Concept definitions from *Elements of Programming*

Alexander Stepanov   Paul McJones

April 28, 2011

Introduction

This is a summary of the concept definitions from *Elements of Programming*, published by Addison-Wesley Professional in June 2009. For more information, see `www.elementsofprogramming.com`.

Chapter 1: Foundations

$\mathit{Regular}(\mathsf{T}) \triangleq$

T's computational basis includes equality, assignment, destructor, default constructor, copy constructor, total ordering (or default total ordering) and underlying type.

$\mathit{FunctionalProcedure}(\mathsf{F}) \triangleq$

F is a *regular* procedure defined on regular types: replacing its inputs with equal objects results in equal output objects.

$\mathit{UnaryFunction}(\mathsf{F}) \triangleq$

$\quad \mathit{FunctionalProcedure}(\mathsf{F})$
$\wedge\ \mathsf{Arity}(\mathsf{F}) = 1$
$\wedge\ \mathsf{Domain} : \mathit{UnaryFunction} \to \mathit{Regular}$
$\quad\quad \mathsf{F} \mapsto \mathsf{InputType}(\mathsf{F}, 0)$

$\mathit{HomogeneousFunction}(\mathsf{F}) \triangleq$

$\quad \mathit{FunctionalProcedure}(\mathsf{F})$
$\wedge\ \mathsf{Arity}(\mathsf{F}) > 0$
$\wedge\ (\forall \mathsf{i}, \mathsf{j} \in \mathbb{N})(\mathsf{i}, \mathsf{j} < \mathsf{Arity}(\mathsf{F})) \Rightarrow (\mathsf{InputType}(\mathsf{F}, \mathsf{i}) = \mathsf{InputType}(\mathsf{F}, \mathsf{j}))$
$\wedge\ \mathsf{Domain} : \mathit{HomogeneousFunction} \to \mathit{Regular}$
$\quad\quad \mathsf{F} \mapsto \mathsf{InputType}(\mathsf{F}, 0)$

property$(\mathsf{F} : \mathit{UnaryFunction})$
regular_unary_function : F

$$\begin{aligned}&\mathsf{f} \mapsto (\forall \mathsf{f}' \in \mathsf{F})(\forall x, x' \in \mathsf{Domain}(\mathsf{F}))\\ &\qquad (\mathsf{f} = \mathsf{f}' \wedge x = x') \Rightarrow (\mathsf{f}(x) = \mathsf{f}'(x'))\end{aligned}$$

Chapter 2: Transformations and Their Orbits

$$\begin{aligned}&\mathit{Predicate}(\mathsf{P}) \triangleq\\ &\qquad \mathit{FunctionalProcedure}(\mathsf{P})\\ &\quad \wedge\ \mathsf{Codomain}(\mathsf{P}) = \mathsf{bool}\end{aligned}$$

$$\begin{aligned}&\mathit{HomogeneousPredicate}(\mathsf{P}) \triangleq\\ &\qquad \mathit{Predicate}(\mathsf{P})\\ &\quad \wedge\ \mathit{HomogeneousFunction}(\mathsf{P})\end{aligned}$$

$$\begin{aligned}&\mathit{UnaryPredicate}(\mathsf{P}) \triangleq\\ &\qquad \mathit{Predicate}(\mathsf{P})\\ &\quad \wedge\ \mathit{UnaryFunction}(\mathsf{P})\end{aligned}$$

$$\begin{aligned}&\mathit{Operation}(\mathsf{Op}) \triangleq\\ &\qquad \mathit{HomogeneousFunction}(\mathsf{Op})\\ &\quad \wedge\ \mathsf{Codomain}(\mathsf{Op}) = \mathsf{Domain}(\mathsf{Op})\end{aligned}$$

$$\begin{aligned}&\mathit{Transformation}(\mathsf{F}) \triangleq\\ &\qquad \mathit{Operation}(\mathsf{F})\\ &\quad \wedge\ \mathit{UnaryFunction}(\mathsf{F})\\ &\quad \wedge\ \mathsf{DistanceType} : \mathit{Transformation} \to \mathit{Integer}\end{aligned}$$

Chapter 3: Associative Operations

$$\begin{aligned}&\mathit{BinaryOperation}(\mathsf{Op}) \triangleq\\ &\qquad \mathit{Operation}(\mathsf{Op})\\ &\quad \wedge\ \mathsf{Arity}(\mathsf{Op}) = 2\end{aligned}$$

$$\begin{aligned}&\textbf{property}(\mathsf{Op} : \mathit{BinaryOperation})\\ &\mathsf{associative} : \mathsf{Op}\\ &\quad op \mapsto (\forall a, b, c \in \mathsf{Domain}(op))\ op(op(a, b), c) = op(a, op(b, c))\end{aligned}$$

$$\begin{aligned}&\mathit{Integer}(I) \triangleq\\ &\qquad \mathsf{successor} : I \to I\\ &\qquad\quad n \mapsto n + 1\\ &\quad \wedge\ \mathsf{predecessor} : I \to I\\ &\qquad\quad n \mapsto n - 1\\ &\quad \wedge\ \mathsf{twice} : I \to I\\ &\qquad\quad n \mapsto n + n\\ &\quad \wedge\ \mathsf{half\_nonnegative} : I \to I\\ &\qquad\quad n \mapsto \lfloor n/2 \rfloor, \text{ where } n \geqslant 0\end{aligned}$$

$\wedge$ binary_scale_down_nonnegative : $I \times I \to I$
$(n, k) \mapsto \lfloor n/2^k \rfloor$, where $n, k \geqslant 0$
$\wedge$ binary_scale_up_nonnegative : $I \times I \to I$
$(n, k) \mapsto 2^k n$, where $n, k \geqslant 0$
$\wedge$ positive : $I \to$ bool
$n \mapsto n > 0$
$\wedge$ negative : $I \to$ bool
$n \mapsto n < 0$
$\wedge$ zero : $I \to$ bool
$n \mapsto n = 0$
$\wedge$ one : $I \to$ bool
$n \mapsto n = 1$
$\wedge$ even : $I \to$ bool
$n \mapsto (n \bmod 2) = 0$
$\wedge$ odd : $I \to$ bool
$n \mapsto (n \bmod 2) \neq 0$

