Concept definitions from *Elements of Programming*

Alexander Stepanov Paul McJones

January 16, 2017

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.

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

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

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

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

property(F : *UnaryFunction*)
regular_unary_function : F

$$\mathsf{f} \mapsto (\forall \mathsf{f}' \in \mathsf{F})(\forall x, x' \in \mathsf{Domain}(\mathsf{F}))$$
$$(\mathsf{f} = \mathsf{f}' \wedge x = x') \Rightarrow (\mathsf{f}(x) = \mathsf{f}'(x'))$$

Chapter 2: Transformations and Their Orbits

$Predicate(\mathsf{P}) \triangleq$
$\quad FunctionalProcedure(\mathsf{P})$
$\wedge\ \mathsf{Codomain}(\mathsf{P}) = \mathsf{bool}$

$HomogeneousPredicate(\mathsf{P}) \triangleq$
$\quad Predicate(\mathsf{P})$
$\wedge\ HomogeneousFunction(\mathsf{P})$

$UnaryPredicate(\mathsf{P}) \triangleq$
$\quad Predicate(\mathsf{P})$
$\wedge\ UnaryFunction(\mathsf{P})$

$Operation(\mathsf{Op}) \triangleq$
$\quad HomogeneousFunction(\mathsf{Op})$
$\wedge\ \mathsf{Codomain}(\mathsf{Op}) = \mathsf{Domain}(\mathsf{Op})$

$Transformation(\mathsf{F}) \triangleq$
$\quad Operation(\mathsf{F})$
$\wedge\ UnaryFunction(\mathsf{F})$
$\wedge\ \mathsf{DistanceType} : Transformation \to Integer$

Chapter 3: Associative Operations

$BinaryOperation(\mathsf{Op}) \triangleq$
$\quad Operation(\mathsf{Op})$
$\wedge\ \mathsf{Arity}(\mathsf{Op}) = 2$

property$(\mathsf{Op} : BinaryOperation)$
associative : Op
$\quad \mathsf{op} \mapsto (\forall a, b, c \in \mathsf{Domain}(\mathsf{op}))\ \mathsf{op}(\mathsf{op}(a, b), c) = \mathsf{op}(a, \mathsf{op}(b, c))$

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

$\wedge$ 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

Relation(R) $\triangleq$
    HomogeneousPredicate(R)
$\wedge$ Arity(R) $= 2$

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
    $r \mapsto \mathsf{transitive}(r) \wedge \mathsf{reflexive}(r) \wedge \mathsf{symmetric}(r)$

$\textsf{\textbf{property}}(\mathsf{F} : \mathit{UnaryFunction}, \mathsf{R} : \mathit{Relation})$
$\quad \textsf{\textbf{requires}}(\mathsf{Domain}(\mathsf{F}) = \mathsf{Domain}(\mathsf{R}))$
$\mathsf{key\_function} : \mathsf{F} \times \mathsf{R}$
$\quad (f, r) \mapsto (\forall a, b \in \mathsf{Domain}(\mathsf{F}))\,(r(a, b) \Leftrightarrow f(a) = f(b))$

$\textsf{\textbf{property}}(\mathsf{R} : \mathit{Relation})$
$\mathsf{total\_ordering} : \mathsf{R}$
$\quad r \mapsto \mathsf{transitive}(r) \wedge$
$\qquad (\forall a, b \in \mathsf{Domain}(\mathsf{R}))$ exactly one of the following holds:
$\qquad\quad r(a, b), r(b, a),$ or $a = b$

$\textsf{\textbf{property}}(\mathsf{R} : \mathit{Relation})$
$\mathsf{total\_ordering} : \mathsf{R}$
$\quad r \mapsto \mathsf{transitive}(r) \wedge$
$\qquad (\forall a, b \in \mathsf{Domain}(\mathsf{R}))$ exactly one of the following holds:
$\qquad\quad r(a, b), r(b, a),$ or $a = b$

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

$\mathit{TotallyOrdered}(\mathsf{T}) \triangleq$
$\qquad \mathit{Regular}(\mathsf{T})$
$\quad \wedge\ <: \mathsf{T} \times \mathsf{T} \to \mathsf{bool}$
$\quad \wedge\ \mathsf{total\_ordering}(<)$

