(Imperative) Program Logic
Part 2

Robert Keller
February 2013
Recall the **while** rule

- In order to use the while rule in JAPE, it is necessary to supply an **invariant**, $I$.

\[
\{I \land P\} \rightarrow S \{I\}
\]

\[
\{I\} \quad \textbf{while}(P) \rightarrow S \quad \{I \land \neg P\}
\]
Inferring invariants

• There is no fully general automation for inferring invariants (as there is for the weakest pre-condition/assumption for assignment statements).

• This is one of the things that makes totally automated verification difficult.

• Finding the right invariant is still a human intellectual activity.
Using the **while** rule

- In JAPE, an assertion **implying** the invariant (by using the consequent(L) rule) is included as an assertion before the while, and also doubly serves as a post-condition for the preceding statement.

(What does this code do?)

```plaintext
...  
i=Ki \land j=Kj \land i \geq 0)(k:=0)  
1. \{i \geq 0 \land k+i \times j = K_i \times K_j\}  
   while i \neq 0 do k:=k+j;i:=i-1 od  
   \{k=K_i \times K_j\}
```

Should imply the invariant
Invariant and negation of test should imply this.
Using the **while** rule

- Before using Jape’s while rule, this setup is decomposed using Jape’s **Ntuple** rule.

\[
\begin{align*}
1: & \{i=Ki \land j=Kj \land i \geq 0\}(k:=0) \\
& \{i \geq 0 \land k+i \cdot j = Ki \times Kj\} \\
\vdots \\
& \{i \geq 0 \land k+i \cdot j = Ki \times Kj\} \\
2: & \text{while } i \neq 0 \text{ do } k:=k+j; i:=i-1 \text{ od} \\
& \{k=Ki \times Kj\} \\
& \{i=Ki \land j=Kj \land i \geq 0\}(k:=0) \\
& \{i \geq 0 \land k+i \cdot j = Ki \times Kj\} \\
3: & \text{while } i \neq 0 \text{ do } k:=k+j; i:=i-1 \text{ od} \\
& \{k=Ki \times Kj\}
\end{align*}
\]

Prove using assignment rule

Prove using while rule

Ntuple rule used
A proof of a simple program

1: \( i = 10 \land i > 0 \)
2: \( i - 1 = 10 \)
3: \( i = 10 \land i > 0 \rightarrow i - 1 = 10 \)
4: \( \{ i - 1 = 10 \} \{ i := i - 1 \} \{ i = 10 \} \)
5: \( \{ i = 10 \land i > 0 \} \{ i := i - 1 \} \{ i = 10 \} \)
6: \( i = 10 \land i > 0 \)
7: \( i > 0 \)
8: \( i = 10 \land i > 0 \rightarrow i > 0 \)

9: integer \( K_m \)
10: \( i = 10 \land i > 0 \land i = K_m \)
11: \( i - 1 < K_m \)
12: \( i = 10 \land i > 0 \land i = K_m \rightarrow i - 1 < K_m \)
13: \( \{ i - 1 < K_m \} \{ i := i - 1 \} \{ i < K_m \} \)
14: \( \{ i = 10 \land i > 0 \land i = K_m \} \{ i := i - 1 \} \{ i < K_m \} \)
15: \( \{ i = 10 \} \text{while } i > 0 \text{ do } i := i - 1 \text{ od} \{ i = 10 \land \neg (i > 0) \} \) \( \text{while } 5, 8, 9-14 \)
16: \( i = 10 \land \neg (i > 0) \rightarrow i = 0 \)
17: \( \{ i = 10 \} \text{while } i > 0 \text{ do } i := i - 1 \text{ od} \{ i = 0 \} \)

assumption
obviously
\( \rightarrow \text{ intro } 1-2 \)
variable-assignment
consequence(L) 3, 4
assumption
\( \land \text{ elim } 6 \)
\( \rightarrow \text{ intro } 6-7 \)
assumption
assumption
obviously
\( \rightarrow \text{ intro } 10-11 \)
variable-assignment
consequence(L) 12, 13
consequence(R) 15, 16
Subtleties about loop invariants

