# CIS 301: Lecture Notes on Program Verification Part II

Torben Amtoft
Department of Computing and Information Sciences
Kansas State University

May 4, 2004

These notes are a supplement to Part I[1]; also this part is inspired by [1].

## 1 Arrays

Whereas Part I of the notes on verification only considered simple data structures like integers, this part shall consider *arrays*. Arrays are much like lists, in that they can hold a sequence of values; the difference is that in an array, each element can be accessed directly, rather than by following a chain of pointers (as for lists).

Below is depicted an array $a$ with 5 elements: 7,3,9,5,2.

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 7 | 3 | 9 | 5 | 2 |

We thus have $a[0] = 7$, $a[1] = 3$, etc.

Individual elements of arrays can be updated; after issuing the command $\mathtt{a}[3] := 8$ the array $a$ will now look like

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 7 | 3 | 9 | 8 | 2 |

We shall talk about two arrays being *permutations* of each other if they contain the same elements, though perhaps in different order. This is, e.g., the case for the two arrays given below:

[1] available at `http://www.cis.ksu.edu/~tamtoft/CIS301/Spring04/verificationI.pdf`.

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 8 | 3 | 9 | 8 | 2 |

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | 9 | 2 | 8 | 8 |

We shall write perm$(a_1, a_2)$ if $a_1$ and $a_2$ are permutations of each other.

## 2 Verifying Programs Reading Arrays

Let us first consider programs which are *read-only* on arrays. For such programs, the verification principles from Part I carry through unchanged[2].

As an example, let us construct a program that stores in `m` the maximum of the first `k` elements of the array `a`, that is the maximum of $\mathtt{a}[0], \ldots, \mathtt{a}[\mathtt{k}-1]$. We assume that $\mathtt{k} \geq 1$, and that `a` indeed has at least `k` elements.

The desired postcondition can be expressed as follows:

$$\forall j(j < \mathtt{k} \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \quad \wedge \quad \exists j(j < \mathtt{k} \wedge \mathtt{a}[j] = \mathtt{m})$$

We shall need a loop, and it seems reasonable to guess that its test should be $\mathtt{i} \neq \mathtt{k}$ and its invariant should be

$$\phi : \forall j(j < \mathtt{i} \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \quad \wedge \quad \exists j(j < \mathtt{i} \wedge \mathtt{a}[j] = \mathtt{m})$$

since then the loop invariant, together with the negation of the loop test, will imply the postcondition. With the aim of establishing and maintaining the invariant $\phi$, we construct the following program:

```
i := 1;
m := a[0];
while i ≠ k do
    if a[i] > m
    then
        m := a[i];
        i := i + 1
    else
        i := i + 1
    fi
od
```

[2]For programs manipulating arrays, invariants and other properties will almost certainly contain quantifiers, whereas for programs without arrays, invariants can often be expressed in propositional logic.

To prove the correctness of this program, we annotate it:

| | |
|---|---|
| $\{k \geq 1\}$ | |
| $\{\forall j(j < 1 \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0]) \wedge$ $\exists j(j < 1 \wedge \mathtt{a}[j] = \mathtt{a}[0])\}$ | **Implies**(A) |
| `i` $:= 1;$ | |
| $\{\forall j(j < \mathtt{i} \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0]) \wedge$ $\exists j(j < \mathtt{i} \wedge \mathtt{a}[j] = \mathtt{a}[0])\}$ | **Assignment** |
| `m` $:= \mathtt{a}[0];$ | |
| $\{\phi\}$ | **Assignment** |
| `while` $\mathtt{i} \neq \mathtt{k}$ `do` | |
| $\{\phi \wedge \mathtt{i} \neq \mathtt{k}\}$ | **WhileTrue** |
| `if` $\mathtt{a}[\mathtt{i}] > \mathtt{m}$ | |
| `then` | |
| $\{\phi \wedge \mathtt{i} \neq \mathtt{k} \wedge \mathtt{a}[\mathtt{i}] > \mathtt{m}\}$ | **IfTrue** |
| $\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{a}[\mathtt{i}]) \wedge$ $\exists j(j < \mathtt{i} + 1 \wedge \mathtt{a}[j] = \mathtt{a}[\mathtt{i}])\}$ | **Implies**(B) |
| $\mathtt{m} := \mathtt{a}[\mathtt{i}];$ | |
| $\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \wedge$ $\exists j(j < \mathtt{i} + 1 \wedge \mathtt{a}[j] = \mathtt{m})\}$ | **Assignment** |
| $\mathtt{i} := \mathtt{i} + 1$ | |
| $\{\phi\}$ | **Assignment** |
| `else` | |
| $\{\phi \wedge \mathtt{i} \neq \mathtt{k} \wedge \mathtt{a}[\mathtt{i}] \leq \mathtt{m}\}$ | **IfFalse** |
| $\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \wedge$ $\exists j(j < \mathtt{i} + 1 \wedge \mathtt{a}[j] = \mathtt{m})\}$ | **Implies**(C) |
| $\mathtt{i} := \mathtt{i} + 1$ | |
| $\{\phi\}$ | **Assignment** |
| `fi` | |
| $\{\phi\}$ | **IfEnd** |
| `od` | |
| $\{\phi \wedge \mathtt{i}{=}\mathtt{k}\}$ | **WhileFalse** |
| $\{\forall j(j < \mathtt{k} \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \wedge$ $\exists j(j < \mathtt{k} \wedge \mathtt{a}[j] = \mathtt{m})\}$ | **Implies** |

Below we shall show the validity of (A) and (B) and (C); it is then an easy exercise to check the validity of the rest of the annotations.

