Making Program Logics Intelligible

John C. Reynolds
Carnegie Mellon University

Lovelace Lecture — June 8, 2011

Research partially supported by National Science Foundation Grant CCF-0916808

Dedication

To the British computer scientists who taught me so much when I was young.

Those who are gone:

Christopher Strachey   Peter Landin   Robin Milner

and those who continue to instruct me:

Tony Hoare   Rod Burstall   Alan Robinson

Introduction

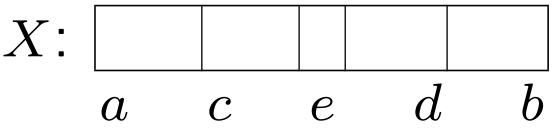
To verify program specifications, rather than generic safety properties, it will be necessary to integrate verification into the process of programming.

Program proving is unlike theorem proving in mathematics - mathematical conjectures may give no hint as to how they could be proved, but programs are written by programmers, who must understand informally why their programs work. The job of verification is not to explore some immense search space, but to formalize the programmer's intuitions until any faults are revealed.

This requires specifications and proofs that are succinct and intelligible - which in turn require logics that go beyond predicate calculus (the assembly language of program proving). In this talk, I will recount and illustrate several steps, old and new, towards this goal.

The Distance between Traditional Logic and Programming

Particularly when describing arrays, program specifications are typically full of inequalities and set definitions using inequalities. For example, a diagram that a programmer might write, e.g.,

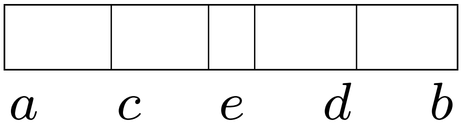


might be expressed by:

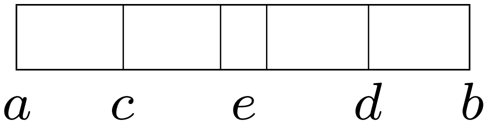
$$a \leq c \leq e \leq d \leq b \quad \text{and} \quad \mathsf{dom}\, X = \{\, i \mid a < i < b \,\}.$$

Partition Diagrams

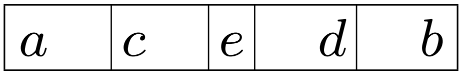
The obvious difficulty with diagrams such as



is that expressions tend to migrate slightly:



This fertile source of program errors can be avoided by capturing the expressions inside the boxes:



.

We call such a diagram a *partition diagram*.

Interval Diagrams

An *interval diagram* is an annotated box denoting a finite consecutive set of integers. In particular

$$\left.\begin{array}{c} a\,\boxed{\quad\ b} \\ \boxed{a\quad b} \\ a\,\boxed{\qquad}\,b \\ \boxed{a\qquad}\,b \\ \boxed{a} \end{array}\right\} \begin{array}{l}\text{denotes the set of} \\ \text{integers i such that}\end{array} \left\{\begin{array}{c} a < i \le b \\ a \le i \le b \\ a < i < b \\ a \le i < b \\ i = a. \end{array}\right.$$

Partition Diagrams

A diagram of the form

$$a_0 \begin{array}{|r|r|r|}\hline \qquad a_1 & \qquad a_2 & \qquad \cdots a_n \\ \hline \end{array}$$

is called a *partition diagram*. The intervals denoted by

$$a_0 \begin{array}{|r|}\hline \qquad a_1 \\ \hline \end{array}, a_1 \begin{array}{|r|}\hline \qquad a_2 \\ \hline \end{array}, \;\cdots\;, a_{n-1} \begin{array}{|r|}\hline \qquad a_n \\ \hline \end{array}$$

are called the *component intervals* of the partition diagram, and the interval denoted by $a_0 \begin{array}{|r|}\hline \qquad a_n \\ \hline \end{array}$ is called the *total interval* of the partition diagram.

A partition diagram is an assertion that its component intervals are a partition of its total interval, i.e., that the component intervals are disjoint and their union is the total interval.