Chapter 4: Linear Orderings

$\mathit{Relation}(\mathsf{Op}) \triangleq$
$\mathit{HomogeneousPredicate}(\mathsf{Op})$
$\wedge$ $\mathsf{Arity}(\mathsf{Op}) = 2$

property(R : *Relation*)
transitive : R
$r \mapsto (\forall a, b, c \in \mathsf{Domain}(R))\ (r(a, b) \wedge r(b, c) \Rightarrow r(a, c))$

property(R : *Relation*)
strict : R
$r \mapsto (\forall a \in \mathsf{Domain}(R))\ \neg r(a, a)$

property(R : *Relation*)
reflexive : R
$r \mapsto (\forall a \in \mathsf{Domain}(R))\ r(a, a)$

property(R : *Relation*)
symmetric : R
$r \mapsto (\forall a, b \in \mathsf{Domain}(R))\ (r(a, b) \Rightarrow r(b, a))$

property(R : *Relation*)
asymmetric : R
$r \mapsto (\forall a, b \in \mathsf{Domain}(R))\ (r(a, b) \Rightarrow \neg r(b, a))$

property(R : *Relation*)
equivalence : R
$r \mapsto \mathsf{transitive}(r) \wedge \mathsf{reflexive}(r) \wedge \mathsf{symmetric}(r)$

$$\begin{array}{l}
\textbf{property}(\mathsf{F} : \mathit{UnaryFunction}, \mathsf{R} : \mathit{Relation}) \\
\quad \textbf{requires}(\mathsf{Domain}(\mathsf{F}) = \mathsf{Domain}(\mathsf{R})) \\
\mathsf{key_function} : \mathsf{F} \times \mathsf{R} \\
\quad (f, r) \mapsto (\forall a, b \in \mathsf{Domain}(\mathsf{F}))\, (r(a, b) \Leftrightarrow f(a) = f(b))
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{R} : \mathit{Relation}) \\
\mathsf{total_ordering} : \mathsf{R} \\
\quad r \mapsto \mathsf{transitive}(r) \wedge \\
\qquad (\forall a, b \in \mathsf{Domain}(\mathsf{R})) \text{ exactly one of the following holds:} \\
\qquad\qquad r(a, b), r(b, a), \text{ or } a = b
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{R} : \mathit{Relation}) \\
\mathsf{total_ordering} : \mathsf{R} \\
\quad r \mapsto \mathsf{transitive}(r) \wedge \\
\qquad (\forall a, b \in \mathsf{Domain}(\mathsf{R})) \text{ exactly one of the following holds:} \\
\qquad\qquad r(a, b), r(b, a), \text{ or } a = b
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{R} : \mathit{Relation}, \mathsf{E} : \mathit{Relation})\ \textbf{requires}(\mathsf{Domain}(\mathsf{R}) = \mathsf{Domain}(\mathsf{E})) \\
\mathsf{weak_ordering} : \mathsf{R} \\
\quad r \mapsto \mathsf{transitive}(r) \wedge (\exists e \in \mathsf{E})\, \mathsf{equivalence}(e) \wedge \\
\qquad\qquad (\forall a, b \in \mathsf{Domain}(\mathsf{R})) \text{ exactly one of the following holds:} \\
\qquad\qquad\qquad r(a, b), r(b, a), \text{ or } e(a, b)
\end{array}$$

$$\begin{array}{l}
\mathit{TotallyOrdered}(\mathsf{T}) \triangleq \\
\qquad \mathit{Regular}(\mathsf{T}) \\
\quad \wedge\ <: \mathsf{T} \times \mathsf{T} \to \mathsf{bool} \\
\quad \wedge\ \mathsf{total_ordering}(<)
\end{array}$$

Chapter 5: Ordered Algebraic Structures

$$\begin{array}{l}
\textbf{property}(\mathsf{T} : \mathit{Regular}, \mathsf{Op} : \mathit{BinaryOperation}) \\
\quad \textbf{requires}(\mathsf{T} = \mathsf{Domain}(\mathsf{Op})) \\
\mathsf{identity_element} : \mathsf{T} \times \mathsf{Op} \\
\quad (e, op) \mapsto (\forall a \in \mathsf{T})\, op(a, e) = op(e, a) = a
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{F} : \mathit{Transformation}, \mathsf{T} : \mathit{Regular}, \mathsf{Op} : \mathit{BinaryOperation}) \\
\quad \textbf{requires}(\mathsf{Domain}(\mathsf{F}) = \mathsf{T} = \mathsf{Domain}(\mathsf{Op})) \\
\mathsf{inverse_operation} : \mathsf{F} \times \mathsf{T} \times \mathsf{Op} \\
\quad (inv, e, op) \mapsto (\forall a \in \mathsf{T})\, op(a, inv(a)) = op(inv(a), a) = e
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{Op} : \mathit{BinaryOperation}) \\
\mathsf{commutative} : \mathsf{Op} \\
\quad op \mapsto (\forall a, b \in \mathsf{Domain}(\mathsf{Op}))\, op(a, b) = op(b, a)
\end{array}$$

$$\begin{array}{l}
\mathit{AdditiveSemigroup}(\mathsf{T}) \triangleq \\
\qquad \mathit{Regular}(\mathsf{T}) \\
\quad \wedge\ + : \mathsf{T} \times \mathsf{T} \to \mathsf{T}
\end{array}$$

$\wedge\ \mathsf{associative}(+)$
$\wedge\ \mathsf{commutative}(+)$

$\mathit{MultiplicativeSemigroup}(\mathsf{T}) \triangleq$
$\quad \mathit{Regular}(\mathsf{T})$
$\wedge\ \cdot : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\wedge\ \mathsf{associative}(\cdot)$

