

Concept definitions from *Elements of Programming*

Alexander Stepanov Paul McJones

May 16, 2018

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

Regular(T) \triangleq

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

FunctionalProcedure(F) \triangleq

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

UnaryFunction(F) \triangleq

FunctionalProcedure(F)

\wedge *Arity*(F) = 1

\wedge *Domain* : *UnaryFunction* \rightarrow *Regular*

F \mapsto *InputType*(F, 0)

HomogeneousFunction(F) \triangleq

FunctionalProcedure(F)

\wedge *Arity*(F) > 0

\wedge $(\forall i, j \in \mathbb{N})(i, j < \text{Arity}(F)) \Rightarrow (\text{InputType}(F, i) = \text{InputType}(F, j))$

\wedge *Domain* : *HomogeneousFunction* \rightarrow *Regular*

F \mapsto *InputType*(F, 0)

property(F : *UnaryFunction*)

`regular_unary_function` : F

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

Chapter 2: Transformations and Their Orbits

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

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

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

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

$$\text{Transformation}(F) \triangleq \\ \text{Operation}(F) \\ \wedge \text{UnaryFunction}(F) \\ \wedge \text{DistanceType} : \text{Transformation} \rightarrow \text{Integer}$$

Chapter 3: Associative Operations

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

property(Op : BinaryOperation)

associative : Op

$$\text{op} \mapsto (\forall a, b, c \in \text{Domain}(\text{op})) \text{op}(\text{op}(a, b), c) = \text{op}(a, \text{op}(b, c))$$

$$\text{Integer}(I) \triangleq \\ \text{successor} : I \rightarrow I \\ \quad n \mapsto n + 1 \\ \wedge \text{predecessor} : I \rightarrow I \\ \quad n \mapsto n - 1 \\ \wedge \text{twice} : I \rightarrow I \\ \quad n \mapsto n + n \\ \wedge \text{half_nonnegative} : I \rightarrow I \\ \quad n \mapsto \lfloor n/2 \rfloor, \text{ where } n \geq 0$$

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

Chapter 4: Linear Orderings

$\text{Relation}(\mathbf{R}) \triangleq$
 $\text{HomogeneousPredicate}(\mathbf{R})$
 $\wedge \text{Arity}(\mathbf{R}) = 2$

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

property($\mathbf{R} : \text{Relation}$)
`strict` : \mathbf{R}
 $r \mapsto (\forall a \in \text{Domain}(\mathbf{R})) \neg r(a, a)$

property($\mathbf{R} : \text{Relation}$)
`reflexive` : \mathbf{R}
 $r \mapsto (\forall a \in \text{Domain}(\mathbf{R})) r(a, a)$

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

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

property($\mathbf{R} : \text{Relation}$)
`equivalence` : \mathbf{R}
 $r \mapsto \text{transitive}(r) \wedge \text{reflexive}(r) \wedge \text{symmetric}(r)$

property($F : \text{UnaryFunction}, R : \text{Relation}$)
requires($\text{Domain}(F) = \text{Domain}(R)$)
key_function : $F \times R$
 $(f, r) \mapsto (\forall a, b \in \text{Domain}(F)) (r(a, b) \Leftrightarrow f(a) = f(b))$

property($R : \text{Relation}$)
total_ordering : R
 $r \mapsto \text{transitive}(r) \wedge$
 $(\forall a, b \in \text{Domain}(R))$ exactly one of the following holds:
 $r(a, b), r(b, a),$ or $a = b$

property($R : \text{Relation}, E : \text{Relation}$) **requires**($\text{Domain}(R) = \text{Domain}(E)$)
weak_ordering : R
 $r \mapsto \text{transitive}(r) \wedge (\exists e \in E) \text{equivalence}(e) \wedge$
 $(\forall a, b \in \text{Domain}(R))$ exactly one of the following holds:
 $r(a, b), r(b, a),$ or $e(a, b)$

$\text{TotallyOrdered}(T) \triangleq$
 $\text{Regular}(T)$
 $\wedge < : T \times T \rightarrow \text{bool}$
 $\wedge \text{total_ordering}(<)$