- Can the following be proved?

\[
\begin{align*}
\text{\cdots} \\
1: & \{ y=0 \land i=0 \land n \geq 0 \} \od \{ y=i \times i \land i \leq n \land i \geq 0 \} \\
& \text{while } i < n \text{ do } y := y + j; j := j + 2; i := i + 1 \text{ od} \{ y = n \times n \}
\end{align*}
\]

Provided:
DISTINCT \( i, j, n, y \)
The loop invariant is not strong enough to enable induction

- This is more likely provable.

\[
\begin{align*}
1.: \{y=0 \land i=0 \land n \geq 0\} (j:=1) & \{y=i \times i \land i \leq n \land i \geq 0 \land j=2 \times i + 1\} \\
\text{while } i < n \text{ do } y := y + j; j := j + 2; i := i + 1 \od \{y=n \times n\}
\end{align*}
\]

Provided:

DISTINCT i, j, n, y
• What is \( M \) for termination?

\[
\begin{align*}
\ ... \\
1. \{y=0 \land i=0 \land n \geq 0 \} (j:=1) \{y=i \times i \land i \leq n \land i \geq 0 \land j=2 \times i+1 \} \\
\text{while } i < n \text{ do } y:=y+j; j:=j+2; i:=i+1 \text{ od} \{y=n \times n\}
\end{align*}
\]

Provided:
DISTINCT i, j, n, y
A Completed Proof (lines 1-24 of 51)

1. \( y=0 \land i=0 \land n \geq 0 \)
2. \( y=0 \)
3. \( i=0 \)
4. \( n \geq 0 \)
5. \( y=i \times i \)
6. \( i \leq n \)
7. \( i \geq 0 \)
8. \( 1=2 \times i+1 \)
9. \( y=i \times i \leq n \land i \geq 0 \land 1=2 \times i+1 \)
10. \( y=0 \land i=0 \land n \geq 0 \rightarrow y=i \times i \leq n \land i \geq 0 \land 1=2 \times i+1 \)
11. \( \{ y=i \times i \leq n \land i \geq 0 \land 1=2 \times i+1 \}(j:=1) \{ y=i \times i \leq n \land i \geq 0 \land j=2 \times i+1 \} \)
12. \( \{ y=i \times i \leq n \land i \geq 0 \land j=2 \times i+1 \} \)
13. \( y=i \times i \leq n \land i \geq 0 \land j=2 \times i+1 \land i < n \)
14. \( y=i \times i \)
15. \( i \geq 0 \)
16. \( j=2 \times i+1 \)
17. \( i < n \)
18. \( y+j=(i+1) \times (i+1) \)
19. \( i+1 \leq n \)
20. \( i+1 \geq 0 \)
21. \( j+2=2 \times (i+1)+1 \)
22. \( y+j=(i+1) \times (i+1) \land i+1 \leq n \land i+1 \geq 0 \land j+2=2 \times (i+1)+1 \)
23. \( y=i \times i \leq n \land i \geq 0 \land j=2 \times i+1 \land i < n \rightarrow y+j=(i+1) \times (i+1) \land i+1 \leq n \land i+1 \geq 0 \land j+2=2 \times (i+1)+1 \)
24. \( \{ y+j=(i+1) \times (i+1) \land i+1 \leq n \land i+1 \geq 0 \land j+2=2 \times (i+1)+1 \}(y:=y+j) \{ y=(i+1) \times (i+1) \land i+1 \leq n \land i+1 \geq 0 \land j+2=2 \times (i+1)+1 \} \)

Initialization

First assignment in loop body
The Completed Proof (lines 25-51 of 51)

25: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \} \land y = j + i \times (i + 1) \times i + 1 \leq n \land i + 1 \leq 0 \land j + 2 = 2 \times (i + 1) + 1
collection(23, 24)
variable-assignment