$\mathit{AdditiveMonoid}(\mathsf{T}) \triangleq$
$\quad \mathit{AdditiveSemigroup}(\mathsf{T})$
$\wedge\ 0 \in \mathsf{T}$
$\wedge\ \mathsf{identity\_element}(0, +)$

$\mathit{MultiplicativeMonoid}(\mathsf{T}) \triangleq$
$\quad \mathit{MultiplicativeSemigroup}(\mathsf{T})$
$\wedge\ 1 \in \mathsf{T}$
$\wedge\ \mathsf{identity\_element}(1, \cdot)$

$\mathit{AdditiveGroup}(\mathsf{T}) \triangleq$
$\quad \mathit{AdditiveMonoid}(\mathsf{T})$
$\wedge\ - : \mathsf{T} \to \mathsf{T}$
$\wedge\ \mathsf{inverse\_operation}(\text{unary } -, 0, +)$
$\wedge\ - : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\quad\quad (a, b) \mapsto a + (-b)$

$\mathit{MultiplicativeGroup}(\mathsf{T}) \triangleq$
$\quad \mathit{MultiplicativeMonoid}(\mathsf{T})$
$\wedge\ \mathsf{multiplicative\_inverse} : \mathsf{T} \to \mathsf{T}$
$\wedge\ \mathsf{inverse\_operation}(\mathsf{multiplicative\_inverse}, 1, \cdot)$
$\wedge\ / : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\quad\quad (a, b) \mapsto a \cdot \mathsf{multiplicative\_inverse}(b)$

$\mathit{Semiring}(\mathsf{T}) \triangleq$
$\quad \mathit{AdditiveMonoid}(\mathsf{T})$
$\wedge\ \mathit{MultiplicativeMonoid}(\mathsf{T})$
$\wedge\ 0 \neq 1$
$\wedge\ (\forall a \in \mathsf{T})\, 0 \cdot a = a \cdot 0 = 0$
$\wedge\ (\forall a, b, c \in \mathsf{T})$
$\quad\quad\quad a \cdot (b + c) = a \cdot b + a \cdot c$
$\quad\quad \wedge\ (b + c) \cdot a = b \cdot a + c \cdot a$

$\mathit{CommutativeSemiring}(\mathsf{T}) \triangleq$
$\quad \mathit{Semiring}(\mathsf{T})$
$\wedge\ \mathsf{commutative}(\cdot)$

$\mathit{Ring}(\mathsf{T}) \triangleq$
$\quad \mathit{AdditiveGroup}(\mathsf{T})$
$\wedge\ \mathit{Semiring}(\mathsf{T})$

$\mathit{CommutativeRing}(\mathsf{T}) \triangleq$

$$\begin{aligned} & \quad \mathit{AdditiveGroup}(\mathsf{T}) \\ \wedge & \quad \mathit{CommutativeSemiring}(\mathsf{T}) \end{aligned}$$

$\mathit{Semimodule}(\mathsf{T}, \mathsf{S}) \triangleq$

$$\begin{aligned} & \quad \mathit{AdditiveMonoid}(\mathsf{T}) \\ \wedge & \quad \mathit{CommutativeSemiring}(\mathsf{S}) \\ \wedge & \quad \cdot : \mathsf{S} \times \mathsf{T} \to \mathsf{T} \\ \wedge & \quad (\forall \alpha, \beta \in \mathsf{S})(\forall a, b \in \mathsf{T}) \end{aligned}$$

$$\begin{aligned} \alpha \cdot (\beta \cdot a) &= (\alpha \cdot \beta) \cdot a \\ (\alpha + \beta) \cdot a &= \alpha \cdot a + \beta \cdot a \\ \alpha \cdot (a + b) &= \alpha \cdot a + \alpha \cdot b \\ 1 \cdot a &= a \end{aligned}$$

$\mathit{Module}(\mathsf{T}, \mathsf{S}) \triangleq$

$$\begin{aligned} & \quad \mathit{Semimodule}(\mathsf{T}, \mathsf{S}) \\ \wedge & \quad \mathit{AdditiveGroup}(\mathsf{T}) \\ \wedge & \quad \mathit{Ring}(\mathsf{S}) \end{aligned}$$

$\mathit{OrderedAdditiveSemigroup}(\mathsf{T}) \triangleq$

$$\begin{aligned} & \quad \mathit{AdditiveSemigroup}(\mathsf{T}) \\ \wedge & \quad \mathit{TotallyOrdered}(\mathsf{T}) \\ \wedge & \quad (\forall a, b, c \in \mathsf{T})\; a < b \Rightarrow a + c < b + c \end{aligned}$$

$\mathit{OrderedAdditiveMonoid}(\mathsf{T}) \triangleq$

$$\begin{aligned} & \quad \mathit{OrderedAdditiveSemigroup}(\mathsf{T}) \\ \wedge & \quad \mathit{AdditiveMonoid}(\mathsf{T}) \end{aligned}$$

$\mathit{OrderedAdditiveGroup}(\mathsf{T}) \triangleq$

$$\begin{aligned} & \quad \mathit{OrderedAdditiveMonoid}(\mathsf{T}) \\ \wedge & \quad \mathit{AdditiveGroup}(\mathsf{T}) \end{aligned}$$

$\mathit{CancellableMonoid}(\mathsf{T}) \triangleq$

$$\begin{aligned} & \quad \mathit{OrderedAdditiveMonoid}(\mathsf{T}) \\ \wedge & \quad - : \mathsf{T} \times \mathsf{T} \to \mathsf{T} \\ \wedge & \quad (\forall a, b \in \mathsf{T})\; b \leqslant a \Rightarrow a - b \text{ is defined} \wedge (a - b) + b = a \end{aligned}$$

```
template<typename T>
    requires(CancellableMonoid(T))
T slow_remainder(T a, T b)
{
    // Precondition: a ⩾ 0 ∧ b > 0
    while (b <= a) a = a - b;
    return a;
}
```