Chapter 5: Ordered Algebraic Structures

property($T : \text{Regular}, \text{Op} : \text{BinaryOperation}$)
requires($T = \text{Domain}(\text{Op})$)
identity_element : $T \times \text{Op}$
 $(e, \text{op}) \mapsto (\forall a \in T) \text{op}(a, e) = \text{op}(e, a) = a$

property($F : \text{Transformation}, T : \text{Regular}, \text{Op} : \text{BinaryOperation}$)
requires($\text{Domain}(F) = T = \text{Domain}(\text{Op})$)
inverse_operation : $F \times T \times \text{Op}$
 $(\text{inv}, e, \text{op}) \mapsto (\forall a \in T) \text{op}(a, \text{inv}(a)) = \text{op}(\text{inv}(a), a) = e$

property($\text{Op} : \text{BinaryOperation}$)
commutative : Op
 $\text{op} \mapsto (\forall a, b \in \text{Domain}(\text{Op})) \text{op}(a, b) = \text{op}(b, a)$

$\text{AdditiveSemigroup}(T) \triangleq$
 $\text{Regular}(T)$
 $\wedge + : T \times T \rightarrow T$
 $\wedge \text{associative}(+)$
 $\wedge \text{commutative}(+)$

$\text{MultiplicativeSemigroup}(T) \triangleq$
 $\text{Regular}(T)$
 $\wedge \cdot : T \times T \rightarrow T$

$$\wedge \text{associative}(\cdot)$$

$$\text{AdditiveMonoid}(\mathbb{T}) \triangleq$$

$$\quad \text{AdditiveSemigroup}(\mathbb{T})$$

$$\wedge 0 \in \mathbb{T}$$

$$\wedge \text{identity_element}(0, +)$$

$$\text{MultiplicativeMonoid}(\mathbb{T}) \triangleq$$

$$\quad \text{MultiplicativeSemigroup}(\mathbb{T})$$

$$\wedge 1 \in \mathbb{T}$$

$$\wedge \text{identity_element}(1, \cdot)$$

$$\text{AdditiveGroup}(\mathbb{T}) \triangleq$$

$$\quad \text{AdditiveMonoid}(\mathbb{T})$$

$$\wedge - : \mathbb{T} \rightarrow \mathbb{T}$$

$$\wedge \text{inverse_operation}(\text{unary } -, 0, +)$$

$$\wedge - : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{T}$$

$$\quad (\mathbf{a}, \mathbf{b}) \mapsto \mathbf{a} + (-\mathbf{b})$$

$$\text{MultiplicativeGroup}(\mathbb{T}) \triangleq$$

$$\quad \text{MultiplicativeMonoid}(\mathbb{T})$$

$$\wedge \text{multiplicative_inverse} : \mathbb{T} \rightarrow \mathbb{T}$$

$$\wedge \text{inverse_operation}(\text{multiplicative_inverse}, 1, \cdot)$$

$$\wedge / : \mathbb{T} \times \mathbb{T} \rightarrow \mathbb{T}$$

$$\quad (\mathbf{a}, \mathbf{b}) \mapsto \mathbf{a} \cdot \text{multiplicative_inverse}(\mathbf{b})$$

$$\text{Semiring}(\mathbb{T}) \triangleq$$

$$\quad \text{AdditiveMonoid}(\mathbb{T})$$

$$\wedge \text{MultiplicativeMonoid}(\mathbb{T})$$

$$\wedge 0 \neq 1$$

$$\wedge (\forall \mathbf{a} \in \mathbb{T}) 0 \cdot \mathbf{a} = \mathbf{a} \cdot 0 = 0$$

$$\wedge (\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{T})$$

$$\quad \mathbf{a} \cdot (\mathbf{b} + \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \cdot \mathbf{c}$$

$$\quad \wedge (\mathbf{b} + \mathbf{c}) \cdot \mathbf{a} = \mathbf{b} \cdot \mathbf{a} + \mathbf{c} \cdot \mathbf{a}$$