26: \{y = (i + 1) \times (i + 1) \times i + 1 \leq n \land i + 1 \leq 0 \land j + 2 = 2 \times (i + 1) + 1 \} \land j = i + 2 \land y = (i + 1) \times (i + 1) \times i + 1 \leq n \land i + 1 \leq 0 \land j = 2 \times (i + 1) + 1
collection(25, 26, 27)
variable-assignment

27: \{y = (i + 1) \times (i + 1) \times i + 1 \leq n \land i + 1 \leq 0 \land j = 2 \times (i + 1) + 1 \} \land (i = i + 1) \land \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1\}

28: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \} \land (y = y + j + j = 2 \times i + 1) \land \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1\}

Other assignments in loop body

Condition on _M

29: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n\}

assumption
\land elim 29
obviously, from 30
→ intro 29-31

30: i < n

31: n - i > 0

32: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \land n - i > 0\}

integer _K_m

33: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \land n - i = K_m\}

assumption

34: n = K_m

35: n - i = K_m

36: n - (i + 1) = K_m

37: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \land n - i = K_m \land n - (i + 1) < K_m\}

38: \{n - (i + 1) < K_m \} \land y = j \land n - (i + 1) = K_m

39: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \land n - i = K_m \} \land (y = j + j = 2 \times i + 1) \land n - (i + 1) = K_m

40: \{n - (i + 1) < K_m \} \land \{j = i + 2 \} \land \{n - (i + 1) < K_m \}

41: \{n - (i + 1) < K_m \} \land \{i = i + 1 \} \land \{n - i < K_m \}

42: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land i < n \land n - i = K_m \} \land (y = j + j = 2 \times i + 1) \land \{i = i + 1 \} \land \{n - i < K_m \}

43: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \} \land \{\text{while } i < n \} \land \{y = y + j \land j = 2 \times i + 1 \} \land \{n - i < K_m \}

\text{while } 28, 32, 33, 42

Termination of loop body

Exit consequence

44: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land \neg (i < n)\}

assumption
\land elim 44
\land elim 44
\land elim 44
obviously, from 47, 46, 45
→ intro 44-48

45: \{y = i \times i\}

46: \leq n

47: \neg (i < n)

48: n = n \times n

49: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land \neg (i < n) \} \land \neg y = n \times n

50: \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \} \land \{\text{while } i < n \} \land \{y = y + j \land j = 2 \times i + 1 \} \land \{n - i < K_m \} \land \{d = y \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \land \neg (i < n)\}

51: \{y = 0 \land i = 0 \land n \geq 0 \} \land \{j = 1 \} \land \{y = i \times i \leq n \land i \leq 0 \land j = 2 \times i + 1 \} \land \{\text{while } i < n \} \land \{y = y + j \land j = 2 \times i + 1 \} \land \{n - i < K_m \} \land \{d = y \times n \land n \}

Ntupl 12, 50
Verifying Array Programs

- Arrays present extra challenges and interesting issues.
- A useful dichotomy:
  - Programs with read-only arrays
  - Programs with modifiable arrays
Array Mathematics

- An array can be treated as a **function**:
  - It maps indices into values.
  - e.g. a 1-dimensional array with dimension 10 maps \{0, ..., 9\} into values of the type stored in the array.
  - \(a[i]\) is the value of this function with argument \(i\)

- Because several indices can have the same value, arrays are more susceptible to variable **aliasing**, e.g.

  \[i := 5; j = 6-1; a[j] = a[i]+1\]
Read-Only Array Example

This program sets j to the last index i such that a[i] = 0. The array is assumed to be indexed 0..n-1. If there is no such value, it leaves j at its initial value n.

```
i := 0;
j := n;
while i < n do
  if a[i] = 0 then j := i
  else skip
  fi
  i := i+1
od
```

What invariant do we need?