$\mathit{ArchimedeanMonoid}(\mathsf{T}) \triangleq$

$$\begin{aligned} & \quad \mathit{CancellableMonoid}(\mathsf{T}) \\ \wedge & \quad (\forall a, b \in \mathsf{T})\; (a \geqslant 0 \wedge b > 0) \Rightarrow \mathsf{slow\_remainder}(a, b) \text{ terminates} \\ \wedge & \quad \mathsf{QuotientType} : \mathit{ArchimedeanMonoid} \to \mathit{Integer} \end{aligned}$$

$\mathit{HalvableMonoid}(\mathsf{T}) \triangleq$
$\quad \mathit{ArchimedeanMonoid}(\mathsf{T})$
$\wedge\ \mathsf{half} : \mathsf{T} \to \mathsf{T}$
$\wedge\ (\forall a, b \in \mathsf{T})\,(b > 0 \wedge a = b + b) \Rightarrow \mathsf{half}(a) = b$

```
template<typename T>
    requires(ArchimedeanMonoid(T))
T subtractive_gcd_nonzero(T a, T b)
{
    // Precondition: a > 0 ∧ b > 0
    while (true) {
        if (b < a)      a = a - b;
        else if (a < b) b = b - a;
        else            return a;
    }
}
```

$\mathit{EuclideanMonoid}(\mathsf{T}) \triangleq$
$\quad \mathit{ArchimedeanMonoid}(\mathsf{T})$
$\wedge\ (\forall a, b \in \mathsf{T})\,(a > 0 \wedge b > 0) \Rightarrow \mathsf{subtractive\_gcd\_nonzero}(a, b)$ terminates

$\mathit{EuclideanSemiring}(\mathsf{T}) \triangleq$
$\quad \mathit{CommutativeSemiring}(\mathsf{T})$
$\wedge\ \mathsf{NormType} : \mathit{EuclideanSemiring} \to \mathit{Integer}$
$\wedge\ \mathsf{w} : \mathsf{T} \to \mathsf{NormType}(\mathsf{T})$
$\wedge\ (\forall a \in \mathsf{T})\, \mathsf{w}(a) \geqslant 0$
$\wedge\ (\forall a \in \mathsf{T})\, \mathsf{w}(a) = 0 \Leftrightarrow a = 0$
$\wedge\ (\forall a, b \in \mathsf{T})\, b \neq 0 \Rightarrow \mathsf{w}(a \cdot b) \geqslant \mathsf{w}(a)$
$\wedge\ \mathsf{remainder} : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\wedge\ \mathsf{quotient} : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\wedge\ (\forall a, b \in \mathsf{T})\, b \neq 0 \Rightarrow a = \mathsf{quotient}(a, b) \cdot b + \mathsf{remainder}(a, b)$
$\wedge\ (\forall a, b \in \mathsf{T})\, b \neq 0 \Rightarrow \mathsf{w}(\mathsf{remainder}(a, b)) < \mathsf{w}(b)$

$\mathit{EuclideanSemimodule}(\mathsf{T}, \mathsf{S}) \triangleq$
$\quad \mathit{Semimodule}(\mathsf{T}, \mathsf{S})$
$\wedge\ \mathsf{remainder} : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\wedge\ \mathsf{quotient} : \mathsf{T} \times \mathsf{T} \to \mathsf{S}$
$\wedge\ (\forall a, b \in \mathsf{T})\, b \neq 0 \Rightarrow a = \mathsf{quotient}(a, b) \cdot b + \mathsf{remainder}(a, b)$
$\wedge\ (\forall a, b \in \mathsf{T})\,(a \neq 0 \vee b \neq 0) \Rightarrow \mathsf{gcd}(a, b)$ terminates

```
template<typename T, typename S>
    requires(EuclideanSemimodule(T, S))
T gcd(T a, T b)
{
    // Precondition: ¬(a = 0 ∧ b = 0)
    while (true) {
        if (b == T(0)) return a;
        a = remainder(a, b);
```

```
        if (a == T(0)) return b;
        b = remainder(b, a);
    }
}
```

$\mathit{ArchimedeanGroup}(\mathsf{T}) \triangleq$
$\quad \mathit{ArchimedeanMonoid}(\mathsf{T})$
$\wedge\ \mathit{AdditiveGroup}(\mathsf{T})$

$\mathit{DiscreteArchimedeanSemiring}(\mathsf{T}) \triangleq$
$\quad \mathit{CommutativeSemiring}(\mathsf{T})$
$\wedge\ \mathit{ArchimedeanMonoid}(\mathsf{T})$
$\wedge\ (\forall a, b, c \in \mathsf{T})\, a < b \wedge 0 < c \Rightarrow a \cdot c < b \cdot c$
$\wedge\ \neg(\exists a \in \mathsf{T})\, 0 < a < 1$

$\mathit{NonnegativeDiscreteArchimedeanSemiring}(\mathsf{T}) \triangleq$
$\quad \mathit{DiscreteArchimedeanSemiring}(\mathsf{T})$
$\wedge\ (\forall a \in \mathsf{T})\, 0 \leqslant a$

$\mathit{DiscreteArchimedeanRing}(\mathsf{T}) \triangleq$
$\quad \mathit{DiscreteArchimedeanSemiring}(\mathsf{T})$
$\wedge\ \mathit{AdditiveGroup}(\mathsf{T})$

Chapter 6: Iterators

$\mathit{Readable}(\mathsf{T}) \triangleq$
$\quad \mathit{Regular}(\mathsf{T})$
$\wedge\ \mathsf{ValueType} : \mathit{Readable} \to \mathit{Regular}$
$\wedge\ \mathsf{source} : \mathsf{T} \to \mathsf{ValueType}(\mathsf{T})$

