Notational Conventions

Notational Conventions - Ranges and Arrays

Before proceeding, we introduce some helpful notation for describing ranges and arrays. For natural numbers i, x and y:

$$\begin{array}{rcl} i \in (x..y) & \triangleq & x < i < y \\ i \in [x..y) & \triangleq & x \leq i < y \\ i \in (x..y] & \triangleq & x < i \leq y \\ i \in [x..y] & \triangleq & x \leq i \leq y \end{array}$$

This notation has a natural lifting to arrays. For an array a, value v and natural numbers x and y:

$$v \in \mathsf{a}[x..y) \triangleq \exists i \in [x..y).[\ 0 \leq i < \texttt{a.size} \wedge \mathsf{a}[i] = v\ ]$$

We can now elegantly describe properties of arrays, for example:

$$\mathsf{a}[x..y) \leq z \triangleq \forall v \in \mathsf{a}[x..y).[\ v \leq z\ ]$$

Note: we usually prefer closed-open intervals, e.g. $[x..y)$ or $\mathsf{a}[x..y)$.

Some more examples of array properties that we are able to define with the above notation are:

- $\mathsf{a}(x..y] \geq z \triangleq \forall v \in \mathsf{a}(x..y].[\ v \geq z\ ]$
- $\mathsf{a}[x..y] \neq z \triangleq \forall v \in \mathsf{a}[x..y].[\ v \neq z\ ]$

We will see later that using closed-open intervals for array slices is often convenient for our reasoning.

Notational Conventions - Array Slices

We refer to the closed-open interval $\mathtt{a}[x..y)$ as an **array slice**.

For an array `a` and natural numbers x and y, we can formally define an array slice inductively as follows:

$$\mathtt{a}[x..y) \triangleq \begin{cases} \mathtt{[]} & \text{if } y \leq x \\ \mathtt{a}[x{+}1..y) & \text{if } x < 0 \text{ or } x \geq \mathtt{a.size} \\ \mathtt{a}[x] : \mathtt{a}[x{+}1..y) & \text{otherwise} \end{cases}$$

Notice how the interval's values are truncated to the bounds of the array `a`.

Although we rarely use them, we can define the three other types of array intervals (to match the other number ranges) in terms of our array slice notation. Specifically:

- $\mathtt{a}(x..y) \triangleq \mathtt{a}[x{+}1..y)$
- $\mathtt{a}(x..y] \triangleq \mathtt{a}[x{+}1..y{+}1)$
- $\mathtt{a}[x..y] \triangleq \mathtt{a}[x..y{+}1)$

Notational Conventions - Arrays and their Contents

To simplify our notation, we drop the interval bounds when they refer to the start or end of an array. Thus, for an array a and natural number i:

$\mathtt{a}[..i)$	describes the array slice from 0 up to, but not including, i
$\mathtt{a}[i..)$	describes the array slice from i up to its end
$\mathtt{a}[..)$	describes the whole array

We also need to be able to distinguish the reference to an array from the array's contents when making $_{-old}$ or $_{-pre}$ annotations.

$\mathtt{a}[..)_{old}$ or $\mathtt{a}[..)_{pre}$ refer to previous values of an array's **contents**

We will often need to use the array contents annotations $a[..)_{old}$ or $a[..)_{pre}$, as it is very common for methods to update (or potentially update) the contents of arrays.

Notational Conventions - Array Shorthands

We introduce several notational shorthands for arrays.
For arbitrary arrays a and b, and value v:

- $NrOccurs(\mathtt{a}[..), v) \triangleq |\{ k \mid \mathtt{a}[k] = v \}|$
- $\mathtt{a}[..) \sim \mathtt{b}[..)$ means that the contents of array a are a permutation of the contents of array b:

$$\mathtt{a}[..) \sim \mathtt{b}[..) \triangleq \forall v.[\ NrOccurs(\mathtt{a}[..), v) = NrOccurs(\mathtt{b}[..), v)\ ]$$

- $\mathtt{a}[..) \approx \mathtt{b}[..)$ means that the contents of arrays a and b are identical:

$$\mathtt{a}[..) \approx \mathtt{b}[..) \triangleq \mathtt{a.size} = \mathtt{b.size} \wedge \forall j \in [0..\mathtt{a.size}).[\ \mathtt{a[j]} = \mathtt{b[j]}\ ]$$

We will often refer to $\approx$ as **deep equality**.

For a set S, the term |S| denotes the cardinality (or size) of the set. So, NrOccurs counts the number of occurrences of a value v in an array. The definition of ~ then requires that both arrays contain the same number of occurrences of each possible value. The definitions of ~ and ≈ can also be applied to arbitrary array

slices, in which case they only consider their respective array contents in the given intervals. e.g.