Some Abbreviations

Just as with interval diagrams, we may write $|a$ instead of $a{-}1|$ and $\boxed{a}$ instead of $\boxed{a \quad a}$ within partition diagrams. For example,

$$\begin{array}{|l|c|r|}\hline a \quad & k & \quad b \\ \hline\end{array}$$

has the same meaning as

$$\begin{array}{|l|cc|r|}\hline a \quad & k & k & \quad b \\ \hline\end{array}\,,$$

which has the same meaning as

$$a{-}1\ \begin{array}{|r|r|r|}\hline \quad k{-}1 & \quad k & \quad b \\ \hline\end{array}\,.$$

Summing an Array (Annotated Specification)

```
$\{\boxed{a \quad b} \subseteq \mathsf{dom}\, X\}$
$\textbf{newvar}\ k\ \textbf{in}\ (k := a - 1\ ;\ s := 0\ ;$
$\{\textbf{inv}\text{:}\ \boxed{a \quad k}\!\boxed{\quad b} \wedge s = \sum_{i \in \boxed{a \quad k}} X(i)\}$
$\textbf{while}\ k < b\ \textbf{do}\ ($
   $k := k + 1\ ;$
   $\{\boxed{a \quad}\!\boxed{k}\!\boxed{\quad b} \wedge s = \sum_{i \in \boxed{a \quad}\, k} X(i)\}$
   $s := s + X(k)))$
$\{s = \sum_{i \in \boxed{a \quad b}} X(i)\}$
```

Array Values as Functions

We regard an array as a variable whose value is a function, whose domain is an interval. For example,

$$\textbf{real array } X(1:10)\ \cdots$$

declares X to be an array whose values have the domain $\boxed{1\quad 10}$.

When $S \subseteq \mathsf{dom}\, X$, we write $X\rceil S$ for the *restriction* of X to S, which is the function such that

$$\mathsf{dom}(X\rceil S) = S \qquad \forall i \in S.\ (X\rceil S)\, i = X\, i.$$

We also write $\{X\}$ for the *image* of X, i.e., the set

$$X = \{\, X\,i \mid i \in \mathsf{dom}\, X \,\}.$$

For example, suppose sq is the function with domain $\boxed{0 \quad 5}$ that maps each number between 0 and 5 into its square. Then

$$\{sq\} = \{0, 1, 4, 9, 16, 25\} \qquad \{sq\rceil\,\boxed{2 \quad 4}\,\} = \{4, 9, 16\}.$$

Pointwise Extension of a Relation

Suppose ρ is a binary relation between values. Then the pointwise extension of ρ, written ρ^*, is the relation between sets of such values such that

$$S \, \rho^* \, T \text{ iff } \forall x \in S, y \in T. \; x \, \rho \, y.$$

For example,

$$\{2, 3\} \leq^* \{3, 4\} \qquad \{2, 3\} \neq^* \{4, 5\}$$

are both true, but

$$\{2, 3\} <^* \{3, 4\} \qquad \{2, 3\} =^* \{2, 3\} \qquad \{2, 3\} \neq^* \{2, 3\}$$

are all false.

We also abbreviate

$$\{x\}\,\rho^*\,T \text{ by } x\,\rho^*\,T$$

$$S\,\rho^*\,\{y\} \text{ by } S\,\rho^*\,y.$$

Ordered Arrays

We define the assertion

$$\mathbf{ord}_\rho\, X \text{ iff } \forall i, j \in \mathsf{dom}\, X.\ i < j \text{ implies } X(i)\, \rho\, X(j).$$

For example, $\mathbf{ord}_\leq\, X$ asserts that X is ordered in nonstrict increasing order.