$\mathit{Iterator}(\mathsf{T}) \triangleq$
$\quad \mathit{Regular}(\mathsf{T})$
$\wedge\ \mathsf{DistanceType} : \mathit{Iterator} \to \mathit{Integer}$
$\wedge\ \mathsf{successor} : \mathsf{T} \to \mathsf{T}$
$\wedge$ successor is not necessarily regular

property$(I : \mathit{Iterator})$
weak_range : $I \times \mathsf{DistanceType}(I)$
$\quad (f, n) \mapsto (\forall i \in \mathsf{DistanceType}(I))$
$\qquad (0 \leqslant i \leqslant n) \Rightarrow \mathsf{successor}^i(f)$ is defined

property$(I : \mathit{Iterator}, \mathsf{N} : \mathit{Integer})$
counted_range : $I \times \mathsf{N}$
$\quad (f, n) \mapsto \mathsf{weak\_range}(f, n) \wedge$
$\qquad (\forall i, j \in \mathsf{N})\, (0 \leqslant i < j \leqslant n) \Rightarrow$
$\qquad\quad \mathsf{successor}^i(f) \neq \mathsf{successor}^j(f)$

property$(\mathrm{I} : \mathit{Iterator})$
bounded_range : $\mathrm{I} \times \mathrm{I}$
$\quad (\mathsf{f}, \mathsf{l}) \mapsto (\exists \mathsf{k} \in \mathsf{DistanceType}(\mathrm{I}))\, \mathsf{counted\_range}(\mathsf{f}, \mathsf{k}) \wedge \mathsf{successor}^{\mathsf{k}}(\mathsf{f}) = \mathsf{l}$

property$(\mathrm{I} : \mathit{Readable})$
  requires(Iterator(I))
readable_bounded_range : $\mathrm{I} \times \mathrm{I}$
$\quad (\mathsf{f}, \mathsf{l}) \mapsto \mathsf{bounded\_range}(\mathsf{f}, \mathsf{l}) \wedge (\forall \mathsf{i} \in [\mathsf{f}, \mathsf{l}))\, \mathsf{source}(\mathsf{i})$ is defined

property$(\mathsf{Op} : \mathit{BinaryOperation})$
partially_associative : Op
$\quad \mathsf{op} \mapsto (\forall \mathsf{a}, \mathsf{b}, \mathsf{c} \in \mathsf{Domain}(\mathsf{op}))$
If $\mathsf{op}(\mathsf{a}, \mathsf{b})$ and $\mathsf{op}(\mathsf{b}, \mathsf{c})$ are defined,
$\mathsf{op}(\mathsf{op}(\mathsf{a}, \mathsf{b}), \mathsf{c})$ and $\mathsf{op}(\mathsf{a}, \mathsf{op}(\mathsf{b}, \mathsf{c}))$ are defined
and are equal.

$\mathit{ForwardIterator}(\mathsf{T}) \triangleq$
$\quad \mathit{Iterator}(\mathsf{T})$
$\wedge$ regular_unary_function(successor)

$\mathit{IndexedIterator}(\mathsf{T}) \triangleq$
$\quad \mathit{ForwardIterator}(\mathsf{T})$
$\wedge\ + : \mathsf{T} \times \mathsf{DistanceType}(\mathsf{T}) \to \mathsf{T}$
$\wedge\ - : \mathsf{T} \times \mathsf{T} \to \mathsf{DistanceType}(\mathsf{T})$
$\wedge\ +$ takes constant time
$\wedge\ -$ takes constant time

$\mathit{BidirectionalIterator}(\mathsf{T}) \triangleq$
$\quad \mathit{ForwardIterator}(\mathsf{T})$
$\wedge$ predecessor : $\mathsf{T} \to \mathsf{T}$
$\wedge$ predecessor takes constant time
$\wedge\ (\forall \mathsf{i} \in \mathsf{T})\, \mathsf{successor}(\mathsf{i})$ is defined $\Rightarrow$
predecessor(successor(i)) is defined and equals i
$\wedge\ (\forall \mathsf{i} \in \mathsf{T})\, \mathsf{predecessor}(\mathsf{i})$ is defined $\Rightarrow$
successor(predecessor(i)) is defined and equals i

$\mathit{RandomAccessIterator}(\mathsf{T}) \triangleq$
$\quad \mathit{IndexedIterator}(\mathsf{T}) \wedge \mathit{BidirectionalIterator}(\mathsf{T})$
$\wedge\ \mathit{TotallyOrdered}(\mathsf{T})$
$\wedge\ (\forall \mathsf{i}, \mathsf{j} \in \mathsf{T})\, \mathsf{i} < \mathsf{j} \Leftrightarrow \mathsf{i} \prec \mathsf{j}$
$\wedge$ DifferenceType : $\mathit{RandomAccessIterator} \to \mathit{Integer}$
$\wedge\ + : \mathsf{T} \times \mathsf{DifferenceType}(\mathsf{T}) \to \mathsf{T}$
$\wedge\ - : \mathsf{T} \times \mathsf{DifferenceType}(\mathsf{T}) \to \mathsf{T}$
$\wedge\ - : \mathsf{T} \times \mathsf{T} \to \mathsf{DifferenceType}(\mathsf{T})$
$\wedge\ <$ takes constant time
$\wedge\ -$ between an iterator and an integer takes constant time

Chapter 7: Coordinate Structures

$\mathit{BifurcateCoordinate}(\mathsf{T}) \triangleq$
$\quad \mathit{Regular}(\mathsf{T})$
$\wedge$ WeightType : $\mathit{BifurcateCoordinate} \to \mathit{Integer}$
$\wedge$ empty : T $\to$ bool
$\wedge$ has_left_successor : T $\to$ bool
$\wedge$ has_right_successor : T $\to$ bool
$\wedge$ left_successor : T $\to$ T
$\wedge$ right_successor : T $\to$ T
$\wedge$ $(\forall i, j \in T)$ (left_successor(i) = j $\vee$ right_successor(i) = j) $\Rightarrow$ $\neg$empty(j)

property(C : *BifurcateCoordinate*)
tree : C
$\quad x \mapsto$ the descendants of x form a tree