\{n \geq 0 \land \text{length}(a)=n\}

\{j < n \rightarrow a[j] = 0\}
Read-Only Array Example

This program sets $j$ to the last index $i$ such that $a[i] = 0$. The array is assumed to be indexed $0..n-1$. If there is no such value, it leaves $j$ at its initial value $n$.

```
\ldots
1: \{ n \geq 0 \land \text{length}(a) = n \} \langle \mathbf{i} := 0; \mathbf{j} := n \rangle \{ i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \}
\begin{array}{c}
\text{while } i < n \text{ do if } a[i] = 0 \text{ then } j := i \text{ else skip fi; } i := i + 1 \text{ od}
\end{array}
\{ j < n \rightarrow a[j] = 0 \}
```

Provided:

DISTINCT $a$, $i$, $j$, $n$
Read-Only Example (lines 1-15)

1: \( n \geq 0 \land \text{length}(a) = n \)
2: \( n \geq 0 \)
3: \( \text{length}(a) = n \)
4: \( 0 \leq n \)
5: \( 0 \geq 0 \)
6: \( n < n \rightarrow a[n] = 0 \)
7: \( \bot \)
8: \( a[n] = 0 \)
9: \( n < n \rightarrow a[n] = 0 \)
10: \( 0 \leq n \land 0 \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0) \)
11: \( n \geq 0 \land \text{length}(a) = n \land 0 \leq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0) \)
12: \( 0 \leq n \land 0 \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0) \)
13: \( \{i \leq n \land i \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0) \}
14: \( \{i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \}
15: \{i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \}

assumption
\( \land \) elim 1
\( \land \) elim 1
\( A \vdash B \vdash A \)
obviously
assumption
obviously, from 6
contra (constructive) 7
\( \rightarrow \) intro 6–8
\( \land \) intro 4,5,3,9
\( \rightarrow \) intro 1–10
variable-assignment
consequence(L) 11,12
variable-assignment
sequence 13,14
16: \( i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \land i < n \)
17: \( i \geq 0 \)
18: \( \text{length}(a) = n \)
19: \( j < n \rightarrow a[j] = 0 \)
20: \( i < n \)
21: \( a[i] = 0 \)
22: \( i + 1 \leq n \)
23: \( i + 1 \geq 0 \)
24: \( i < n \)
25: \( a[i] = 0 \)
26: \( i < n \rightarrow a[i] = 0 \)
27: \( i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (i < n \rightarrow a[i] = 0) \)
28: \( a[i] = 0 \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (i < n \rightarrow a[i] = 0) \)
29: \( \neg(a[i] = 0) \)
30: \( i + 1 \leq n \)
31: \( i + 1 \geq 0 \)
32: \( i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \)
33: \( \neg(a[i] = 0) \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \)
34: \( 0 \leq i \)
35: \( 1 \leq \text{length}(a) \)
36: \( a[i] = 0 \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (i < n \rightarrow a[i] = 0) \land (\neg(a[i] = 0) \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)) \land 0 \leq i \land i < \text{length}(a) \)
37: \( i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0) \land i < n \land (\neg(a[i] = 0) \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)) \land 0 \leq i \land i < \text{length}(a) \)
Read-Only Example (lines 38-47)

38: \(i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (i < n - a[i] = 0)\) if \(j := i\) \((i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\)

39: \((i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) skip \((i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\)

40: \((a[i] = 0 \rightarrow i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (i < n - a[i] = 0)\) \wedge \lnot (a[i] = 0) \rightarrow i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) \wedge 0 \leq i \wedge \text{length}(a)\)

41: \(i \leq n \wedge i \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) \wedge i < n \) if \(a[i] = 0\) then \(j := i\) else skip \((i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\)

42: \((i+1 \leq n \wedge i+1 \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) \wedge (i := i+1) \wedge i \leq n \wedge i \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\)

43: \((i \leq n \wedge i \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) \wedge i < n) \) if \(a[i] = 0\) then \(j := i\) else skip \((i := i+1) \wedge i \leq n \wedge i \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\)

44: \((i \leq n \wedge i \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) \wedge i < n \)

45: \(i < n\)

46: \(n-i > 0\)

47: \((i \leq n \wedge i \geq 0 \wedge \text{length}(a) = n \wedge (j < n - a[j] = 0)\) \wedge i < n \rightarrow n-i > 0\)