$$\text{CommutativeSemiring}(\mathbb{T}) \triangleq$$

$$\quad \text{Semiring}(\mathbb{T})$$

$$\wedge \text{commutative}(\cdot)$$

$$\text{Ring}(\mathbb{T}) \triangleq$$

$$\quad \text{AdditiveGroup}(\mathbb{T})$$

$$\wedge \text{Semiring}(\mathbb{T})$$

$$\text{CommutativeRing}(\mathbb{T}) \triangleq$$

$$\quad \text{AdditiveGroup}(\mathbb{T})$$

$$\wedge \text{CommutativeSemiring}(\mathbb{T})$$

$$\text{Semimodule}(\mathbb{T}, \mathbb{S}) \triangleq$$

$$\quad \text{AdditiveMonoid}(\mathbb{T})$$

$$\wedge \text{CommutativeSemiring}(\mathbb{S})$$

```

 $\wedge \cdot : S \times T \rightarrow T$ 
 $\wedge (\forall \alpha, \beta \in S)(\forall a, b \in T)$ 
 $\quad \alpha \cdot (\beta \cdot a) = (\alpha \cdot \beta) \cdot a$ 
 $\quad (\alpha + \beta) \cdot a = \alpha \cdot a + \beta \cdot a$ 
 $\quad \alpha \cdot (a + b) = \alpha \cdot a + \alpha \cdot b$ 
 $\quad 1 \cdot a = a$ 

Module(T, S)  $\triangleq$ 
    Semimodule(T, S)
 $\wedge$  AdditiveGroup(T)
 $\wedge$  Ring(S)

OrderedAdditiveSemigroup(T)  $\triangleq$ 
    AdditiveSemigroup(T)
 $\wedge$  TotallyOrdered(T)
 $\wedge (\forall a, b, c \in T) a < b \Rightarrow a + c < b + c$ 

OrderedAdditiveMonoid(T)  $\triangleq$ 
    OrderedAdditiveSemigroup(T)
 $\wedge$  AdditiveMonoid(T)

OrderedAdditiveGroup(T)  $\triangleq$ 
    OrderedAdditiveMonoid(T)
 $\wedge$  AdditiveGroup(T)

CancellableMonoid(T)  $\triangleq$ 
    OrderedAdditiveMonoid(T)
 $\wedge - : T \times T \rightarrow T$ 
 $\wedge (\forall a, b \in T) b \leq a \Rightarrow a - b$  is defined  $\wedge (a - b) + b = a$ 

template<typename T>
    requires(CancellableMonoid(T))
T slow_remainder(T a, T b)
{
    // Precondition:  $a \geq 0 \wedge b > 0$ 
    while (b <= a) a = a - b;
    return a;
}

ArchimedeanMonoid(T)  $\triangleq$ 
    CancellableMonoid(T)
 $\wedge (\forall a, b \in T) (a \geq 0 \wedge b > 0) \Rightarrow$  slow_remainder(a, b) terminates
 $\wedge$  QuotientType : ArchimedeanMonoid  $\rightarrow$  Integer

HalvableMonoid(T)  $\triangleq$ 
    ArchimedeanMonoid(T)
 $\wedge$  half : T  $\rightarrow$  T
 $\wedge (\forall a, b \in T) (b > 0 \wedge a = b + b) \Rightarrow$  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;
    }
}

```

$EuclideanMonoid(T) \triangleq$
 $ArchimedeanMonoid(T)$
 $\wedge (\forall a, b \in T) (a > 0 \wedge b > 0) \Rightarrow \text{subtractive_gcd_nonzero}(a, b) \text{ terminates}$

$EuclideanSemiring(T) \triangleq$
 $CommutativeSemiring(T)$
 $\wedge \text{NormType} : EuclideanSemiring \rightarrow Integer$
 $\wedge w : T \rightarrow \text{NormType}(T)$
 $\wedge (\forall a \in T) w(a) \geq 0$
 $\wedge (\forall a \in T) w(a) = 0 \Leftrightarrow a = 0$
 $\wedge (\forall a, b \in T) b \neq 0 \Rightarrow w(a \cdot b) \geq w(a)$
 $\wedge \text{remainder} : T \times T \rightarrow T$
 $\wedge \text{quotient} : T \times T \rightarrow T$
 $\wedge (\forall a, b \in T) b \neq 0 \Rightarrow a = \text{quotient}(a, b) \cdot b + \text{remainder}(a, b)$
 $\wedge (\forall a, b \in T) b \neq 0 \Rightarrow w(\text{remainder}(a, b)) < w(b)$

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

```

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);
    }
}

```

$$\begin{aligned}
& \text{ArchimedeanGroup}(\mathbb{T}) \triangleq \\
& \quad \text{ArchimedeanMonoid}(\mathbb{T}) \\
& \quad \wedge \text{AdditiveGroup}(\mathbb{T}) \\
& \text{DiscreteArchimedeanSemiring}(\mathbb{T}) \triangleq \\
& \quad \text{CommutativeSemiring}(\mathbb{T}) \\
& \quad \wedge \text{ArchimedeanMonoid}(\mathbb{T}) \\
& \quad \wedge (\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbb{T}) \mathbf{a} < \mathbf{b} \wedge 0 < \mathbf{c} \Rightarrow \mathbf{a} \cdot \mathbf{c} < \mathbf{b} \cdot \mathbf{c} \\
& \quad \wedge \neg(\exists \mathbf{a} \in \mathbb{T}) 0 < \mathbf{a} < 1 \\
& \text{NonnegativeDiscreteArchimedeanSemiring}(\mathbb{T}) \triangleq \\
& \quad \text{DiscreteArchimedeanSemiring}(\mathbb{T}) \\
& \quad \wedge (\forall \mathbf{a} \in \mathbb{T}) 0 \leq \mathbf{a} \\
& \text{DiscreteArchimedeanRing}(\mathbb{T}) \triangleq \\
& \quad \text{DiscreteArchimedeanSemiring}(\mathbb{T}) \\
& \quad \wedge \text{AdditiveGroup}(\mathbb{T})
\end{aligned}$$