Chapter 5: Ordered Algebraic Structures

$\textsf{\textbf{property}}(\mathsf{T} : \mathit{Regular}, \mathsf{Op} : \mathit{BinaryOperation})$
$\quad \textsf{\textbf{requires}}(\mathsf{T} = \mathsf{Domain}(\mathsf{Op}))$
$\mathsf{identity\_element} : \mathsf{T} \times \mathsf{Op}$
$\quad (e, op) \mapsto (\forall a \in \mathsf{T})\ op(a, e) = op(e, a) = a$

$\textsf{\textbf{property}}(\mathsf{F} : \mathit{Transformation}, \mathsf{T} : \mathit{Regular}, \mathsf{Op} : \mathit{BinaryOperation})$
$\quad \textsf{\textbf{requires}}(\mathsf{Domain}(\mathsf{F}) = \mathsf{T} = \mathsf{Domain}(\mathsf{Op}))$
$\mathsf{inverse\_operation} : \mathsf{F} \times \mathsf{T} \times \mathsf{Op}$
$\quad (inv, e, op) \mapsto (\forall a \in \mathsf{T})\ op(a, inv(a)) = op(inv(a), a) = e$

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

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

$\wedge$ associative$(+)$
$\wedge$ commutative$(+)$

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

$AdditiveMonoid(\mathsf{T}) \triangleq$
$AdditiveSemigroup(\mathsf{T})$
$\wedge\ 0 \in \mathsf{T}$
$\wedge$ identity_element$(0, +)$

$MultiplicativeMonoid(\mathsf{T}) \triangleq$
$MultiplicativeSemigroup(\mathsf{T})$
$\wedge\ 1 \in \mathsf{T}$
$\wedge$ identity_element$(1, \cdot)$

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

$MultiplicativeGroup(\mathsf{T}) \triangleq$
$MultiplicativeMonoid(\mathsf{T})$
$\wedge$ multiplicative_inverse : $\mathsf{T} \to \mathsf{T}$
$\wedge$ inverse_operation(multiplicative_inverse$, 1, \cdot)$
$\wedge\ / : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$(a, b) \mapsto a \cdot$ multiplicative_inverse(b)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
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;
}
```

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

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

$HalvableMonoid(\mathsf{T}) \triangleq$

$\quad ArchimedeanMonoid(\mathsf{T})$

$\wedge\ \mathsf{half} : \mathsf{T} \to \mathsf{T}$

$\wedge\ (\forall a, b \in \mathsf{T})\,(b > 0 \wedge a = b + b) \Rightarrow \mathsf{half}(a) = b$

```
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;
    }
}
```

$EuclideanMonoid(\mathsf{T}) \triangleq$

$\quad ArchimedeanMonoid(\mathsf{T})$

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

$EuclideanSemiring(\mathsf{T}) \triangleq$

$\quad CommutativeSemiring(\mathsf{T})$

$\wedge\ \mathsf{NormType} : EuclideanSemiring \to Integer$

$\wedge\ \mathsf{w} : \mathsf{T} \to \mathsf{NormType}(\mathsf{T})$

$\wedge\ (\forall a \in \mathsf{T})\,\mathsf{w}(a) \geqslant 0$

$\wedge\ (\forall a \in \mathsf{T})\,\mathsf{w}(a) = 0 \Leftrightarrow a = 0$

$\wedge\ (\forall a, b \in \mathsf{T})\,b \neq 0 \Rightarrow \mathsf{w}(a \cdot b) \geqslant \mathsf{w}(a)$

$\wedge\ \mathsf{remainder} : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$

$\wedge\ \mathsf{quotient} : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$

$\wedge\ (\forall a, b \in \mathsf{T})\,b \neq 0 \Rightarrow a = \mathsf{quotient}(a, b) \cdot b + \mathsf{remainder}(a, b)$

$\wedge\ (\forall a, b \in \mathsf{T})\,b \neq 0 \Rightarrow \mathsf{w}(\mathsf{remainder}(a, b)) < \mathsf{w}(b)$

$EuclideanSemimodule(\mathsf{T}, \mathsf{S}) \triangleq$

$\quad Semimodule(\mathsf{T}, \mathsf{S})$

$\wedge\ \mathsf{remainder} : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$

$\wedge\ \mathsf{quotient} : \mathsf{T} \times \mathsf{T} \to \mathsf{S}$

$\wedge\ (\forall a, b \in \mathsf{T})\,b \neq 0 \Rightarrow a = \mathsf{quotient}(a, b) \cdot b + \mathsf{remainder}(a, b)$

$\wedge\ (\forall a, b \in \mathsf{T})\,(a \neq 0 \vee b \neq 0) \Rightarrow \mathsf{gcd}(a, b)$ 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);
    }
}
```

$$\begin{aligned}&\mathit{ArchimedeanGroup}(\mathsf{T}) \triangleq\\ &\qquad \mathit{ArchimedeanMonoid}(\mathsf{T})\\ &\quad\wedge\ \mathit{AdditiveGroup}(\mathsf{T})\end{aligned}$$