---

variable-assignment

skip

choice 38,39

consequence(1) 37,40

variable-assignment

sequence 41,42

assumption

\wedge\ elim 44

obviously, from 45

\rightarrow intro 44–46
### Read-Only Example (lines 48-74)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>[\text{integer } K_m]</td>
<td>assumption</td>
</tr>
<tr>
<td>49</td>
<td>[1 \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0) \land i &lt; n \land i = K_m]</td>
<td>assumption</td>
</tr>
<tr>
<td>50</td>
<td>[i \geq 0]</td>
<td>[\land \text{elim } 49]</td>
</tr>
<tr>
<td>51</td>
<td>[i &lt; n]</td>
<td>[\land \text{elim } 49]</td>
</tr>
<tr>
<td>52</td>
<td>[n - i = K_m]</td>
<td>assumption</td>
</tr>
<tr>
<td>53</td>
<td>[a[i] = 0]</td>
<td>obviously, from 52</td>
</tr>
<tr>
<td>54</td>
<td>[n - (i + 1) &lt; K_m]</td>
<td>[\rightarrow \text{intro } 53-54]</td>
</tr>
<tr>
<td>55</td>
<td>[a[i] = 0 \land n - (i + 1) &lt; K_m]</td>
<td>assumption</td>
</tr>
<tr>
<td>56</td>
<td>[\neg (a[i] = 0)]</td>
<td>obviously, from 52</td>
</tr>
<tr>
<td>57</td>
<td>[n - (i + 1) &lt; K_m]</td>
<td>[\rightarrow \text{intro } 56-57]</td>
</tr>
<tr>
<td>58</td>
<td>[\neg (a[i] = 0) \land n - (i + 1) &lt; K_m]</td>
<td>[A \geq B \geq A\ G50]</td>
</tr>
<tr>
<td>59</td>
<td>[0 \leq i]</td>
<td>[\land \text{intro } 55, 58, 59, 60]</td>
</tr>
<tr>
<td>60</td>
<td>[i &lt; \text{length}(a)]</td>
<td>[\rightarrow \text{intro } 49-61]</td>
</tr>
<tr>
<td>61</td>
<td>[a[i] = 0 \land n - (i + 1) &lt; K_m \land \neg (a[i] = 0) \land n - (i + 1) &lt; K_m \land 0 \leq i &lt; \text{length}(a)]</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>62</td>
<td>[i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0) \land i &lt; n \land i = K_m \land (a[i] = 0 \land n - (i + 1) &lt; K_m) \land \neg (a[i] = 0) \land n - (i + 1) &lt; K_m) \land 0 \leq i &lt; \text{length}(a)]</td>
<td>skip</td>
</tr>
<tr>
<td>63</td>
<td>[\neg (a[i] = 0) \land n - (i + 1) &lt; K_m]</td>
<td>choice 63, 64</td>
</tr>
<tr>
<td>64</td>
<td>[\neg (a[i] = 0) \land n - (i + 1) &lt; K_m]</td>
<td>consequence 62, 65</td>
</tr>
<tr>
<td>65</td>
<td>[\neg (a[i] = 0) \land n - (i + 1) &lt; K_m]</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>66</td>
<td>[\neg (a[i] = 0) \land n - (i + 1) &lt; K_m]</td>
<td>sequence 66, 67</td>
</tr>
<tr>
<td>67</td>
<td>[\neg (a[i] = 0) \land n - (i + 1) &lt; K_m]</td>
<td>while 43, 47, 48-68</td>
</tr>
<tr>
<td>68</td>
<td>[i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0) \land i &lt; n \land i = K_m]</td>
<td>assumption</td>
</tr>
<tr>
<td>69</td>
<td>[\text{while } i &lt; n \text{ do if } a[i] = 0 \text{ then } j := i + 1 \text{ od}]</td>
<td>[\land \text{elim } 70]</td>
</tr>
<tr>
<td>70</td>
<td>[i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0) \land (i &lt; n)]</td>
<td>[\rightarrow \text{intro } 70-71]</td>
</tr>
<tr>
<td>71</td>
<td>[i &lt; n - a[j] = 0]</td>
<td>consequence 69, 72</td>
</tr>
<tr>
<td>72</td>
<td>[i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0) \land (i &lt; n - j &lt; n - a[j] = 0)]</td>
<td>Tuple 15, 73</td>
</tr>
<tr>
<td>73</td>
<td>[i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0)]</td>
<td>while 43, 47, 48-68</td>
</tr>
<tr>
<td>74</td>
<td>[n \geq 0 \land \text{length}(a) = n \land i = 0 \land j = n]</td>
<td>assumption</td>
</tr>
<tr>
<td></td>
<td>[i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j &lt; n - a[j] = 0)]</td>
<td>consequence 69, 72</td>
</tr>
<tr>
<td></td>
<td>[\text{while } i &lt; n \text{ do if } a[i] = 0 \text{ then } j := i + 1 \text{ od}]</td>
<td>Tuple 15, 73</td>
</tr>
</tbody>
</table>
Quantifiers

