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

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

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.

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

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

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

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

property(F : *UnaryFunction*)
regular_unary_function : F

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

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

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

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

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

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

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

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

Chapter 5: Ordered Algebraic Structures

property(T : *Regular*, Op : *BinaryOperation*)
  requires(T = Domain(Op))
identity_element : T × Op
  $(e, op) \mapsto (\forall a \in T)\, op(a, e) = op(e, a) = a$

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

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

AdditiveSemigroup(T) $\triangleq$
    Regular(T)
  $\wedge$ $+ : T \times T \to T$
  $\wedge$ associative($+$)
  $\wedge$ commutative($+$)

MultiplicativeSemigroup(T) $\triangleq$
    Regular(T)
  $\wedge$ $\cdot : T \times T \to T$

$\wedge$ associative$(\cdot)$

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

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

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

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

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

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

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

$CommutativeRing(\mathsf{T}) \triangleq$
$\quad AdditiveGroup(\mathsf{T})$
$\wedge\ CommutativeSemiring(\mathsf{T})$

$Semimodule(\mathsf{T}, \mathsf{S}) \triangleq$
$\quad AdditiveMonoid(\mathsf{T})$
$\wedge\ CommutativeSemiring(\mathsf{S})$

$\wedge \ \cdot : \mathsf{S} \times \mathsf{T} \to \mathsf{T}$
$\wedge \ (\forall \alpha, \beta \in \mathsf{S})(\forall a, b \in \mathsf{T})$

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

Module$(\mathsf{T}, \mathsf{S}) \triangleq$
  Semimodule(T, S)
$\wedge$ *AdditiveGroup*(T)
$\wedge$ *Ring*(S)

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

OrderedAdditiveMonoid$(\mathsf{T}) \triangleq$
  OrderedAdditiveSemigroup(T)
$\wedge$ *AdditiveMonoid*(T)

OrderedAdditiveGroup$(\mathsf{T}) \triangleq$
  OrderedAdditiveMonoid(T)
$\wedge$ *AdditiveGroup*(T)

CancellableMonoid$(\mathsf{T}) \triangleq$
  OrderedAdditiveMonoid(T)
$\wedge \ - : \mathsf{T} \times \mathsf{T} \to \mathsf{T}$
$\wedge \ (\forall a, b \in \mathsf{T})\, b \leqslant a \Rightarrow a - b \text{ is defined} \wedge (a - b) + b = a$

```
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;
}
```

ArchimedeanMonoid$(\mathsf{T}) \triangleq$
  CancellableMonoid(T)
$\wedge \ (\forall a, b \in \mathsf{T})\, (a \geqslant 0 \wedge b > 0) \Rightarrow \mathsf{slow\_remainder}(a, b)$ terminates
$\wedge$ QuotientType : *ArchimedeanMonoid* $\to$ *Integer*