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

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

$$\begin{aligned}&\mathit{DiscreteArchimedeanRing}(\mathsf{T}) \triangleq\\ &\qquad \mathit{DiscreteArchimedeanSemiring}(\mathsf{T})\\ &\quad\wedge\ \mathit{AdditiveGroup}(\mathsf{T})\end{aligned}$$

Chapter 6: Iterators

$$\begin{aligned}&\mathit{Readable}(\mathsf{T}) \triangleq\\ &\qquad \mathit{Regular}(\mathsf{T})\\ &\quad\wedge\ \mathsf{ValueType} : \mathit{Readable} \to \mathit{Regular}\\ &\quad\wedge\ \mathsf{source} : \mathsf{T} \to \mathsf{ValueType}(\mathsf{T})\end{aligned}$$

$$\begin{aligned}&\mathit{Iterator}(\mathsf{T}) \triangleq\\ &\qquad \mathit{Regular}(\mathsf{T})\\ &\quad\wedge\ \mathsf{DistanceType} : \mathit{Iterator} \to \mathit{Integer}\\ &\quad\wedge\ \mathsf{successor} : \mathsf{T} \to \mathsf{T}\\ &\quad\wedge\ \mathsf{successor} \text{ is not necessarily regular}\end{aligned}$$

$$\begin{aligned}&\textbf{property}(I : \mathit{Iterator})\\ &\mathsf{weak\_range} : I \times \mathsf{DistanceType}(I)\\ &\quad (f, n) \mapsto (\forall i \in \mathsf{DistanceType}(I))\\ &\qquad\qquad (0 \leqslant i \leqslant n) \Rightarrow \mathsf{successor}^i(f) \text{ is defined}\end{aligned}$$

$$\begin{aligned}&\textbf{property}(I : \mathit{Iterator})\\ &\mathsf{counted\_range} : I \times \mathsf{DistanceType}(I)\\ &\quad (f, n) \mapsto \mathsf{weak\_range}(f, n) \wedge\\ &\qquad\qquad (\forall i, j \in \mathsf{DistanceType}(I))\, (0 \leqslant i < j \leqslant n) \Rightarrow\\ &\qquad\qquad\qquad \mathsf{successor}^i(f) \neq \mathsf{successor}^j(f)\end{aligned}$$

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

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

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

$\textit{ForwardIterator}(T) \triangleq$
$\quad \textit{Iterator}(T)$
$\wedge\ \mathsf{regular\_unary\_function}(\mathsf{successor})$

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

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

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

Chapter 7: Coordinate Structures

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

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

BidirectionalBifurcateCoordinate$(\mathsf{T}) \triangleq$
BifurcateCoordinate(T)
$\wedge$ has_predecessor : T $\rightarrow$ bool
$\wedge$ $(\forall i \in \mathsf{T})\, \neg$empty$(i) \Rightarrow$ has_predecessor(i) is defined
$\wedge$ predecessor : T $\rightarrow$ T
$\wedge$ $(\forall i \in \mathsf{T})$ has_left_successor$(i) \Rightarrow$
predecessor(left_successor(i)) is defined and equals i
$\wedge$ $(\forall i \in \mathsf{T})$ has_right_successor$(i) \Rightarrow$
predecessor(right_successor(i)) is defined and equals i
$\wedge$ $(\forall i \in \mathsf{T})$ 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))

$\mathsf{readable\_tree} : \mathsf{C}$

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

Chapter 8: Coordinates with Mutable Successors

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

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

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

property$(\mathrm{I} : \mathit{Iterator})$

$\mathsf{disjoint} : \mathrm{I} \times \mathrm{I} \times \mathrm{I} \times \mathrm{I}$