• Quantifiers are handy representing information about arrays, e.g.

• $\forall i \ ((0 < i) \land (i < n)) \rightarrow a[i-1] \leq a[i]

• $\exists i \ ((0 \leq i) \land (i < n) \land a[i] = 0)$
Quantifier Example

This program assumes there is an array element having value 0. It returns an index of such a value.

This is from the JAPE “factory samples”. Will the invariant do the job?
Subtleties with Array Programs

\[
\{\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0) \} (i := 0)
\]
\[
\{0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)\} \text{while } a[i] \neq 0 \text{ do } i := i + 1 \text{ od}\{a[i] = 0\}
\]

- Look at part of the invariant here.
- Note that the lower bound on x is a function of the index i.
- This is important, because it says that the element such that \(a[x] = 0\) is yet to be found.
- We need this invariant to prove termination.
- The loop test will stop when \(a[i] = 0\).

- The expansion order is tricky.
Quantifier Example Proved

- Things go pretty routinely, until this ...

How do we get $i+1 < \text{length}(a)$?
∃-elimination to the rescue

3: $\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0$

4: $0 \leq i$

5: $i < \text{length}(a)$

6: $\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)$

7: $a[i] \neq 0$

8: integer $i_1$, $i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0$

...$

9: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)$

10: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)$
Now case analysis

3: $0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0$

4: $0 \leq i$

5: $i < \text{length}(a)$

6: $\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)$

7: $a[i] \neq 0$

8: \text{integer } i_1, i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0$

9: $i \leq i_1$

10: $i_1 < \text{length}(a)$

11: $a[i_1] = 0$

...  

12: $0 \leq i+1 \land i+1 < \text{length}(a) \land \exists x. (i+1 \leq x \land x < \text{length}(a) \land a[x] = 0)$

13: $0 \leq i+1 \land i+1 < \text{length}(a) \land \exists x. (i+1 \leq x \land x < \text{length}(a) \land a[x] = 0)$
Setting up for case analysis

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3:</td>
<td>$0 \leq i \land i &lt; \text{length}(a) \land \exists x. (i \leq x \land x &lt; \text{length}(a) \land a[x] = 0) \land a[i] \neq 0$</td>
</tr>
<tr>
<td>4:</td>
<td>$0 \leq i$</td>
</tr>
<tr>
<td>5:</td>
<td>$i &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>6:</td>
<td>$\exists x. (i \leq x \land x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
<tr>
<td>7:</td>
<td>$a[i] \neq 0$</td>
</tr>
<tr>
<td>8:</td>
<td>integer $i_1$, $i \leq i_1 \land i_1 &lt; \text{length}(a) \land a[i_1] = 0$</td>
</tr>
<tr>
<td>9:</td>
<td>$i \leq i_1$</td>
</tr>
<tr>
<td>10:</td>
<td>$i_1 &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>11:</td>
<td>$a[i_1] = 0$</td>
</tr>
<tr>
<td>12:</td>
<td>$i &lt; i_1 \lor i = i_1$</td>
</tr>
<tr>
<td>13:</td>
<td>$0 \leq i + 1 \land i + 1 &lt; \text{length}(a) \land \exists x. (i + 1 \leq x \land x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
</tbody>
</table>
| 14: | $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)$ | from 9
Strategy: $\forall$-Elimination

```
3: 0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0
4: 0 \leq i
5: i < \text{length}(a)
6: \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)
7: a[i] \neq 0
8: \text{integer } i_1, i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0
9: i \leq i_1
10: i_1 < \text{length}(a)
11: a[i_1] = 0
12: i < i_1 \land i = i_1
13: i < i_1
14: \ldots
15: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
16: i_1 = i
17: \ldots
18: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
19: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
20: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
```
Upper branch: $\exists$-introduction

```
12: $i < i_1 \lor i = i_1$
13: $i < i_1$
   ...
14: $0 \leq i + 1$
   ...
15: $i + 1 < \text{length}(a)$
   ...
16: $\exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)$
17: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)$
```
### Upper branch closure

