Part I
Requirement Algebra

1 Introduction

In order to build a scheduling algorithm that is capable of dealing with heterogeneous resources and consumers, we must learn how to build a scheduling process that is decomposable. With a decomposable scheduling process we will be able to impose a solution on each prime component of the process and therefore solve the larger process. The reinforcement solution will analogously follow the decomposition of the problem, for it too is something that can be decomposed.

The reinforcement learning algorithm decomposition depends on the ability for the system itself to be highly seperable. Therefore that is where the solution will first be addressed. The system depends on resources and consumers who want to use resources for a certain period of time. In order for the consumer to tell the system what it wants, it must enter a resource specification that explains the different requirements that must be met in order for them to have the acceptable product that they wanted. This is the place where the system will undergo the first decomposition, the decomposition of consumers. To properly do this we have constructed a simple algebra that can aid in this endeavor.

2 Requirement

Definition 2.1. A *requirement* is a function, $\rho : X \to \mathbb{B}$, where X is a set, and $\mathbb{B} = \{0, 1\}$.

The set, X, is referred to as the required set of the requirement ρ.

Proposition 2.1 (Properties)**.** (1)

Definition 2.2. A requirement, ρ, is said to be *seperable* 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 2.3. 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 2.2. Given requirement ρ, if $\rho = \rho_1 \cdot \rho_2$, where ρ_1 and ρ_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 ρ be a requirement with required set X. Then,

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

□

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

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

2.1 Specification

Definition 2.4. 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 2.5. A *specification* is a requirement, $\phi : X \to \mathbb{B}$, such that the following conditions hold:

1. The required set, X, is inspectable.
2. There exists a requirement, $\varphi : \prod_{i \in I} X_i \to \mathbb{B}$, such that $\rho = \varphi \circ \psi$, where ψ is an inspection function of X, with $X \neq X_i, \forall i \in I$. The requirement, φ, is referred to as an *acceptance requirement* of X.

Theorem. *Given a separable specification, ϕ, with an acceptance requirement that is mutually independent, $\varphi(x_i)_{i \in I}$, then following isomorphism holds:*

$$\mathbf{Req}_{\phi}(X) = \prod_{i \in I} \mathbf{Req}_{\varphi_i}(X_i)$$

Part II
Consumers, Producers, and Resources

3 Resources

Definition 3.1. Let $U = (R, \mu)$

4 Consumer

Definition 4.1. A *consumer*, c, is a triple (k, ϕ, t), where $k \in \mathbb{N}$, ϕ is a specification, and $t \in \mathbb{R}^+$. The comsumer C is the subset of comsumer c which defined such that:

$$c = (\phi, \sigma_m)$$
$$C = 2^{\phi} * [0, \sigma_m)$$

Definition 4.2. Let *service* v be the available resource can be provide which related with a subset of resource R' and a boolean variable β can be defined such that:

$$v := (R', \beta)$$
$$V = 2^{R} * ß$$

Part III
Scheduling Process

5 Process Decomposition

Definition 5.1. let a represent the arriving process $a = (c, t_a)$. Then the assignment action A defined as:

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

Definition 5.2. 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 5.3. 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 IV
Schedule Learning

6 Policy Routing

6.1 Policy Graph

Definition 6.1. 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 6.2. 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:

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