HalvableMonoid$(\mathsf{T}) \triangleq$
  ArchimedeanMonoid(T)
$\wedge$ 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);
    }
}
```

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

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

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

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

Chapter 6: Iterators

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

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

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

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

$$\begin{aligned}
&\textbf{property}(I : \textit{Iterator}) \\
&\mathsf{bounded_range} : I \times I \\
&\quad (f, l) \mapsto (\exists k \in \mathsf{DistanceType}(I))\ \mathsf{counted_range}(f, k) \wedge \mathsf{successor}^k(f) = l
\end{aligned}$$

$$\begin{aligned}
&\textbf{property}(I : \textit{Readable}) \\
&\quad \textbf{requires}(\mathsf{Iterator}(I))
\end{aligned}$$

$\mathsf{readable\_bounded\_range} : \mathrm{I} \times \mathrm{I}$
$\quad (\mathsf{f}, \mathsf{l}) \mapsto \mathsf{bounded\_range}(\mathsf{f}, \mathsf{l}) \wedge (\forall \mathsf{i} \in [\mathsf{f}, \mathsf{l}))\, \mathsf{source}(\mathsf{i})$ is defined

property$(\mathsf{Op} : \mathit{BinaryOperation})$
$\mathsf{partially\_associative} : \mathsf{Op}$
$\quad \mathsf{op} \mapsto (\forall \mathsf{a}, \mathsf{b}, \mathsf{c} \in \mathsf{Domain}(\mathsf{op}))$
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.

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

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

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

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

Chapter 7: Coordinate Structures

$\mathit{BifurcateCoordinate}(\mathsf{T}) \triangleq$
$\quad \mathit{Regular}(\mathsf{T})$
$\wedge\ \mathsf{WeightType} : \mathit{BifurcateCoordinate} \to \mathit{Integer}$
$\wedge\ \mathsf{empty} : \mathsf{T} \to \mathsf{bool}$

$\wedge$ has_left_successor : $\mathsf{T} \rightarrow$ bool
$\wedge$ has_right_successor : $\mathsf{T} \rightarrow$ bool
$\wedge$ left_successor : $\mathsf{T} \rightarrow \mathsf{T}$
$\wedge$ right_successor : $\mathsf{T} \rightarrow \mathsf{T}$
$\wedge$ $(\forall i, j \in \mathsf{T})$ (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(T) $\triangleq$
BifurcateCoordinate(T)
$\wedge$ has_predecessor : $\mathsf{T} \rightarrow$ bool
$\wedge$ $(\forall i \in \mathsf{T})\, \neg$empty$(i) \Rightarrow$ has_predecessor(i) is defined
$\wedge$ predecessor : $\mathsf{T} \rightarrow \mathsf{T}$
$\wedge$ $(\forall i \in \mathsf{T})$ 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))
readable_tree : C
$x \mapsto$ tree$(x) \wedge (\forall y \in \mathsf{C})$ reachable$(x, y) \Rightarrow$ source(y) is defined

Chapter 8: Coordinates with Mutable Successors

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

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

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

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

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

$\mathit{EmptyLinkedBifurcateCoordinate}(\mathsf{T}) \triangleq$
$\quad \mathit{LinkedBifurcateCoordinate}(\mathsf{T})$
$\wedge\ \mathsf{empty}(\mathsf{T}())$[1]
$\wedge\ \neg\mathsf{empty}(i) \Rightarrow$
$\quad\quad \mathsf{left\_successor}(i)$ and $\mathsf{right\_successor}(i)$ are defined
$\wedge\ \neg\mathsf{empty}(i) \Rightarrow$
$\quad\quad (\neg\mathsf{has\_left\_successor}(i) \Leftrightarrow \mathsf{empty}(\mathsf{left\_successor}(i)))$
$\wedge\ \neg\mathsf{empty}(i) \Rightarrow$
$\quad\quad (\neg\mathsf{has\_right\_successor}(i) \Leftrightarrow \mathsf{empty}(\mathsf{right\_successor}(i)))$

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

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

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

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

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

$\mathsf{writable\_weak\_range}$ and $\mathsf{writable\_counted\_range}$ are defined similarly.

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

$\mathsf{mutable\_weak\_range}$ and $\mathsf{mutable\_counted\_range}$ are defined similarly.

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

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

$\textbf{property}(I : \mathit{Readable}, \mathsf{O} : \mathit{Writable})$
$\quad \textbf{requires}(\mathsf{Iterator}(I) \wedge \mathsf{Iterator}(\mathsf{O}))$
$\mathsf{not\_overlapped} : I \times I \times \mathsf{O} \times \mathsf{O}$

$(f_i, l_i, f_o, l_o) \mapsto$
$\mathsf{readable\_bounded\_range}(f_i, l_i) \wedge$
$\mathsf{writable\_bounded\_range}(f_o, l_o) \wedge$
$(\forall k_i \in [f_i, l_i))\,(\forall k_o \in [f_o, l_o))\,\neg\mathsf{aliased}(k_o, k_i)$

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

property$(\mathsf{O}_0 : \mathit{Writable}, \mathsf{O}_1 : \mathit{Writable})$
requires$(\mathsf{Iterator}(\mathsf{O}_0) \wedge \mathsf{Iterator}(\mathsf{O}_1))$
not_write_overlapped : $\mathsf{O}_0 \times \mathsf{O}_0 \times \mathsf{O}_1 \times \mathsf{O}_1$
$(f_0, l_0, f_1, l_1) \mapsto$
$\mathsf{writable\_bounded\_range}(f_0, l_0) \wedge$
$\mathsf{writable\_bounded\_range}(f_1, l_1) \wedge$
$(\forall k_0 \in [f_0, l_0))(\forall k_1 \in [f_1, l_1))\,\neg\mathsf{write\_aliased}(k_0, k_1)$

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

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

Chapter 10: Rearrangements

Chapter 11: Partition and Merging

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

$$\begin{aligned}
&\mathsf{mergeable} : \mathrm{I} \times \mathsf{N} \times \mathrm{I} \times \mathsf{N} \times \mathsf{R} \\
&\quad (f_0, n_0, f_1, n_1, r) \mapsto f_0 + n_0 = f_1 \wedge \\
&\qquad \mathsf{mutable_counted_range}(f_0, n_0 + n_1) \wedge \\
&\qquad \mathsf{weak_ordering}(r) \wedge \\
&\qquad \mathsf{increasing_counted_range}(f_0, n_0, r) \wedge \\
&\qquad \mathsf{increasing_counted_range}(f_1, n_1, r)
\end{aligned}$$

Chapter 12: Composite Objects

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

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

Index