$a[1..a.size) \approx b[1..b.size)$

describes two arrays a and b that are identical, except for their first element.

Notational Conventions - Array Operations

We also introduce some notational shorthands for operations on arrays. For an array a and natural numbers x and y:

- $\sum \mathsf{a}[x..y)$ is the sum of the elements of array a from index x up to but not including index y:

$$\begin{array}{rcl} \sum \mathsf{a}[x..y) & \triangleq & \sum_{k=x}^{y-1} \mathsf{a}[k] \\ & \triangleq & \mathsf{a}[x] + \mathsf{a}[x+1] + ... + \mathsf{a}[y-1] \end{array}$$

- $\Pi \mathsf{a}[x..y)$ is the product of the elements of array a from index x up to but not including index y:

$$\begin{array}{rcl} \Pi\, \mathsf{a}[x..y) & \triangleq & \Pi_{k=x}^{y-1} \mathsf{a}[k] \\ & \triangleq & \mathsf{a}[x] * \mathsf{a}[x+1] * ... * \mathsf{a}[y-1] \end{array}$$

Important: $\sum \mathsf{a}[x..y)$ and $\Pi \mathsf{a}[x..y)$ are only well-defined for ranges $[x..y)$ that lie within the bounds of array a.

Note that the sum of an empty array slice (e.g.$sum(a[.\,.0))$) is defined to be 0 (the unit of addition).
Similarly, the product of an empty array slice (e.g.$product(a[a.size..))$) is defined to be 1 (the unit of multiplication).

Notational Conventions - Helpful Predicates

We introduce some predicates which will be helpful in upcoming examples. For arbitrary arrays a and b and natural numbers i and j:

- $Sorted(\,\mathtt{a}[..]\,)$ means that array a is ordered:
  $Sorted(\,\mathtt{a}[..]\,) \triangleq \forall j,k \in [0..\mathtt{a.size}).[\; j \leq k \longrightarrow \mathtt{a}[j] \leq \mathtt{a}[k] \;]$

- $Swapped(\,\mathtt{a}[..],\mathtt{b}[..],i,j\,)$ means that the arrays a and b are identical, except for the elements at indexes i and j, which have been swapped:

$$Swapped(\,\mathtt{a}[..],\mathtt{b}[..],i,j\,) \triangleq \begin{array}{l} \mathtt{a.size} = \mathtt{b.size} \\ \quad \wedge\; i,j \in [0..\mathtt{a.size}) \\ \quad \wedge\; \mathtt{b}[i] = \mathtt{a}[j] \;\wedge\; \mathtt{b}[j] = \mathtt{a}[i] \\ \quad \wedge\; \forall k \in [0..\mathtt{a.size}) \backslash \{i,j\}.[\; \mathtt{a}[k] = \mathtt{b}[k] \;] \end{array}$$

Important: $Sorted$ is only well-defined for an array whose contents are comparable (such as Integers).

We can apply the definition of Sorted to an arbitrary array slice if desired:

- $Sorted(\,\mathtt{a}[x..y)\,) \triangleq \forall j,k \in [x..y).[\; 0 \leq j \leq k < \mathtt{a.size} \longrightarrow \mathtt{a[j]} \leq \mathtt{a[k]} \;]$

Notational Conventions - Helpful Functions

We also introduce some functions which will be helpful in upcoming examples. For an array a:

- $min(\,\mathtt{a}[..]\,)$ gives the smallest element in the array a:

$$min(\,\mathtt{a}[..]\,) \triangleq min\{\; z \mid \exists j \in [0..\mathtt{a.size}).\; \mathtt{a}[j] = z \;\}$$

- $max(\,\mathtt{a}[..]\,)$ gives the biggest element in the array a:

$$max(\,\mathtt{a}[..]\,) \triangleq max\{\; z \mid \exists j \in [0..\mathtt{a.size}).\; \mathtt{a}[j] = z \;\}$$

Important: min and max are only well-defined for an array whose contents are comparable (such as Integers).

We can apply these definitions to arbitrary array slices if desired.