To see that (A) is valid, observe that 0 is the only $j$ such that $j < 1$.

To see that (B) is valid, we must prove that

$$\forall j(j < \mathtt{i} \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \text{ and} \tag{1}$$
$$\exists j(j < \mathtt{i} \wedge \mathtt{a}[j] = \mathtt{m}) \text{ and} \tag{2}$$
$$\mathtt{a}[\mathtt{i}] > \mathtt{m} \tag{3}$$

implies

$$\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{a}[\mathtt{i}]) \text{ and} \tag{4}$$
$$\exists j(j < \mathtt{i} + 1 \wedge \mathtt{a}[j] = \mathtt{a}[\mathtt{i}]). \tag{5}$$

To establish (4), let $j$ be given such that $j < \mathtt{i} + 1$: if $j = \mathtt{i}$, the claim is trivial; otherwise, $j < \mathtt{i}$ and the claim follows from (1) and (3). For (5), we can use $j = \mathtt{i}$.

To see that (C) is valid, we must prove that

$$\forall j(j < \mathtt{i} \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \text{ and} \tag{6}$$
$$\exists j(j < \mathtt{i} \wedge \mathtt{a}[j] = \mathtt{m}) \text{ and} \tag{7}$$
$$\mathtt{a}[\mathtt{i}] \leq \mathtt{m} \tag{8}$$

implies

$$\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{m}) \text{ and} \tag{9}$$
$$\exists j(j < \mathtt{i} + 1 \wedge \mathtt{a}[j] = \mathtt{m}). \tag{10}$$

To establish (9), let $j$ be given such that $j < \mathtt{i} + 1$. If $j = \mathtt{i}$, the claim follows from (8). Otherwise, $j < \mathtt{i}$ and the claim follows from (6). Finally, (10) follows from (7).

# 3 Verifying Programs Updating Arrays

Next we consider programs which also *write* on arrays, that is contain commands of the form $\mathtt{a}[\mathtt{i}] := E$. For such assignments, we want to apply the proof rule

$\{\psi(E)\}$
$\quad \mathtt{x} := E$
$\triangleright \quad \{\psi(x)\}$ **Assignment**

But if we apply that rule *naively* to the assignment $\mathtt{a}[2] := \mathtt{x}$ and the postcondition $\forall j(j < 10 \rightarrow \mathtt{a}[j] > 5)$, substituting the right hand side of the assignment for the left hand side, we would infer (since $\mathtt{a}[2]$ does not occur in the postcondition) that the following program is well-annotated:

$\{\forall j(j < 10 \rightarrow \mathtt{a}[j] > 5)\}$
$\quad \mathtt{a}[2] := \mathtt{x}$
$\{\forall j(j < 10 \rightarrow \mathtt{a}[j] > 5)\}$

This is clearly unsound, as can be seen by taking $\mathtt{x} = 3$.

Instead, the proper treatment is to interpret an assignment $\mathtt{a}[\mathtt{i}] := E$ as being really the assignment

$$\mathtt{a} := \mathtt{a}\{\mathtt{i} \mapsto E\}$$

That is, we assign to $\mathtt{a}$ an array that is like $\mathtt{a}$, except that in position $\mathtt{i}$ it behaves like $E$. More formally, we have

$$\begin{array}{lcll} \mathtt{a}\{\mathtt{i} \mapsto E\}[j] & = & E & \text{if } j = \mathtt{i} \\ \mathtt{a}\{\mathtt{i} \mapsto E\}[j] & = & \mathtt{a}[j] & \text{if } j \neq \mathtt{i} \end{array}$$

Then, in the above example, we get the well-annotated program

$\{\forall j(j < 10 \rightarrow \mathtt{a}\{2 \mapsto \mathtt{x}\}[j] > 5)\}$
$\quad \mathtt{a}[2] := \mathtt{x}$
$\{\forall j(j < 10 \rightarrow \mathtt{a}[j] > 5)\}$

where the precondition can be simplified to

$$\forall j((j < 10 \wedge j \neq 2) \rightarrow \mathtt{a}[j] > 5) \wedge \mathtt{x} > 5$$

which is as expected.

As an example, let us construct a program that rearranges the first $\mathtt{k}$ elements of an array $\mathtt{a}$ such that the highest element is placed in position number 0.