Binary Search

The following is the invariant of a simple while-loop program for binary search:

$$\begin{aligned}
&\boxed{a \quad b} \subseteq \mathsf{dom}\, X \wedge \mathbf{ord}_{\le}(X \rceil \boxed{a \quad b}) \wedge \\
&\begin{array}{|l|c|r|}\hline a \quad & c \quad d & \quad b \\ \hline\end{array} \wedge \\
&\{X \rceil \boxed{a \quad\;} c\} \le^* y \wedge y <^* \{X \rceil d \boxed{\quad\; b}\}.
\end{aligned}$$

Compare this with

$$\begin{aligned}
&(\forall i.\ a \le i \le b \text{ implies } i \in \mathsf{dom}\, X) \wedge \\
&(\forall i, j.\ a \le i < j \le b \text{ implies } X(i) \le X(j)) \wedge \\
&a - 1 \le c - 1 \le d \le b \wedge \\
&(\forall i.\ a \le i < c \text{ implies } X(i) \le y) \wedge \\
&(\forall i.\ d < i \le b \text{ implies } y < X(i)).
\end{aligned}$$

Partition: The Invariants

$$\begin{aligned}
I = \quad & \begin{array}{|l|r|}\hline m \quad & i \quad n \\ \hline\end{array} \wedge \{A\rceil \begin{array}{|l|}\hline m \quad\quad \\ \hline\end{array} i\} \leq^* r \wedge \\
& \begin{array}{|r|r|}\hline m \quad j & \quad n \\ \hline\end{array} \wedge r \leq^* \{A\rceil j \begin{array}{|r|}\hline \quad n \\ \hline\end{array} \} \wedge \\
& (i \leq j \text{ implies } (\exists p.\ \begin{array}{|l|l|r|}\hline i \quad & p & \quad n \\ \hline\end{array} \wedge r \leq A(p)) \wedge \\
& \qquad (\exists q.\ \begin{array}{|l|l|r|}\hline m \quad & q & \quad j \\ \hline\end{array} \wedge A(q) \leq r))
\end{aligned}$$

$$\begin{aligned}
I_1 = \quad & \begin{array}{|l|r|}\hline m \quad & i \quad n \\ \hline\end{array} \wedge \{A\rceil \begin{array}{|l|}\hline m \quad\quad \\ \hline\end{array} i\} \leq^* r \wedge \\
& \qquad (\exists p.\ \begin{array}{|l|l|r|}\hline i \quad & p & \quad n \\ \hline\end{array} \wedge r \leq A(p))
\end{aligned}$$

$$\begin{aligned}
I_2 = \quad & \begin{array}{|r|r|}\hline m \quad j & \quad n \\ \hline\end{array} \wedge r \leq^* \{A\rceil j \begin{array}{|r|}\hline \quad n \\ \hline\end{array} \} \wedge \\
& \qquad (\exists q.\ \begin{array}{|l|l|r|}\hline m \quad & q & \quad j \\ \hline\end{array} \wedge A(q) \leq r)
\end{aligned}$$

$$\begin{aligned}
I' = \quad & \begin{array}{|r|r|}\hline m \quad i & \quad n \\ \hline\end{array} \wedge \{A\rceil \begin{array}{|r|}\hline m \quad i \\ \hline\end{array} \} \leq^* r \wedge \\
& \begin{array}{|l|l|}\hline m \quad & j \quad n \\ \hline\end{array} \wedge r \leq^* \{A\rceil \begin{array}{|l|}\hline j \quad n \\ \hline\end{array} \} \wedge \\
& (i{+}1 \leq j{-}1 \text{ implies } (\exists p.\ i \begin{array}{|l|l|r|}\hline \quad & p & \quad n \\ \hline\end{array} \wedge r \leq A(p)) \wedge \\
& \qquad (\exists q.\ \begin{array}{|l|l|l|}\hline m \quad & q & \quad \\ \hline\end{array} j \wedge A(q) \leq r))
\end{aligned}$$

