{Req}_\rho(X) &= \{ \; x \in X \mid \rho(x) = 1 \; \} && (1)\\
&= \{ \; x \in X \mid \rho_1(x)\rho_2(x) = 1 \; \} && (2)\\
&= \{ \; x \in X \mid \rho_1(x) = 1 and \rho_2(x) = 1 \; \} && (3)\\
&= \{ \; x \in X \mid \rho_1(x) = 1 \; \} \cdot \{ \; x \in X \mid \rho_2(x) = 1 \; \} && (4)\\
&= \mathbf{Req}_{\rho_1}(X) \cdot \mathbf{Req}_{\rho_2}(X) && (5)
\end{aligned}$$

$\square$

*Remark.* The binary operation between two requirements is the same as the symbol, $\wedge$, used in boolean algebra to represent the join, *and*, between two boolean statements.

**Proposition 0.2.** Given requirement $\rho = \prod_{i \in I} \rho_i$, where $\rho_i$ are requirements all with required set X,

$$\mathbf{Req}_\rho(X) = \bigcap_{i \in I} \mathbf{Req}_{\rho_i}(X)$$

## 0.1 Specification

**Definition 0.3.** A set, $X$, is said to be inspectable if and only if, there exists a function, $\psi : X \to \prod_{i \in I} X_i$, where $X \neq X_i, \forall i \in I$. This function is referred to as an *inspection function* of $X$.

*Remark.* The inspection function may also be expressed as, $\psi(x) = (\psi_i(x))_{i \in I}$.

**Definition 0.4.** A *specification* is a requirement, $\phi : X \to \mathbb{B}$, such that the following conditions hold:

## 2 Consumer

**Definition 2.1.** A *consumer*, $c$, is a triple $(k, \phi, t)$, where $k \in \mathbb{N}$, $\phi$ is a specification, and $t \in \mathbb{R}^+$. The consumer $C$ is the subset of consumer $c$ which defined such that:

$$c = (\phi, \sigma_m)$$

$$C = 2^\phi * [0, \sigma_m)$$

**Definition 2.2.** Let *service* $v$ be the available resource can be provide which related with a subset of resource $R'$ and a boolean variable $\beta$ can be defined such that:

$$v := (R', \beta)$$

$$V = 2^R * ß$$

# Part II
# Scheduling Process

## 3 Process Decomposition

**Definition 3.1.** let $a$ represent the arriving process $a = (c, t_a)$. Then the assignment action A defined as:

$$A = C * [0, \infty)$$

**Definition 3.2.** let $g$ reprosent the assigment process, $g = (c, \sigma_w, v)$. Then the assignment action G can be defined as:

$$G = C * [0, \sigma_m) * V$$

**Definition 3.3.** let $d$ represent the departure process $d = (c, t_d)$, we also defined $\gamma(c) = min\sigma_r, \sigma_m$. Then the departure action D defined as:

$$D = C * [t_a, t_a + \sigma_w + \gamma(c)]$$

# Part III
# Schedule Learning

## 4 Policy Routing

### 4.1 Policy Graph

**Definition 4.1.** Let $G = (V, E, src, tgt)$ be a graph. A *path of length* $n$ in $G$, denoted $p \in \mathbf{Path}_G^{(n)}$, is a head-to-tail sequence:

$$p = (v_1 \overset{a_1}{\to} v_2 \overset{a_2}{\to} \cdots \overset{a_{n-2}}{\to} v_{n-1} \overset{a_{n-1}}{\to} v_n)$$

The *set of all paths on* $G$ is defined such that:

$$\mathbf{Path}_G := \bigsqcup_{n \in \mathbb{N}} \mathbf{Path}_G^{(n)}$$

**Definition 4.2.** let $\mathbf{Path}_G(v)$ be the all path from v back to v.

The *set of all path on* $G$ is defined such that:

$$\mathbf{Path}_G(s,t) = \mathbf{Path}_G(s,v) \sqcup \mathbf{Cycle}_G(v) \sqcup \mathbf{Path}_G(v,t).$$

The $\mathbf{Cycle}_G$ is defined as that:

$$\mathbf{Cycle}_G(v) = \mathbf{Path}_G(v,v)$$