Chapter 6: Iterators

$$\begin{aligned}
& \text{Readable}(\mathbb{T}) \triangleq \\
& \quad \text{Regular}(\mathbb{T}) \\
& \quad \wedge \text{ValueType} : \text{Readable} \rightarrow \text{Regular} \\
& \quad \wedge \text{source} : \mathbb{T} \rightarrow \text{ValueType}(\mathbb{T}) \\
& \text{Iterator}(\mathbb{T}) \triangleq \\
& \quad \text{Regular}(\mathbb{T}) \\
& \quad \wedge \text{DistanceType} : \text{Iterator} \rightarrow \text{Integer} \\
& \quad \wedge \text{successor} : \mathbb{T} \rightarrow \mathbb{T} \\
& \quad \wedge \text{successor is not necessarily regular}
\end{aligned}$$

property($I : \text{Iterator}$)
 $\text{weak_range} : I \times \text{DistanceType}(I)$
 $(f, n) \mapsto (\forall i \in \text{DistanceType}(I))$
 $(0 \leq i \leq n) \Rightarrow \text{successor}^i(f) \text{ is defined}$

property($I : \text{Iterator}$)
 $\text{counted_range} : I \times \text{DistanceType}(I)$
 $(f, n) \mapsto \text{weak_range}(f, n) \wedge$
 $(\forall i, j \in \text{DistanceType}(I)) (0 \leq i < j \leq n) \Rightarrow$
 $\text{successor}^i(f) \neq \text{successor}^j(f)$

property($I : \text{Iterator}$)
 $\text{bounded_range} : I \times I$
 $(f, l) \mapsto (\exists k \in \text{DistanceType}(I)) \text{counted_range}(f, k) \wedge \text{successor}^k(f) = l$

property($I : \text{Readable}$)
requires($\text{Iterator}(I)$)

readable_bounded_range : $I \times I$
 $(f, l) \mapsto \text{bounded_range}(f, l) \wedge (\forall i \in [f, l]) \text{ source}(i) \text{ is defined}$

property(Op : *BinaryOperation*)

partially_associative : Op

op $\mapsto (\forall a, b, c \in \text{Domain}(\text{op}))$

If op(a, b) and op(b, c) are defined,
op(op(a, b), c) and op(a, op(b, c)) are defined
and are equal.

ForwardIterator(T) \triangleq

Iterator(T)

\wedge regular_unary_function(successor)

IndexedIterator(T) \triangleq

ForwardIterator(T)

$\wedge + : T \times \text{DistanceType}(T) \rightarrow T$

$\wedge - : T \times T \rightarrow \text{DistanceType}(T)$

$\wedge +$ takes constant time

$\wedge -$ takes constant time

BidirectionalIterator(T) \triangleq

ForwardIterator(T)

\wedge predecessor : $T \rightarrow T$

\wedge predecessor takes constant time

$\wedge (\forall i \in T) \text{ successor}(i) \text{ is defined} \Rightarrow$
predecessor(successor(i)) is defined and equals i

$\wedge (\forall i \in T) \text{ predecessor}(i) \text{ is defined} \Rightarrow$
successor(predecessor(i)) is defined and equals i

RandomAccessIterator(T) \triangleq

IndexedIterator(T) \wedge *BidirectionalIterator*(T)

\wedge *TotallyOrdered*(T)

$\wedge (\forall i, j \in T) i < j \Leftrightarrow i \prec j$

\wedge *DifferenceType* : *RandomAccessIterator* \rightarrow *Integer*

$\wedge + : T \times \text{DifferenceType}(T) \rightarrow T$

$\wedge - : T \times \text{DifferenceType}(T) \rightarrow T$

$\wedge - : T \times T \rightarrow \text{DifferenceType}(T)$

$\wedge <$ takes constant time

$\wedge -$ between an iterator and an integer takes constant time

Chapter 7: Coordinate Structures

BifurcateCoordinate(T) \triangleq

Regular(T)