Partition: The Procedure

```
let Partition$(A, i, j\,; m, n) =$
  $\{\boxed{m \quad n} \subseteq \mathsf{dom}\, A \wedge \boxed{m}\boxed{\quad n}\}$
  newvar $r$ in (
    $r := A((m + n) \div 2)\,;\, i := m\,;\, j := n\,;$
    $\{$inv: $I\}$
    while $i \le j$ do (
      $\{I_1 \wedge I_2\}$
      while $A(i) < r$ do $i := i + 1\,;$
      while $r < A(j)$ do $j := j - 1\,;$
      $\{I_1 \wedge r \le A(i) \wedge I_2 \wedge A(j) \le r\}$
      if $i \le j$ then (
        newvar $t$ in $(t{:=}A(i)\,;\, A(i){:=}A(j)\,;\, A(j){:=}t)\,;$
        $\{I_1 \wedge A(i) \le r \wedge I_2 \wedge r \le A(j) \wedge i \le j\}$
        $\{I'\}$
        $i := i + 1\,;\, j := j - 1)))$
  $\{\boxed{m \quad j}\boxed{\quad}\boxed{i \quad n} \wedge \{A\rceil\,\boxed{m \quad}\,i\} \le^* \{A\rceil j\,\boxed{\quad n}\}\}$
```

Rearrangement

We write $X \sim Y$, and say that X is a *rearrangement* of Y, when there is a bijection (i.e., a one-to-one correspondence) B from $\mathsf{dom}\, X$ to $\mathsf{dom}\, Y$ such that

$$\forall i \in \mathsf{dom}\, X.\ X(i) = Y(B(i)).$$

Suppose $S \subseteq \mathsf{dom}\, X = \mathsf{dom}\, Y$. We write $X \sim_S Y$, and say that X is a *rearrangement restricted to* S of Y, when X is a rearrangement of Y and, for all $i \in (\mathsf{dom}\, X) - S$, $X(i) = Y(i)$.

Partition Revisited

```
let Partition$(A, i, j\ ;\ m, n) =$
  $\{\boxed{m\ \ n} \subseteq \mathrm{dom}\, A \wedge \boxed{m}\boxed{\ \ \ n} \wedge A = A_0\}$
  newvar $r$ in (
   $r := A((m + n) \div 2)\ ;\ i := m\ ;\ j := n\ ;$
   $\{$inv: $I \wedge A \sim_{\boxed{m\ n}} A_0\}$
   while $i \leq j$ do (
     $\{I_1 \wedge I_2 \wedge A \sim_{\boxed{m\ n}} A_0\}$
     while $A(i) < r$ do $i := i + 1\ ;$
     while $r < A(j)$ do $j := j - 1\ ;$
     $\{I_1 \wedge r \leq A(i) \wedge I_2 \wedge A(j) \leq r \wedge A \sim_{\boxed{m\ n}} A_0\}$
     if $i \leq j$ then (
       newvar $t$ in $(t{:=}A(i)\ ;\ A(i){:=}A(j)\ ;\ A(j){:=}t)\ ;$
       $\{I_1 \wedge A(i) \leq r \wedge I_2 \wedge r \leq A(j) \wedge i \leq j \wedge A \sim_{\boxed{m\ n}} A_0\}$
       $\{I' \wedge A \sim_{\boxed{m\ n}} A_0\}$
       $i := i + 1\ ;\ j := j - 1)))$
  $\{\boxed{m\ \ j}\boxed{\ \ }\boxed{i\ \ n} \wedge \{A\rceil\boxed{m\ \ \ }\,i\} \leq^* \{A\rceil j\boxed{\ \ n}\} \wedge A \sim_{\boxed{m\ n}} A_0\}$
```