$\quad (f0, l0, f1, l1) \mapsto (\forall i \in \mathrm{I})\, \neg(i \in [f0, l0) \wedge i \in [f1, l1))$

$$
\begin{aligned}
&\mathit{LinkedBifurcateCoordinate}(\mathsf{T}) \triangleq \\
&\qquad \mathit{BifurcateCoordinate}(\mathsf{T}) \\
&\quad \wedge\ \mathsf{set\_left\_successor} : \mathsf{T} \times \mathsf{T} \to \mathsf{void} \\
&\qquad\qquad (i, j) \mapsto \text{ establishes } \mathsf{left\_successor}(i) = j \\
&\quad \wedge\ \mathsf{set\_right\_successor} : \mathsf{T} \times \mathsf{T} \to \mathsf{void} \\
&\qquad\qquad (i, j) \mapsto \text{ establishes } \mathsf{right\_successor}(i) = j
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{EmptyLinkedBifurcateCoordinate}(\mathsf{T}) \triangleq \\
&\qquad \mathit{LinkedBifurcateCoordinate}(\mathsf{T}) \\
&\quad \wedge\ \mathsf{empty}(\mathsf{T}())^{1} \\
&\quad \wedge\ \neg\mathsf{empty}(i) \Rightarrow \\
&\qquad\quad \mathsf{left\_successor}(i) \text{ and } \mathsf{right\_successor}(i) \text{ are defined} \\
&\quad \wedge\ \neg\mathsf{empty}(i) \Rightarrow \\
&\qquad\quad (\neg\mathsf{has\_left\_successor}(i) \Leftrightarrow \mathsf{empty}(\mathsf{left\_successor}(i))) \\
&\quad \wedge\ \neg\mathsf{empty}(i) \Rightarrow \\
&\qquad\quad (\neg\mathsf{has\_right\_successor}(i) \Leftrightarrow \mathsf{empty}(\mathsf{right\_successor}(i)))
\end{aligned}
$$

---

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

Chapter 9: Copying

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

$$\begin{aligned}
&\textbf{property}(\mathsf{T} : \textit{Writable}, \mathsf{U} : \textit{Readable}) \\
&\quad \textbf{requires}(\mathsf{ValueType}(\mathsf{T}) = \mathsf{ValueType}(\mathsf{U})) \\
&\mathsf{aliased} : \mathsf{T} \times \mathsf{U} \\
&\quad (x, y) \mapsto \mathsf{sink}(x) \text{ is defined} \wedge \\
&\qquad\qquad \mathsf{source}(y) \text{ is defined} \wedge \\
&\qquad\qquad (\forall v \in \mathsf{ValueType}(\mathsf{T}))\,\mathsf{sink}(x) \leftarrow v \text{ establishes } \mathsf{source}(y) = v
\end{aligned}$$

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

$$\begin{aligned}
&\textbf{property}(I : \textit{Writable}) \\
&\quad \textbf{requires}(\textit{Iterator}(I)) \\
&\mathsf{writable_bounded_range} : I \times I \\
&\quad (f, l) \mapsto \mathsf{bounded_range}(f, l) \wedge (\forall i \in [f, l))\,\mathsf{sink}(i) \text{ is defined}
\end{aligned}$$

writable_weak_range and writable_counted_range are defined similarly.

$$\begin{aligned}
&\textbf{property}(I : \textit{Mutable}) \\
&\quad \textbf{requires}(\textit{ForwardIterator}(I)) \\
&\mathsf{mutable_bounded_range} : I \times I \\
&\quad (f, l) \mapsto \mathsf{bounded_range}(f, l) \wedge (\forall i \in [f, l))\,\mathsf{sink}(i) \text{ is defined}
\end{aligned}$$

mutable_weak_range and mutable_counted_range are defined similarly.

$$\begin{aligned}
&\textbf{property}(I : \textit{Readable}, \mathsf{O} : \textit{Writable}) \\
&\quad \textbf{requires}(\textit{Iterator}(I) \wedge \textit{Iterator}(\mathsf{O})) \\
&\mathsf{not_overlapped_forward} : I \times I \times \mathsf{O} \times \mathsf{O} \\
&\quad (f_i, l_i, f_o, l_o) \mapsto \\
&\qquad \mathsf{readable_bounded_range}(f_i, l_i) \wedge \\
&\qquad \mathsf{writable_bounded_range}(f_o, l_o) \wedge \\
&\qquad (\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o)) \\
&\qquad\quad \mathsf{aliased}(k_o, k_i) \Rightarrow k_i - f_i \leqslant k_o - f_o
\end{aligned}$$

$$\begin{aligned}
&\textbf{property}(I : \textit{Readable}, \mathsf{O} : \textit{Writable}) \\
&\quad \textbf{requires}(\textit{Iterator}(I) \wedge \textit{Iterator}(\mathsf{O})) \\
&\mathsf{not_overlapped_backward} : I \times I \times \mathsf{O} \times \mathsf{O} \\
&\quad (f_i, l_i, f_o, l_o) \mapsto \\
&\qquad \mathsf{readable_bounded_range}(f_i, l_i) \wedge \\
&\qquad \mathsf{writable_bounded_range}(f_o, l_o) \wedge
\end{aligned}$$