\wedge *WeightType* : *BifurcateCoordinate* \rightarrow *Integer*

\wedge empty : $T \rightarrow \text{bool}$

```

    ^ has_left_successor : T → bool
    ^ has_right_successor : T → bool
    ^ left_successor : T → T
    ^ right_successor : T → T
    ^ (∀i, j ∈ T) (left_successor(i) = j ∨ right_successor(i) = j) ⇒ ¬empty(j)

property(C : BifurcateCoordinate)
tree : C
  x ↦ the descendants of x form a tree

BidirectionalBifurcateCoordinate(T) ≜
  BifurcateCoordinate(T)
  ^ has_predecessor : T → bool
  ^ (∀i ∈ T) ¬empty(i) ⇒ has_predecessor(i) is defined
  ^ predecessor : T → T
  ^ (∀i ∈ T) has_left_successor(i) ⇒
    predecessor(left_successor(i)) is defined and equals i
  ^ (∀i ∈ T) has_right_successor(i) ⇒
    predecessor(right_successor(i)) is defined and equals i
  ^ (∀i ∈ T) has_predecessor(i) ⇒
    is_left_successor(i) ∨ 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))
readable_tree : C
  x ↦ tree(x) ∧ (∀y ∈ C) reachable(x, y) ⇒ source(y) is defined

```

Chapter 8: Coordinates with Mutable Successors

$ForwardLinker(S) \triangleq$
 $IteratorType : ForwardLinker \rightarrow ForwardIterator$
 \wedge Let $I = IteratorType(S)$ in:
 $(\forall s \in S) (s : I \times I \rightarrow void)$
 $\wedge (\forall s \in S) (\forall i, j \in I)$ if $successor(i)$ is defined,
 then $s(i, j)$ establishes $successor(i) = j$

$BackwardLinker(S) \triangleq$
 $IteratorType : BackwardLinker \rightarrow BidirectionalIterator$
 \wedge Let $I = IteratorType(S)$ in:
 $(\forall s \in S) (s : I \times I \rightarrow void)$
 $\wedge (\forall s \in S) (\forall i, j \in I)$ if $predecessor(j)$ is defined,
 then $s(i, j)$ establishes $i = predecessor(j)$

$BidirectionalLinker(S) \triangleq ForwardLinker(S) \wedge BackwardLinker(S)$

property($I : Iterator$)
 $disjoint : I \times I \times I \times I$
 $(f0, l0, f1, l1) \mapsto (\forall i \in I) \neg(i \in [f0, l0] \wedge i \in [f1, l1])$

$LinkedBifurcateCoordinate(T) \triangleq$
 $BifurcateCoordinate(T)$
 \wedge $set_left_successor : T \times T \rightarrow void$
 $(i, j) \mapsto$ establishes $left_successor(i) = j$
 \wedge $set_right_successor : T \times T \rightarrow void$
 $(i, j) \mapsto$ establishes $right_successor(i) = j$

$EmptyLinkedBifurcateCoordinate(T) \triangleq$
 $LinkedBifurcateCoordinate(T)$
 \wedge $empty(T())^1$
 \wedge $\neg empty(i) \Rightarrow$
 $left_successor(i)$ and $right_successor(i)$ are defined
 \wedge $\neg empty(i) \Rightarrow$
 $(\neg has_left_successor(i) \Leftrightarrow empty(left_successor(i)))$
 \wedge $\neg empty(i) \Rightarrow$
 $(\neg has_right_successor(i) \Leftrightarrow empty(right_successor(i)))$

Chapter 9: Copying

$Writable(T) \triangleq$
 $ValueType : Writable \rightarrow Regular$
 $\wedge (\forall x \in T) (\forall v \in ValueType(T)) sink(x) \leftarrow v$ is a well-formed statement

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

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

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

property($I : \text{Writable}$)
requires($\text{Iterator}(I)$)
writable_bounded_range : $I \times I$
 $(f, l) \mapsto \text{bounded_range}(f, l) \wedge (\forall i \in [f, l]) \text{sink}(i) \text{ is defined}$

writable_weak_range and **writable_counted_range** are defined similarly.

property($I : \text{Mutable}$)
requires($\text{ForwardIterator}(I)$)
mutable_bounded_range : $I \times I$
 $(f, l) \mapsto \text{bounded_range}(f, l) \wedge (\forall i \in [f, l]) \text{sink}(i) \text{ is defined}$

mutable_weak_range and **mutable_counted_range** are defined similarly.