Quicksort

```
$\mathbf{letrec}\ \mathsf{Quicksort}(A\ ;\ m, n) =$
  $\{\boxed{m \quad n} \subseteq \mathsf{dom}\, A \wedge A = A_0\}$
  $\mathbf{if}\ m < n\ \mathbf{then}\ \mathbf{newvar}\ i, j\ \mathbf{in}\ \big($
    $\{\boxed{m \quad n} \subseteq \mathsf{dom}\, A \wedge \boxed{m}\boxed{\quad}\boxed{n}\}$
    $\mathsf{Partition}(A, i, j\ ;\ m, n)\ ;$
    $\{\boxed{m \quad j}\boxed{\quad}\boxed{i \quad n} \wedge \{A\rceil \boxed{m \quad} i\} \leq^* \{A\rceil j \boxed{\quad n}\} \wedge A {\sim}_{\boxed{m\, n}} A_0\}$
    $\mathsf{Quicksort}(A\ ;\ m, j)\ ;$
    $\{\boxed{m \quad j}\boxed{\quad}\boxed{i \quad n} \wedge \mathbf{ord}_{\leq} A\rceil \boxed{m \quad j} \wedge$
                        $\{A\rceil \boxed{m \quad} i\} \leq^* \{A\rceil j \boxed{\quad n}\} \wedge A {\sim}_{\boxed{m\, n}} A_0\}$
    $\mathsf{Quicksort}(A\ ;\ i, n)$
    $\{\boxed{m \quad j}\boxed{\quad}\boxed{i \quad n} \wedge \mathbf{ord}_{\leq} A\rceil \boxed{m \quad j} \wedge \mathbf{ord}_{\leq} A\rceil \boxed{i \quad n} \wedge$
                        $\{A\rceil \boxed{m \quad} i\} \leq^* \{A\rceil j \boxed{\quad n}\} \wedge A {\sim}_{\boxed{m\, n}} A_0\}\big)$
  $\{\mathbf{ord}_{\leq} A\rceil \boxed{m \quad n} \wedge A {\sim}_{\boxed{m\, n}} A_0\}$
```

Realignment

We write $X \simeq Y$, and say that X is a *realignment* of Y, when there is a *monotonic* bijection B from $\mathsf{dom}\, X$ to $\mathsf{dom}\, Y$ such that

$$\forall i \in \mathsf{dom}\, X.\ X(i) = Y(B(i)).$$

When X and Y are array values, i.e., functions on intervals, $X \simeq Y$ holds only when X is a “shift” of Y.

But when X and Y are functions whose domains can be arbitrary finite sets of integers — which we might call “lacy array values” — things become more interesting.

Eliminating Zeroes

```
$\{\boxed{a \quad b} \subseteq X \wedge X = X_0\}$
$\textbf{newvar}\ d\ \textbf{in}\ \big(c := a\ ;\ d := a\ ;$
  $\{\textbf{inv}\text{:}\ \boxed{a \quad}\boxed{c \quad}\boxed{d \quad b} \wedge$
    $X\rceil \boxed{a \quad}\, c \simeq X_0\rceil(\boxed{a \quad}\, d \cap \{\, i \mid X(i) \neq 0 \,\}) \wedge$
    $X\rceil \boxed{d \quad b} = X_0\rceil \boxed{d \quad b}\}$
  $\textbf{while}\ d \leq b\ \textbf{do}$
    $\textbf{if}\ X(d) = 0\ \textbf{then}\ d := d + 1$
      $\textbf{else}\ (X(c) := X(d)\ ;\ c := c + 1\ ;\ d := d + 1)\big)$
$\{\boxed{a \quad}\boxed{c \quad b} \wedge X\rceil \boxed{a \quad}\, c \simeq X_0\rceil(\boxed{a \quad b} \cap \{\, i \mid X(i) \neq 0 \,\})\}$
```

Separation Logic: An Example

Suppose

$$\mathsf{list}\ \epsilon\ \mathsf{i} \overset{\mathsf{def}}{=} \mathsf{i} = \mathbf{nil}$$

$$\mathsf{list}(\mathsf{a}{\cdot}\alpha)\ \mathsf{i} \overset{\mathsf{def}}{=} \exists \mathsf{j}.\ \mathsf{i} \hookrightarrow \mathsf{a}, \mathsf{j} \wedge \mathsf{list}\ \alpha\ \mathsf{j}$$

and consider the program

$$\begin{aligned} &LREV \overset{\mathsf{def}}{=} \mathsf{j} := \mathbf{nil}; \\ &\quad \mathbf{while}\ \mathsf{i} \neq \mathbf{nil}\ \mathbf{do}\ (\mathsf{k} := [\mathsf{i}+1]\ ;\ [\mathsf{i}+1] := \mathsf{j}\ ;\ \mathsf{j} := \mathsf{i}\ ;\ \mathsf{i} := \mathsf{k}). \end{aligned}$$

To prove $\{\mathsf{list}\ \alpha\ \mathsf{i}\}\ LREV\ \{\mathsf{list}\ \alpha^\dagger\ \mathsf{j}\}$, the invariant

$$\exists \alpha, \beta.\ \mathsf{list}\ \alpha\ \mathsf{i} \wedge \mathsf{list}\ \beta\ \mathsf{j} \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta$$

is inadequate.