$\mathit{BidirectionalBifurcateCoordinate}(\mathsf{T}) \triangleq$
$\quad \mathit{BifurcateCoordinate}(\mathsf{T})$
$\wedge$ has_predecessor : T $\to$ bool
$\wedge$ $(\forall i \in T)$ $\neg$empty(i) $\Rightarrow$ has_predecessor(i) is defined
$\wedge$ predecessor : T $\to$ T
$\wedge$ $(\forall i \in T)$ has_left_successor(i) $\Rightarrow$
predecessor(left_successor(i)) is defined and equals i
$\wedge$ $(\forall i \in T)$ has_right_successor(i) $\Rightarrow$
predecessor(right_successor(i)) is defined and equals i
$\wedge$ $(\forall i \in T)$ has_predecessor(i) $\Rightarrow$
is_left_successor(i) $\vee$ is_right_successor(i)

```
template<typename T>
    requires(BidirectionalBifurcateCoordinate(T))
bool is_left_successor(T j)
{
    // Precondition: has_predecessor(j)
    T i = predecessor(j);
    return has_left_successor(i) && left_successor(i) == j;
}

template<typename T>
    requires(BidirectionalBifurcateCoordinate(T))
bool is_right_successor(T j)
{
    // Precondition: has_predecessor(j)
    T i = predecessor(j);
    return has_right_successor(i) && right_successor(i) == j;
}
```

property(C : *Readable*)
requires(BifurcateCoordinate(C))

$\mathsf{readable\_tree} : \mathsf{C}$
$x \mapsto \mathsf{tree}(x) \wedge (\forall y \in \mathsf{C})\, \mathsf{reachable}(x, y) \Rightarrow \mathsf{source}(y)$ is defined

Chapter 8: Coordinates with Mutable Successors

$$\begin{aligned}
&\mathit{ForwardLinker}(\mathsf{S}) \triangleq \\
&\quad \mathsf{IteratorType} : \mathit{ForwardLinker} \to \mathit{ForwardIterator} \\
&\wedge\ \text{Let } \mathrm{I} = \mathsf{IteratorType}(\mathsf{S}) \text{ in:} \\
&\qquad (\forall s \in \mathsf{S})\, (s : \mathrm{I} \times \mathrm{I} \to \mathsf{void}) \\
&\qquad \wedge\ (\forall s \in \mathsf{S})\, (\forall i, j \in \mathrm{I}) \text{ if } \mathsf{successor}(i) \text{ is defined,} \\
&\qquad\quad \text{then } s(i, j) \text{ establishes } \mathsf{successor}(i) = j
\end{aligned}$$

$$\begin{aligned}
&\mathit{BackwardLinker}(\mathsf{S}) \triangleq \\
&\quad \mathsf{IteratorType} : \mathit{BackwardLinker} \to \mathit{BidirectionalIterator} \\
&\wedge\ \text{Let } \mathrm{I} = \mathsf{IteratorType}(\mathsf{S}) \text{ in:} \\
&\qquad (\forall s \in \mathsf{S})\, (s : \mathrm{I} \times \mathrm{I} \to \mathsf{void}) \\
&\qquad \wedge\ (\forall s \in \mathsf{S})\, (\forall i, j \in \mathrm{I}) \text{ if } \mathsf{predecessor}(j) \text{ is defined,} \\
&\qquad\quad \text{then } s(i, j) \text{ establishes } i = \mathsf{predecessor}(j)
\end{aligned}$$

$$\mathit{BidirectionalLinker}(\mathsf{S}) \triangleq \mathit{ForwardLinker}(\mathsf{S}) \wedge \mathit{BackwardLinker}(\mathsf{S})$$

property$(\mathrm{I} : \mathit{Iterator})$
$\mathsf{disjoint} : \mathrm{I} \times \mathrm{I} \times \mathrm{I} \times \mathrm{I}$
$(f0, l0, f1, l1) \mapsto (\forall i \in \mathrm{I})\, \neg(i \in [f0, l0) \wedge i \in [f1, l1))$

$$\begin{aligned}
&\mathit{LinkedBifurcateCoordinate}(\mathsf{T}) \triangleq \\
&\quad \mathit{BifurcateCoordinate}(\mathsf{T}) \\
&\wedge\ \mathsf{set_left_successor} : \mathsf{T} \times \mathsf{T} \to \mathsf{void} \\
&\qquad (i, j) \mapsto \text{ establishes } \mathsf{left_successor}(i) = j \\
&\wedge\ \mathsf{set_right_successor} : \mathsf{T} \times \mathsf{T} \to \mathsf{void} \\
&\qquad (i, j) \mapsto \text{ establishes } \mathsf{right_successor}(i) = j
\end{aligned}$$

$$\begin{aligned}
&\mathit{EmptyLinkedBifurcateCoordinate}(\mathsf{T}) \triangleq \\
&\quad \mathit{LinkedBifurcateCoordinate}(\mathsf{T}) \\
&\wedge\ \mathsf{empty}(\mathsf{T}())^{1} \\
&\wedge\ \neg\mathsf{empty}(i) \Rightarrow \\
&\qquad \mathsf{left_successor}(i) \text{ and } \mathsf{right_successor}(i) \text{ are defined} \\
&\wedge\ \neg\mathsf{empty}(i) \Rightarrow \\
&\qquad (\neg\mathsf{has_left_successor}(i) \Leftrightarrow \mathsf{empty}(\mathsf{left_successor}(i))) \\
&\wedge\ \neg\mathsf{empty}(i) \Rightarrow \\
&\qquad (\neg\mathsf{has_right_successor}(i) \Leftrightarrow \mathsf{empty}(\mathsf{right_successor}(i)))
\end{aligned}$$

[1] In other words, empty is true on the default constructed value and possibly on other values as well.