property($I : \text{Readable}, O : \text{Writable}$)
requires($\text{Iterator}(I) \wedge \text{Iterator}(O)$)
not_overlapped_forward : $I \times I \times O \times O$
 $(f_i, l_i, f_o, l_o) \mapsto$
 $\text{readable_bounded_range}(f_i, l_i) \wedge$
 $\text{writable_bounded_range}(f_o, l_o) \wedge$
 $(\forall k_i \in [f_i, l_i])(\forall k_o \in [f_o, l_o])$
 $\text{aliased}(k_o, k_i) \Rightarrow k_i - f_i \leq k_o - f_o$

property($I : \text{Readable}, O : \text{Writable}$)
requires($\text{Iterator}(I) \wedge \text{Iterator}(O)$)
not_overlapped_backward : $I \times I \times O \times O$
 $(f_i, l_i, f_o, l_o) \mapsto$
 $\text{readable_bounded_range}(f_i, l_i) \wedge$
 $\text{writable_bounded_range}(f_o, l_o) \wedge$
 $(\forall k_i \in [f_i, l_i])(\forall k_o \in [f_o, l_o])$
 $\text{aliased}(k_o, k_i) \Rightarrow l_i - k_i \leq l_o - k_o$

property($I : \text{Readable}, O : \text{Writable}$)
requires($\text{Iterator}(I) \wedge \text{Iterator}(O)$)
not_overlapped : $I \times I \times O \times O$

$(f_i, l_i, f_o, l_o) \mapsto$
 $\text{readable_bounded_range}(f_i, l_i) \wedge$
 $\text{writable_bounded_range}(f_o, l_o) \wedge$
 $(\forall k_i \in [f_i, l_i]) (\forall k_o \in [f_o, l_o]) \neg \text{aliased}(k_o, k_i)$

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

property($O_0 : \text{Writable}, O_1 : \text{Writable}$)
requires($\text{Iterator}(O_0) \wedge \text{Iterator}(O_1)$)
 $\text{not_write_overlapped} : O_0 \times O_0 \times O_1 \times O_1$
 $(f_0, l_0, f_1, l_1) \mapsto$
 $\text{writable_bounded_range}(f_0, l_0) \wedge$
 $\text{writable_bounded_range}(f_1, l_1) \wedge$
 $(\forall k_0 \in [f_0, l_0]) (\forall k_1 \in [f_1, l_1]) \neg \text{write_aliased}(k_0, k_1)$

property($I : \text{Readable}, O : \text{Writable}, N : \text{Integer}$)
requires($\text{Iterator}(I) \wedge \text{Iterator}(O)$)
 $\text{backward_offset} : I \times I \times O \times O \times N$
 $(f_i, l_i, f_o, l_o, n) \mapsto$
 $\text{readable_bounded_range}(f_i, l_i) \wedge$
 $n \geq 0 \wedge$
 $\text{writable_bounded_range}(f_o, l_o) \wedge$
 $(\forall k_i \in [f_i, l_i]) (\forall k_o \in [f_o, l_o])$
 $\text{aliased}(k_o, k_i) \Rightarrow k_i - f_i + n \leq k_o - f_o$

property($I : \text{Readable}, O : \text{Writable}, N : \text{Integer}$)
requires($\text{Iterator}(I) \wedge \text{Iterator}(O)$)
 $\text{forward_offset} : I \times I \times O \times O \times N$
 $(f_i, l_i, f_o, l_o, n) \mapsto$
 $\text{readable_bounded_range}(f_i, l_i) \wedge$
 $n \geq 0 \wedge$
 $\text{writable_bounded_range}(f_o, l_o) \wedge$
 $(\forall k_i \in [f_i, l_i]) (\forall k_o \in [f_o, l_o])$
 $\text{aliased}(k_o, k_i) \Rightarrow l_i - k_i + n \leq l_o - k_o$

Chapter 10: Rearrangements

Chapter 11: Partition and Merging

property($I : \text{ForwardIterator}, N : \text{Integer}, R : \text{Relation}$)
requires($\text{Mutable}(I) \wedge \text{ValueType}(I) = \text{Domain}(R)$)

```

mergeable : I × N × I × N × R
(f0, n0, f1, n1, r) ↦ f0 + n0 = f1 ∧
mutable_counted_range(f0, n0 + n1) ∧
weak_ordering(r) ∧
increasing_counted_range(f0, n0, r) ∧
increasing_counted_range(f1, n1, r)