An adequate invariant (in Hoare logic):

$$(\exists \alpha, \beta.\ \mathsf{list}\ \alpha\ \mathsf{i} \wedge \mathsf{list}\ \beta\ \mathsf{j} \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta)$$
$$\wedge\ (\forall \mathsf{k}.\ \mathbf{reachable}(\mathsf{i}, \mathsf{k}) \wedge \mathbf{reachable}(\mathsf{j}, \mathsf{k}) \Rightarrow \mathsf{k} = \mathbf{nil}).$$

An adequate invariant (in separation logic):

$$\exists \alpha, \beta.\ (\mathsf{list}\ \alpha\ \mathsf{i}\ *\ \mathsf{list}\ \beta\ \mathsf{j}) \wedge \alpha_0^\dagger = \alpha^\dagger{\cdot}\beta.$$

where $*$ is the *separating conjunction*.

Enriching the Concept of State

In separation logic, the *state* consists of two components:

— A *store*, which maps variables into their values,

— A *heap*, which maps addresses (which are values) into their values.

The Separating Conjunction

The assertion $p_1 \ * \ p_2$ holds for a heap h when h can be partitioned into two disjoint subheaps h_1 and h_2 such that p_1 holds in h_1 and p_2 holds in h_2.

The Points-To Relation

The assertion $\ell \mapsto e$ holds when the heap consists of a single cell mapping the address ℓ into the value of e.

Some Inference Rules

- Frame Rule (O'Hearn)

$$\frac{\{p\}\ c\ \{q\}}{\{p\ *\ r\}\ c\ \{q\ *\ r\}},$$

where the free variables of r are not modified by c.

- Existential Quantification

$$\frac{\{p\}\ c\ \{q\}}{\{\exists v.\ p\}\ c\ \{\exists v.\ q\}},$$

where v does not occur free in c.

- Concurrent Composition (O'Hearn)

$$\frac{\{p_1\}\ c_1\ \{q_1\} \qquad \{p_2\}\ c_2\ \{q_2\}}{\{p_1\ *\ p_2\}\ c_1 \parallel c_2\ \{q_1\ *\ q_2\},}$$

where the free variables of p_1, c_1, and q_1 are not modified by c_2, and vice-versa.

The Iterated Separating Conjunction

Suppose S is a finite set. Then the assertion $\bigodot_{i \in S} P(i)$ holds when the heap can be partitioned into an family h_i of heaplets indexed by i, such that, for all $i \in S$, $P(i)$ holds in h_i.

For example, $\bigodot_{i \in S} \ell + i \mapsto A(i)$ describes an array with base address ℓ, whose subscripts range over S, and whose value is the function A.

More Definitions

When S is a finite set of integers,

$$\ell \mapsto^{S} A \overset{\mathsf{def}}{=} S \subseteq \mathsf{dom}\, A \wedge \bigodot_{i \in S} \ell + i \mapsto A(i)$$

$$A \sim^{S} A' \overset{\mathsf{def}}{=} (A \rceil S) \sim (A' \rceil S).$$

Quicksort Revisited: Some Assumptions

$\{\ell \mapsto^{\boxed{m\ n}} A_0\}$

Quicksort(; ℓ, m, n)

$\{\exists A.\ \ell \mapsto^{\boxed{m\ n}} A \wedge A \sim^{\boxed{m\ n}} A_0 \wedge \mathbf{ord}_{\leq}(A\rceil \boxed{m\quad n})\}$

$\{\ell \mapsto^{\boxed{m\ n}} A_0 \wedge \boxed{m \,\vert\quad n}\}$

Partition(i, j ; ℓ, m, n)