Chapter 9: Copying

$\mathit{Writable}(\mathsf{T}) \triangleq$
$\quad \mathsf{ValueType} : \mathit{Writable} \to \mathit{Regular}$
$\wedge\ (\forall x \in \mathsf{T})\, (\forall v \in \mathsf{ValueType}(\mathsf{T}))\, \mathsf{sink}(x) \leftarrow v$ is a well-formed statement

property$(\mathsf{T} : \mathit{Writable}, \mathsf{U} : \mathit{Readable})$
requires$(\mathsf{ValueType}(\mathsf{T}) = \mathsf{ValueType}(\mathsf{U}))$
$\mathsf{aliased} : \mathsf{T} \times \mathsf{U}$
$(x, y) \mapsto \mathsf{sink}(x)$ is defined $\wedge$
$\mathsf{source}(y)$ is defined $\wedge$
$(\forall v \in \mathsf{ValueType}(\mathsf{T}))\, \mathsf{sink}(x) \leftarrow v$ establishes $\mathsf{source}(y) = v$

$\mathit{Mutable}(\mathsf{T}) \triangleq$
$\quad \mathit{Readable}(\mathsf{T}) \wedge \mathit{Writable}(\mathsf{T})$
$\wedge\ (\forall x \in \mathsf{T})\, \mathsf{sink}(x)$ is defined $\Leftrightarrow \mathsf{source}(x)$ is defined
$\wedge\ (\forall x \in \mathsf{T})\, \mathsf{sink}(x)$ is defined $\Rightarrow \mathsf{aliased}(x, x)$
$\wedge\ \mathsf{deref} : \mathsf{T} \to \mathsf{ValueType}(\mathsf{T})\&$
$\wedge\ (\forall x \in \mathsf{T})\, \mathsf{sink}(x)$ is defined $\Leftrightarrow \mathsf{deref}(x)$ is defined

property$(\mathrm{I} : \mathit{Writable})$
requires$(\mathit{Iterator}(\mathrm{I}))$
$\mathsf{writable\_bounded\_range} : \mathrm{I} \times \mathrm{I}$
$(f, l) \mapsto \mathsf{bounded\_range}(f, l) \wedge (\forall i \in [f, l))\, \mathsf{sink}(i)$ is defined

writable_weak_range and writable_counted_range are defined similarly.

property$(\mathrm{I} : \mathit{Mutable})$
requires$(\mathit{ForwardIterator}(\mathrm{I}))$
$\mathsf{mutable\_bounded\_range} : \mathrm{I} \times \mathrm{I}$
$(f, l) \mapsto \mathsf{bounded\_range}(f, l) \wedge (\forall i \in [f, l))\, \mathsf{sink}(i)$ is defined

mutable_weak_range and mutable_counted_range are defined similarly.

property$(\mathrm{I} : \mathit{Readable}, \mathsf{O} : \mathit{Writable})$
requires$(\mathit{Iterator}(\mathrm{I}) \wedge \mathit{Iterator}(\mathsf{O}))$
$\mathsf{not\_overlapped\_forward} : \mathrm{I} \times \mathrm{I} \times \mathsf{O} \times \mathsf{O}$
$(f_i, l_i, f_o, l_o) \mapsto$
$\mathsf{readable\_bounded\_range}(f_i, l_i) \wedge$
$\mathsf{writable\_bounded\_range}(f_o, l_o) \wedge$
$(\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o))$
$\mathsf{aliased}(k_o, k_i) \Rightarrow k_i - f_i \leqslant k_o - f_o$

property$(\mathrm{I} : \mathit{Readable}, \mathsf{O} : \mathit{Writable})$
requires$(\mathit{Iterator}(\mathrm{I}) \wedge \mathit{Iterator}(\mathsf{O}))$
$\mathsf{not\_overlapped\_backward} : \mathrm{I} \times \mathrm{I} \times \mathsf{O} \times \mathsf{O}$
$(f_i, l_i, f_o, l_o) \mapsto$
$\mathsf{readable\_bounded\_range}(f_i, l_i) \wedge$
$\mathsf{writable\_bounded\_range}(f_o, l_o) \wedge$

$$(\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o))$$
$$\quad \mathsf{aliased}(k_o, k_i) \Rightarrow l_i - k_i \leqslant l_o - k_o$$

$$\begin{array}{l}
\textbf{property}(\mathrm{I} : \mathit{Readable}, \mathsf{O} : \mathit{Writable}) \\
\quad \textbf{requires}(\mathsf{Iterator}(\mathrm{I}) \wedge \mathsf{Iterator}(\mathsf{O})) \\
\mathsf{not_overlapped} : \mathrm{I} \times \mathrm{I} \times \mathsf{O} \times \mathsf{O} \\
\quad (f_i, l_i, f_o, l_o) \mapsto \\
\qquad \mathsf{readable_bounded_range}(f_i, l_i) \wedge \\
\qquad \mathsf{writable_bounded_range}(f_o, l_o) \wedge \\
\qquad (\forall k_i \in [f_i, l_i))\,(\forall k_o \in [f_o, l_o))\,\neg\mathsf{aliased}(k_o, k_i)
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{T} : \mathit{Writable}, \mathsf{U} : \mathit{Writable}) \\
\quad \textbf{requires}(\mathsf{ValueType}(\mathsf{T}) = \mathsf{ValueType}(\mathsf{U})) \\
\mathsf{write_aliased} : \mathsf{T} \times \mathsf{U} \\
\quad (x, y) \mapsto \mathsf{sink}(x) \text{ is defined} \wedge \mathsf{sink}(y) \text{ is defined} \wedge \\
\qquad\qquad (\forall \mathsf{V} \in \mathit{Readable})\,(\forall v \in \mathsf{V})\,\mathsf{aliased}(x, v) \Leftrightarrow \mathsf{aliased}(y, v)
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathsf{O}_0 : \mathit{Writable}, \mathsf{O}_1 : \mathit{Writable}) \\
\quad \textbf{requires}(\mathsf{Iterator}(\mathsf{O}_0) \wedge \mathsf{Iterator}(\mathsf{O}_1)) \\
\mathsf{not_write_overlapped} : \mathsf{O}_0 \times \mathsf{O}_0 \times \mathsf{O}_1 \times \mathsf{O}_1 \\
\quad (f_0, l_0, f_1, l_1) \mapsto \\
\qquad \mathsf{writable_bounded_range}(f_0, l_0) \wedge \\
\qquad \mathsf{writable_bounded_range}(f_1, l_1) \wedge \\
\qquad (\forall k_0 \in [f_0, l_0))(\forall k_1 \in [f_1, l_1))\,\neg\mathsf{write_aliased}(k_0, k_1)
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathrm{I} : \mathit{Readable}, \mathsf{O} : \mathit{Writable}, \mathsf{N} : \mathit{Integer}) \\
\quad \textbf{requires}(\mathit{Iterator}(\mathrm{I}) \wedge \mathit{Iterator}(\mathsf{O})) \\
\mathsf{backward_offset} : \mathrm{I} \times \mathrm{I} \times \mathsf{O} \times \mathsf{O} \times \mathsf{N} \\
\quad (f_i, l_i, f_o, l_o, n) \mapsto \\
\qquad \mathsf{readable_bounded_range}(f_i, l_i) \wedge \\
\qquad n \geqslant 0 \wedge \\
\qquad \mathsf{writable_bounded_range}(f_o, l_o) \wedge \\
\qquad (\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o)) \\
\qquad\quad \mathsf{aliased}(k_o, k_i) \Rightarrow k_i - f_i + n \leqslant k_o - f_o
\end{array}$$