The desired postcondition can be expressed as follows:

$$\forall j(j < \mathtt{k} \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0]) \ \wedge \ \mathrm{perm}(\mathtt{a}, a_0)$$

where the logical variable $a_0$ denotes the initial value of $a$; the latter condition $\mathrm{perm}(\mathtt{a}, a_0)$ is also part of the precondition. We shall need a loop, and it seems reasonable to guess that its test should be $\mathtt{i} \neq \mathtt{k}$ and its invariant should be

$$\psi : \forall j(j < \mathtt{i} \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0]) \ \wedge \ \mathrm{perm}(\mathtt{a}, a_0)$$

since then the loop invariant, together with the negation of the loop test, will imply the postcondition. With the aim of establishing and maintaining the invariant $\psi$, we construct the following program:

```
i := 1;
while i ≠ k do
    if a[i] > a[0]
    then
        t := a[0];
        a[0] := a[i];
        a[i] := t;
        i := i + 1
    else
        i := i + 1
    fi
od
```

To prove the correctness of this program, we annotate it, as done in Fig. 1. Below we shall show the validity of (D); it is then an easy exercise to check the validity of the rest of the annotations.

Let $a' = \mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{a}[0]\}$; we must prove that

$$\forall j(j < \mathtt{i} \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0]) \text{ and} \tag{1}$$
$$\mathrm{perm}(\mathtt{a}, a_0) \text{ and} \tag{2}$$
$$\mathtt{a}[\mathtt{i}] > \mathtt{a}[0] \tag{3}$$

implies

$$\forall j(j < \mathtt{i} + 1 \rightarrow a'[j] \leq a'[0]) \tag{4}$$
$$\mathrm{perm}(a', a_0) \tag{5}$$

$\{\mathrm{perm}(\mathtt{a}, a_0)\}$

$\{\forall j(j < 1 \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}, a_0)\}$ **Implies**

$\quad \mathtt{i} := 1;$

$\{\psi\}$ **Assignment**

$\quad$ `while` $\mathtt{i} \neq \mathtt{k}$ `do`

$\{\psi \wedge \mathtt{i} \neq \mathtt{k}\}$ **WhileTrue**

$\quad\quad$ `if` $\mathtt{a}[\mathtt{i}] > \mathtt{a}[0]$

$\quad\quad$ `then`

$\{\psi \wedge \mathtt{i} \neq \mathtt{k} \wedge \mathtt{a}[\mathtt{i}] > \mathtt{a}[0]\}$ **IfTrue**

$\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{a}[0]\}[j] \leq \mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{a}[0]\}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{a}[0]\}, a_0)\}$ **Implies**(D)

$\quad\quad\quad \mathtt{t} := \mathtt{a}[0];$

$\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{t}\}[j] \leq \mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{t}\}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}\{0 \mapsto \mathtt{a}[\mathtt{i}]\}\{\mathtt{i} \mapsto \mathtt{t}\}, a_0)\}$ **Assignment**

$\quad\quad\quad \mathtt{a}[0] := \mathtt{a}[\mathtt{i}];$

$\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}\{\mathtt{i} \mapsto \mathtt{t}\}[j] \leq \mathtt{a}\{\mathtt{i} \mapsto \mathtt{t}\}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}\{\mathtt{i} \mapsto \mathtt{t}\}, a_0)\}$ **Assignment**

$\quad\quad\quad \mathtt{a}[\mathtt{i}] := \mathtt{t};$

$\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}, a_0)\}$ **Assignment**

$\quad\quad\quad \mathtt{i} := \mathtt{i} + 1$

$\{\psi\}$ **Assignment**

$\quad\quad$ `else`

$\{\psi \wedge \mathtt{i} \neq \mathtt{k} \wedge \mathtt{a}[\mathtt{i}] \leq \mathtt{a}[0]\}$ **IfFalse**

$\{\forall j(j < \mathtt{i} + 1 \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}, a_0)\}$ **Implies**

$\quad\quad\quad \mathtt{i} := \mathtt{i} + 1$

$\{\psi\}$ **Assignment**

$\quad\quad$ `fi`

$\{\psi\}$ **IfEnd**

$\quad$ `od`

$\{\psi \wedge \mathtt{i} = \mathtt{k}\}$ **WhileFalse**

$\{\forall j(j < \mathtt{k} \rightarrow \mathtt{a}[j] \leq \mathtt{a}[0])$
$\wedge\ \mathrm{perm}(\mathtt{a}, a_0)\}$ **Implies**



Figure 1: A well-annotated program for putting the highest array value first.

Clearly $a'$ is a permutation of $\mathtt{a}$, so (5) follows from (2). To show (4), let $j < \mathtt{i}+1$ be given; we must show that $a'[j] \leq a'[0]$ which is trivial if $j = 0$ so assume that $0 < j < \mathtt{i}+1$. Since $a'[0] = \mathtt{a}[\mathtt{i}]$, our task can be accomplished by showing that

$$a'[j] \leq \mathtt{a}[\mathtt{i}].$$

We do so by a case analysis on the value of $j$. If $j = \mathtt{i}$, the claim follows from (3) since $a'[j] = \mathtt{a}[0]$. Otherwise, $0 < j < \mathtt{i}$ and therefore $a'[j] = \mathtt{a}[j]$; the claim thus boils down to showing $\mathtt{a}[j] \leq \mathtt{a}[\mathtt{i}]$ which follows from (1) and (3).

## References

[1] David Gries. *The Science of Programming.* Springer-Verlag, 1981.