$\{\exists A.\ \ell \mapsto^{\boxed{m\ n}} A \wedge A \sim^{\boxed{m\ n}} A_0 \wedge \boxed{m\quad j \,\vert\quad\vert\, i\quad n} \wedge$

$\{A\rceil \boxed{m\quad} i\} \leq^* \{A\rceil j \boxed{\quad n}\}\}$

Quicksort Revisited: The Procedure Body

$\{\ell \mapsto^{\boxed{m\ n}} A_0\}$

$\mathbf{if}\ m < n\ \mathbf{then}\ \mathbf{newvar}\ i, j\ \mathbf{in}\ \Big($

$\quad\{\ell \mapsto^{\boxed{m\ n}} A_0 \wedge \begin{array}{|c|c|c|}\hline m & \quad & n \\ \hline\end{array}\}$

$\quad\mathsf{Partition}(i, j\ ;\ \ell, m, n)$

$$\{\exists A_1.\ \ell \mapsto^{\boxed{m\ n}} A_1 \wedge A_1 \sim^{\boxed{m\ n}} A_0 \wedge \begin{array}{|c|c|c|}\hline m\quad j & \quad & i\quad n \\ \hline\end{array} \wedge \{A_1\rceil\boxed{m\quad\ }\,i\} \le^* \{A_1\rceil j\,\boxed{\quad\ n}\}\}$$

$$\left.\begin{array}{l}\left\{\begin{array}{l}\{\ell\mapsto^{\boxed{m\ j}} A_1\}\\ \mathsf{Quicksort}(\ ;\ \ell,m,j)\\ \{\exists A_2.\ \ell\mapsto^{\boxed{m\ j}} A_2\wedge\\ \quad A_2\sim^{\boxed{m\ j}} A_1\wedge\\ \quad \mathbf{ord}_{\le}(A_2\rceil\boxed{m\quad j})\}\end{array}\right\}\,\|\,\left\{\begin{array}{l}\{\ell\mapsto^{\boxed{i\ n}} A_1\}\\ \mathsf{Quicksort}(\ ;\ \ell,i,n)\\ \{\exists A_3.\ \ell\mapsto^{\boxed{i\ n}} A_3\wedge\\ \quad A_3\sim^{\boxed{i\ n}} A_1\wedge\\ \quad \mathbf{ord}_{\le}(A_3\rceil\boxed{i\quad n})\}\end{array}\right\}\\ \qquad * \left(\begin{array}{l}\ell\mapsto^{j\,\square\,i} A_1\wedge A_1\sim^{j\,\square\,i} A_1\wedge\\ A_1\sim^{\boxed{m\ n}} A_0\wedge \begin{array}{|c|c|c|}\hline m\quad j & \quad & i\quad n\\ \hline\end{array}\wedge\\ \{A_1\rceil\boxed{m\quad\ }\,i\}\le^*\{A_1\rceil j\,\boxed{\quad\ n}\}\end{array}\right)\end{array}\right\}\exists A_1$$

$$\{\exists A'.\ \ell\mapsto^{\boxed{m\ n}} A'\wedge A'\sim^{\boxed{m\ n}} A_1\wedge A_1\sim^{\boxed{m\ n}} A_0\wedge \begin{array}{|c|c|c|}\hline m\quad j & \quad & i\quad n\\ \hline\end{array}\wedge\{A_1\rceil\boxed{m\quad\ }\,i\}\le^*\{A_1\rceil j\,\boxed{\quad\ n}\}\wedge \mathbf{ord}_{\le}(A'\rceil\boxed{m\quad j})\wedge\mathbf{ord}_{\le}(A'\rceil\boxed{i\quad n})\}$$

$\{\exists A'.\ \ell\mapsto^{\boxed{m\ n}} A'\wedge A'\sim^{\boxed{m\ n}} A_0\wedge\mathbf{ord}_{\le}(A'\rceil\boxed{m\quad n})\}$

where $A' = (A_2\rceil\boxed{m\quad j}) \cup (A_1\rceil j\,\boxed{\quad}\,i) \cup (A_3\rceil\boxed{i\quad n})$.

What’s Next

— Permissions

— Lacy Arrays