```

Chapter 12: Composite Objects

```

Linearizable(W) ≜
  Regular(W)
  ∧ IteratorType : Linearizable → Iterator
  ∧ ValueType : Linearizable → Regular
    W ↦ ValueType(IteratorType(W))
  ∧ SizeType : Linearizable → Integer
    W ↦ DistanceType(IteratorType(W))
  ∧ begin : W → IteratorType(W)
  ∧ end : W → IteratorType(W)
  ∧ size : W → SizeType(W)
    x ↦ end(x) - begin(x)
  ∧ empty : W → bool
    x ↦ begin(x) = end(x)
  ∧ [] : W × SizeType(W) → ValueType(W)&
    (w, i) ↦ deref(begin(w) + i)

Sequence(S) ≜
  Linearizable(S)
  ∧ (∀s ∈ S) (∀i ∈ [begin(s), end(s)]) deref(i) is a part of s
  ∧ = : S × S → bool
    (s, s') ↦ lexicographical_equal(
      begin(s), end(s), begin(s'), end(s'))
  ∧ < : S × S → bool
    (s, s') ↦ lexicographical_less(
      begin(s), end(s), begin(s'), end(s'))

```

Index

- (product)
 - in multiplicative semigroup, 5
 - in semimodule, 6
- AdditiveGroup* concept, 5
- AdditiveMonoid* concept, 5
- AdditiveSemigroup* concept, 4
- algorithm
 - gcd, 8
 - is_left_successor, 10
 - is_right_successor, 10
 - slow_remainder, 6
 - subtractive_gcd_nonzero, 7
- aliased property, 12
- ArchimedeanGroup* concept, 8
- ArchimedeanMonoid* concept, 6
- associative operation, 9
- associative property, 2
 - partially_associative, 9
- asymmetric property, 3
- backward_offset property, 13
- BackwardLinker* concept, 11
- begin
 - for *Linearizable*, 14
- BidirectionalBifurcateCoordinate* concept, 10
- BidirectionalIterator* concept, 9
- BidirectionalLinker* concept, 11
- BifurcateCoordinate* concept, 10
- binary_scale_down_nonnegative, 3
- binary_scale_up_nonnegative, 3
- BinaryOperation* concept, 2
- bounded_range property, 9
- CancellableMonoid* concept, 6
- commutative property, 4
- CommutativeRing* concept, 5
- CommutativeSemiring* concept, 5
- concept
 - AdditiveGroup*, 5
 - AdditiveMonoid*, 5
 - AdditiveSemigroup*, 4
 - ArchimedeanGroup*, 8
 - ArchimedeanMonoid*, 6
 - BackwardLinker*, 11
 - BidirectionalBifurcateCoordinate*, 10
 - BidirectionalIterator*, 9
 - BidirectionalLinker*, 11
 - BifurcateCoordinate*, 10
 - BinaryOperation*, 2
 - CancellableMonoid*, 6
 - CommutativeRing*, 5
 - CommutativeSemiring*, 5
 - DiscreteArchimedeanRing*, 8
 - DiscreteArchimedeanSemiring*, 8
 - EmptyLinkedBifurcateCoordinate*, 11
 - EuclideanMonoid*, 7
 - EuclideanSemimodule*, 7
 - EuclideanSemiring*, 7
 - ForwardIterator*, 9
 - ForwardLinker*, 11
 - FunctionalProcedure*, 1
 - HalvableMonoid*, 7
 - HomogeneousFunction*, 1
 - HomogeneousPredicate*, 2
 - IndexedIterator*, 9
 - Integer*, 2
 - Iterator*, 8
 - Linearizable*, 14
 - LinkedBifurcateCoordinate*, 11
 - Module*, 6
 - MultiplicativeGroup*, 5
 - MultiplicativeMonoid*, 5
 - MultiplicativeSemigroup*, 5
 - Mutable*, 12
 - NonnegativeDiscreteArchimedeanSemiring*, 8
 - Operation*, 2
 - OrderedAdditiveGroup*, 6
 - OrderedAdditiveMonoid*, 6
 - OrderedAdditiveSemigroup*, 6
 - Predicate*, 2
 - RandomAccessIterator*, 9
 - Readable*, 8
 - Regular*, 1