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

$\textbf{property}(I : \textit{Readable}, O : \textit{Writable})$
$\quad \textbf{requires}(\mathsf{Iterator}(I) \wedge \mathsf{Iterator}(O))$
$\mathsf{not\_overlapped} : I \times I \times O \times O$
$\quad (f_i, l_i, f_o, l_o) \mapsto$
$\quad\quad \mathsf{readable\_bounded\_range}(f_i, l_i) \wedge$
$\quad\quad \mathsf{writable\_bounded\_range}(f_o, l_o) \wedge$
$\quad\quad (\forall k_i \in [f_i, l_i))\,(\forall k_o \in [f_o, l_o))\,\neg\mathsf{aliased}(k_o, k_i)$

$\textbf{property}(T : \textit{Writable}, U : \textit{Writable})$
$\quad \textbf{requires}(\mathsf{ValueType}(T) = \mathsf{ValueType}(U))$
$\mathsf{write\_aliased} : T \times U$
$\quad (x, y) \mapsto \mathsf{sink}(x) \text{ is defined} \wedge \mathsf{sink}(y) \text{ is defined} \wedge$
$\quad\quad\quad (\forall V \in \textit{Readable})\,(\forall v \in V)\,\mathsf{aliased}(x, v) \Leftrightarrow \mathsf{aliased}(y, v)$

$\textbf{property}(O_0 : \textit{Writable}, O_1 : \textit{Writable})$
$\quad \textbf{requires}(\mathsf{Iterator}(O_0) \wedge \mathsf{Iterator}(O_1))$
$\mathsf{not\_write\_overlapped} : O_0 \times O_0 \times O_1 \times O_1$
$\quad (f_0, l_0, f_1, l_1) \mapsto$
$\quad\quad \mathsf{writable\_bounded\_range}(f_0, l_0) \wedge$
$\quad\quad \mathsf{writable\_bounded\_range}(f_1, l_1) \wedge$
$\quad\quad (\forall k_0 \in [f_0, l_0))(\forall k_1 \in [f_1, l_1))\,\neg\mathsf{write\_aliased}(k_0, k_1)$

$\textbf{property}(I : \textit{Readable}, O : \textit{Writable}, N : \textit{Integer})$
$\quad \textbf{requires}(\textit{Iterator}(I) \wedge \textit{Iterator}(O))$
$\mathsf{backward\_offset} : I \times I \times O \times O \times N$
$\quad (f_i, l_i, f_o, l_o, n) \mapsto$
$\quad\quad \mathsf{readable\_bounded\_range}(f_i, l_i) \wedge$
$\quad\quad n \geqslant 0 \wedge$
$\quad\quad \mathsf{writable\_bounded\_range}(f_o, l_o) \wedge$
$\quad\quad (\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o))$
$\quad\quad\quad \mathsf{aliased}(k_o, k_i) \Rightarrow k_i - f_i + n \leqslant k_o - f_o$

$\textbf{property}(I : \textit{Readable}, O : \textit{Writable}, N : \textit{Integer})$
$\quad \textbf{requires}(\textit{Iterator}(I) \wedge \textit{Iterator}(O))$
$\mathsf{forward\_offset} : I \times I \times O \times O \times N$
$\quad (f_i, l_i, f_o, l_o, n) \mapsto$
$\quad\quad \mathsf{readable\_bounded\_range}(f_i, l_i) \wedge$
$\quad\quad n \geqslant 0 \wedge$
$\quad\quad \mathsf{writable\_bounded\_range}(f_o, l_o) \wedge$
$\quad\quad (\forall k_i \in [f_i, l_i))(\forall k_o \in [f_o, l_o))$
$\quad\quad\quad \mathsf{aliased}(k_o, k_i) \Rightarrow l_i - k_i + n \leqslant l_o - k_o$

Chapter 10: Rearrangements

Chapter 11: Partition and Merging

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

Chapter 12: Composite Objects

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

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

Index