$\odot$ as the summing operator. That is, $y_j = x_1 \odot x_2 \odot \cdots \odot x_j$ for $1 \leq j \leq n$. Thus, if the set $\mathcal{A}$ is $\mathbb{N}$, the natural numbers, and $\odot$ is the integer addition operator $+$, then $\mathcal{P}_+^{(n)}$ on the input $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ produces the output $\boldsymbol{y}$, where $y_1 = x_1$, $y_2 = x_1 + x_2$, $y_3 = x_1 + x_2 + x_3$, $\ldots$, $y_n = x_1 + x_2 + \cdots + x_n$. For example, shown below is the prefix function on a 6-vector of integers under integer addition.

$$\boldsymbol{x} = (2, 1, 3, 7, 5, 1)$$
$$\mathcal{P}_+^{(6)}(\boldsymbol{x}) = (2, 3, 6, 13, 18, 19)$$

A prefix function is defined only for operators $\odot$ that are associative over the set $\mathcal{A}$. An **operator over $\mathcal{A}$ is associative** if a) for all a and b in $\mathcal{A}$, $a \odot b$ is in $\mathcal{A}$, and b) for all a, b, and c in $\mathcal{A}$, $(a \odot b) \odot c = a \odot (b \odot c)$—that is, if all groupings of terms in a sum with the operator $\odot$ have the same value. A pair $(\mathcal{A}, \odot)$ in which $\odot$ is associative is called a **semigroup**. Three semigroups on which a prefix function can be defined are

- $(\mathbb{N}, +)$ where $\mathbb{N}$ are the natural numbers and $+$ is integer addition.
- $(\{0,1\}^*, \cdot)$ where $\{0,1\}^*$ is the set of binary strings and $\cdot$ is string concatenation.
- $(\mathcal{A}, \odot_{\text{copy}})$ where $\mathcal{A}$ is a set and $\odot_{\text{copy}}$ is defined by $a \odot_{\text{copy}} b = a$.

It is easy to show that the concatenation operator $\cdot$ on $\{0,1\}^*$ and $\odot_{\text{copy}}$ on a set $\mathcal{A}$ are associative. (See Problem 2.20.) Another important semigroup is the set of matrices under matrix multiplication (see Theorem 6.2.1).

Summarizing, if $(\mathcal{A}, \odot)$ is a semigroup, the prefix function $\mathcal{P}_\odot^{(n)} : \mathcal{A}^n \mapsto \mathcal{A}^n$ on input $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ produces as output $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$, where $y_j = x_1 \odot x_2 \odot \cdots \odot x_j$ for $1 \leq j \leq n$.

Load balancing on a parallel machine is an important application of prefix computation. A simple example of load balancing is the following: We assume that p processors, numbered from 0 to $p-1$, are running processes in parallel. We also assume that processes are born and die, resulting in a possible imbalance in the number of processes active on processors. Since it is desirable that all processors be running the same number of processes, processes are periodically redistributed among processors to balance the load. To rebalance the load, a) processors are given a linear order, and b) each process is assigned a Boolean variable with value 1 if it is alive and 0 otherwise. Each processor computes its number of living processes, n_i. A prefix computation is then done on these values using the linear order among processors. This computation provides the jth processor with the sum $n_j + n_{j-1} + \cdots + n_1$ which it uses to give each of its living processes a unique index. The sum $n = n_p + \cdots + n_1$ is then broadcast to all processors. When the processors are in balance all have $\lceil n/p \rceil$ processes except possibly one that has fewer processes. Assigning the sth process to processor $(s \bmod p)$ insures that the load is balanced.

Another important type of prefix computation is the **segmented prefix computation**. In this case two n-vectors are given, a **value vector $\boldsymbol{x}$** and a **flag vector $\boldsymbol{\phi}$**. The value of the ith entry y_i in the result vector $\boldsymbol{y}$ is x_i if ϕ_i is 1 and otherwise is the associative combination with $\odot$ of x_i and the values between it and the first value x_j to the left of x_i for which the flag $\phi_j = 1$. The first bit of ϕ is always 1. An example of a segmented prefix computation is shown below for integer values and integer addition as the associative operation:

$$\boldsymbol{x} = (2, 1, 3, 7, 5, 1)$$