- Relation*, 3
- Ring*, 5
- Semimodule*, 6
- Semiring*, 5
- Sequence*, 14
- TotallyOrdered*, 4
- Transformation*, 2
- UnaryFunction*, 1
- UnaryPredicate*, 2
- Writable*, 12
- counted_range property, 8
- deref, 12
- DifferenceType type function, 9
- DiscreteArchimedeanRing* concept, 8
- DiscreteArchimedeanSemiring* concept, 8
- disjoint property, 11
- DistanceType type function, 2, 8
- Domain type function, 1
- empty
 - for *Linearizable*, 14
- EmptyLinkedBifurcateCoordinate* concept, 11
- end
 - for *Linearizable*, 14
- equivalence property, 3
- EuclideanMonoid* concept, 7
- EuclideanSemimodule* concept, 7
- EuclideanSemiring* concept, 7
- even, 3
- forward_offset property, 13
- ForwardIterator* concept, 9
- ForwardLinker* concept, 11
- FunctionalProcedure* concept, 1
- gcd algorithm, 8
- half_nonnegative, 2
- HalvableMonoid* concept, 7
- HomogeneousFunction* concept, 1
- HomogeneousPredicate* concept, 2
- identity_element property, 4
- IndexedIterator* concept, 9
- Integer* concept, 2
- inverse_operation property, 4
- is_left_successor algorithm, 10
- is_right_successor algorithm, 10
- Iterator* concept, 8
- IteratorType type function, 11, 14
- Linearizable* concept, 14
- LinkedBifurcateCoordinate* concept, 11
- mergeable property, 14
- Module* concept, 6
- MultiplicativeGroup* concept, 5
- MultiplicativeMonoid* concept, 5
- MultiplicativeSemigroup* concept, 5
- Mutable* concept, 12
- mutable_bounded_range property, 12
- mutable_counted_range property, 12
- mutable_weak_range property, 12
- negative, 3
- NonnegativeDiscreteArchimedeanSemiring* concept, 8
- not_overlapped property, 13
- not_overlapped_backward property, 12
- not_overlapped_forward property, 12
- not_write_overlapped property, 13
- odd, 3
- one, 3
- Operation* concept, 2
- OrderedAdditiveGroup* concept, 6
- OrderedAdditiveMonoid* concept, 6
- OrderedAdditiveSemigroup* concept, 6
- partially_associative property, 9
- positive, 3
- predecessor
 - of integer, 2
 - of iterator, 9
- Predicate* concept, 2
- product (·)
 - in multiplicative semigroup, 5
 - in semimodule, 6
- property
 - aliased, 12
 - associative, 2

- asymmetric, 3
- backward_offset, 13
- bounded_range, 9
- commutative, 4
- counted_range, 8
- disjoint, 11
- equivalence, 3
- forward_offset, 13
- identity_element, 4
- inverse_operation, 4
- mergeable, 14
- mutable_bounded_range, 12
- mutable_counted_range, 12
- mutable_weak_range, 12
- not_overlapped, 13
- not_overlapped_backward, 12
- not_overlapped_forward, 12
- not_write_overlapped, 13
- partially_associative, 9
- readable_bounded_range, 9
- readable_tree, 11
- reflexive, 3
- regular_unary_function, 1
- strict, 3
- symmetric, 3
- total_ordering, 4
- transitive, 3
- tree, 10
- weak_ordering, 4
- weak_range, 8
- writable_bounded_range, 12
- writable_counted_range, 12
- writable_weak_range, 12
- write_alias, 13

quotient

- in Euclidean semimodule, 7
- in Euclidean semiring, 7

QuotientType type function, 6

RandomAccessIterator concept, 9

Readable concept, 8

readable_bounded_range property, 9

readable_tree property, 11

reflexive property, 3

Regular concept, 1

regular_unary_function property, 1

Relation concept, 3

remainder

- in Euclidean semimodule, 7
- in Euclidean semiring, 7

Ring concept, 5

Semimodule concept, 6

Semiring concept, 5

Sequence concept, 14

sink, 12

size

- for *Linearizable*, 14

SizeType type function, 14

slow_remainder algorithm, 6

source, 8

strict property, 3

subtractive_gcd_nonzero algorithm, 7

successor

- of integer, 2
- of iterator, 8

symmetric property, 3

total_ordering property, 4

TotallyOrdered concept, 4

Transformation concept, 2

transitive property, 3

tree property, 10

twice, 2

type function

- DifferenceType, 9
- DistanceType, 2, 8
- Domain, 1
- IteratorType, 11, 14
- QuotientType, 6
- SizeType, 14
- ValueType, 8, 12, 14
- WeightType, 10

UnaryFunction concept, 1

UnaryPredicate concept, 2

ValueType type function, 8, 12, 14

weak_ordering property, 4

weak_range property, 8

WeightType type function, 10

Writable concept, [12](#)
writable_bounded_range property, [12](#)
writable_counted_range property, [12](#)
writable_weak_range property, [12](#)
write_aliased property, [13](#)

zero, [3](#)