- $min(\,\mathtt{a}[x..y)\,) \triangleq min\{\ z \mid \exists j \in [x..y).[\ 0 \leq j < \mathtt{a.size} \ \wedge\ \mathtt{a}[j] = z\ ]\ \}$
- $max(\,\mathtt{a}[x..y)\,) \triangleq max\{\ z \mid \exists j \in [x..y).[\ 0 \leq j < \mathtt{a.size} \ \wedge\ \mathtt{a}[j] = z\ ]\ \}$

Array Lemmas - Deep Equality and Permutation

For all arrays a and b:

Deep Equality Lemmas:

- $\mathtt{a}[..) \approx \mathtt{b}[..) \longrightarrow \mathtt{b}[..) \approx \mathtt{a}[..)$ **(≈Symm)**
- $\mathtt{a}[..) \approx \mathtt{b}[..) \ \wedge\ \mathtt{b}[..) \approx \mathtt{c}[..) \longrightarrow \mathtt{a}[..) \approx \mathtt{c}[..)$ **(≈Trans)**
- $\mathtt{a}[..) \approx \mathtt{b}[..) \longrightarrow \mathtt{a.siez} = \mathtt{b.size}$ **(≈Size)**
- $\mathtt{a}[..) \approx \mathtt{b}[..) \longrightarrow \mathtt{a}[..) \sim \mathtt{b}[..)$ **(≈IsPrm)**

Permutation Lemmas:

- $\mathtt{a}[..) \sim \mathtt{b}[..) \longrightarrow \mathtt{b}[..) \sim \mathtt{a}[..)$ **(∼Symm)**
- $\mathtt{a}[..) \sim \mathtt{b}[..) \ \wedge\ \mathtt{b}[..) \sim \mathtt{c}[..) \longrightarrow \mathtt{a}[..) \sim \mathtt{c}[..)$ **(∼Trans)**
- $\mathtt{a}[..) \sim \mathtt{b}[..) \longrightarrow \mathtt{a.size} = \mathtt{b.size}$ **(∼Size)**

Notational Conventions - Array Concatenation

We define the **concatenation** of arrays (:) via their deep equality relation with a third "observer" array. That is, for arbitrary arrays a, b and c we have:

$$\mathtt{a}[..) \approx \mathtt{b}[..) : \mathtt{c}[..) \quad \triangleq \quad \mathtt{a}[..\mathtt{b.size}) \approx \mathtt{b}[..) \ \wedge\ \mathtt{a}[\mathtt{b.size}..) \approx \mathtt{c}[..)$$

Notice that from the definition of ≈ we know that when $\mathtt{a}[..) \approx \mathtt{b}[..) : \mathtt{c}[..)$ then $\mathtt{a.size} = \mathtt{b.size} + \mathtt{c.size}$

Using deep equality, array slices and concatenation we can describe many interesting properties of arrays. For example:

$$\texttt{a}[..] \approx \texttt{a}[0..1) : \texttt{b}[..] : \texttt{a}[\texttt{b.size} + 1..\texttt{a.size})$$

says that the contents of the array `a` from index 1 to index `b.size` is the same as the array b. It does not say anything about the other elements of the array `a`. This notation can be very useful for describing changes to just part of an array, or even for specifying which parts of an array are unmodified.

Array Lemmas - Swapping and Ranges

For all arrays `a` and `b` and for all integers i, j, k, x and y:

Swapped Lemmas:

- $Swapped(\texttt{a}[..], \texttt{b}[..], i, i) \longrightarrow \texttt{a}[..] \approx \texttt{b}[..]$ **(≈Swpd)**
- $Swapped(\texttt{a}[..], \texttt{b}[..], i, j) \longrightarrow \texttt{a}[..] \sim \texttt{b}[..]$ **(∼Swpd)**
- $Swapped(\texttt{a}[..], \texttt{b}[..], i, j) \longleftrightarrow Swapped(\texttt{a}[..], \texttt{b}[..], j, i)$
  $\longleftrightarrow Swapped(\texttt{b}[..], \texttt{a}[..], i, j)$ **(SwpSm)**

Ranges Lemmas:

- $\texttt{b}[..) \approx \texttt{a}[0..i) : \texttt{b}[i..j) : \texttt{a}[j..) \ \wedge \ x \leq i \ \wedge \ j \leq y$
  $\longrightarrow \texttt{b}[..) \approx \texttt{a}[0..x) : \texttt{b}[x..y) : \texttt{a}[y..)$ **(RngWeak)**
- $\texttt{b}[..) \approx \texttt{a}[0..i) : \texttt{b}[i..j) : \texttt{a}[j..) \ \wedge \ \texttt{a}[i..j) \sim \texttt{b}[i..j)$
  $\longrightarrow \texttt{a}[..) \sim \texttt{b}[..)$ **(RngPrm)**
- $\texttt{a}[..) \approx \texttt{b}[..) \ \wedge \ \texttt{c}[..) \approx \texttt{c}[0..i) : \texttt{a}[i..j) : \texttt{c}[j..)$
  $\longrightarrow \texttt{c}[..) \approx \texttt{c}[0..i) : \texttt{b}[i..j) : \texttt{c}[j..)$ **(RngSwap)**

In `RngWeak` the premise says a and b are identical except for the range $[i..j)$, and the conclusion says that a and b are identical except for the range $[m..n)$. The lemma holds, since the range $[m..n)$ includes the range $[i..j)$