$$\begin{array}{l}
\textbf{property}(\mathrm{I} : \mathit{Readable}, \mathsf{O} : \mathit{Writable}, \mathsf{N} : \mathit{Integer}) \\
\quad \textbf{requires}(\mathit{Iterator}(\mathrm{I}) \wedge \mathit{Iterator}(\mathsf{O})) \\
\mathsf{forward_offset} : \mathrm{I} \times \mathrm{I} \times \mathsf{O} \times \mathsf{O} \times \mathsf{N} \\
\quad (f_i, l_i, f_o, l_o, n) \mapsto \\
\qquad \mathsf{readable_bounded_range}(f_i, l_i) \wedge \\
\qquad n \geqslant 0 \wedge \\
\qquad \mathsf{writable_bounded_range}(f_o, l_o) \wedge \\
\qquad (\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o)) \\
\qquad\quad \mathsf{aliased}(k_o, k_i) \Rightarrow l_i - k_i + n \leqslant l_o - k_o
\end{array}$$

Chapter 10: Rearrangements

Chapter 11: Partition and Merging

$$
\begin{array}{l}
\textbf{property}(\mathrm{I} : \mathit{ForwardIterator}, \mathsf{N} : \mathit{Integer}, \mathsf{R} : \mathit{Relation}) \\
\quad \textbf{requires}(\mathit{Mutable}(\mathrm{I}) \wedge \mathsf{ValueType}(\mathrm{I}) = \mathsf{Domain}(\mathsf{R})) \\
\mathsf{mergeable} : \mathrm{I} \times \mathsf{N} \times \mathrm{I} \times \mathsf{N} \times \mathsf{R} \\
\quad (f_0, n_0, f_1, n_1, r) \mapsto f_0 + n_0 = f_1 \wedge \\
\qquad\qquad\qquad\qquad \mathsf{mutable\_counted\_range}(f_0, n_0 + n_1) \wedge \\
\qquad\qquad\qquad\qquad \mathsf{weak\_ordering}(r) \wedge \\
\qquad\qquad\qquad\qquad \mathsf{increasing\_counted\_range}(f_0, n_0, r) \wedge \\
\qquad\qquad\qquad\qquad \mathsf{increasing\_counted\_range}(f_1, n_1, r)
\end{array}
$$

Chapter 12: Composite Objects

$$
\begin{array}{l}
\mathit{Linearizable}(W) \triangleq \\
\qquad \mathit{Regular}(W) \\
\quad \wedge\ \mathsf{IteratorType} : \mathit{Linearizable} \to \mathit{Iterator} \\
\quad \wedge\ \mathsf{ValueType} : \mathit{Linearizable} \to \mathit{Regular} \\
\qquad\quad W \mapsto \mathsf{ValueType}(\mathsf{IteratorType}(W)) \\
\quad \wedge\ \mathsf{SizeType} : \mathit{Linearizable} \to \mathit{Integer} \\
\qquad\quad W \mapsto \mathsf{DistanceType}(\mathsf{IteratorType}(W)) \\
\quad \wedge\ \mathsf{begin} : W \to \mathsf{IteratorType}(W) \\
\quad \wedge\ \mathsf{end} : W \to \mathsf{IteratorType}(W) \\
\quad \wedge\ \mathsf{size} : W \to \mathsf{SizeType}(W) \\
\qquad\quad x \mapsto \mathsf{end}(x) - \mathsf{begin}(x) \\
\quad \wedge\ \mathsf{empty} : W \to \mathsf{bool} \\
\qquad\quad x \mapsto \mathsf{begin}(x) = \mathsf{end}(x) \\
\quad \wedge\ [\,] : W \times \mathsf{SizeType}(W) \to \mathsf{ValueType}(W)\& \\
\qquad\quad (w, i) \mapsto \mathsf{deref}(\mathsf{begin}(w) + i)
\end{array}
$$

$$
\begin{array}{l}
\mathit{Sequence}(\mathsf{S}) \triangleq \\
\qquad \mathit{Linearizable}(\mathsf{S}) \\
\quad \wedge\ (\forall s \in \mathsf{S})\,(\forall i \in [\mathsf{begin}(s), \mathsf{end}(s)))\,\mathsf{deref}(i) \text{ is a part of } s \\
\quad \wedge\ = : \mathsf{S} \times \mathsf{S} \to \mathsf{bool} \\
\qquad\quad (s, s') \mapsto \mathsf{lexicographical\_equal}(\\
\qquad\qquad\qquad \mathsf{begin}(s), \mathsf{end}(s), \mathsf{begin}(s'), \mathsf{end}(s')) \\
\quad \wedge\ < : \mathsf{S} \times \mathsf{S} \to \mathsf{bool} \\
\qquad\quad (s, s') \mapsto \mathsf{lexicographical\_less}(\\
\qquad\qquad\qquad \mathsf{begin}(s), \mathsf{end}(s), \mathsf{begin}(s'), \mathsf{end}(s'))
\end{array}
$$

Index