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$0 \leq i &lt; \text{length}(a) \land \exists x. (i \leq x &lt; \text{length}(a) \land a[x] = 0) \land a[i] \neq 0$</td>
</tr>
<tr>
<td>4</td>
<td>$0 \leq i$</td>
</tr>
<tr>
<td>5</td>
<td>$i &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>6</td>
<td>$\exists x. (i \leq x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
<tr>
<td>7</td>
<td>$a[i] \neq 0$</td>
</tr>
<tr>
<td>8</td>
<td>integer $i_1$, $i \leq i_1 &lt; \text{length}(a) \land a[i_1] = 0$</td>
</tr>
<tr>
<td>9</td>
<td>$i \leq i_1$</td>
</tr>
<tr>
<td>10</td>
<td>$i_1 &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>11</td>
<td>$a[i_1] = 0$</td>
</tr>
<tr>
<td>12</td>
<td>$i &lt; i_1 \lor i = i_1$</td>
</tr>
<tr>
<td>13</td>
<td>$i &lt; i_1$</td>
</tr>
<tr>
<td>14</td>
<td>$0 \leq i + 1$</td>
</tr>
<tr>
<td>15</td>
<td>$i + 1 &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>16</td>
<td>$i + 1 \leq i_1$</td>
</tr>
<tr>
<td>17</td>
<td>$i + 1 \leq i_1 \land i + 1 &lt; \text{length}(a) \land a[i_1] = 0$</td>
</tr>
<tr>
<td>18</td>
<td>$\exists x. (i + 1 \leq x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
<tr>
<td>19</td>
<td>$0 \leq i + 1 \land i + 1 &lt; \text{length}(a) \land \exists x. (i + 1 \leq x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
</tbody>
</table>
Lower branch

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$0 \leq i \land i &lt; \text{length}(a) \land \exists x. (i \leq x \land x &lt; \text{length}(a) \land a[x] = 0) \land a[i] \neq 0$</td>
</tr>
<tr>
<td>4</td>
<td>$0 \leq i$</td>
</tr>
<tr>
<td>5</td>
<td>$i &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>6</td>
<td>$\exists x. (i \leq x \land x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
<tr>
<td>7</td>
<td>$a[i] \neq 0$</td>
</tr>
<tr>
<td>8</td>
<td>\text{integer } i_1, \ i \leq i_1 \land i_1 &lt; \text{length}(a) \land a[i_1] = 0</td>
</tr>
<tr>
<td>9</td>
<td>$i \leq i_1$</td>
</tr>
<tr>
<td>10</td>
<td>$i_1 &lt; \text{length}(a)$</td>
</tr>
<tr>
<td>11</td>
<td>$a[i_1] = 0$</td>
</tr>
<tr>
<td>12</td>
<td>$i &lt; i_1 \lor i = i_1$</td>
</tr>
<tr>
<td>13</td>
<td>$i = i_1$</td>
</tr>
<tr>
<td>14</td>
<td>$0 \leq i+1 \land i+1 &lt; \text{length}(a) \land \exists x. (i+1 \leq x \land x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
<tr>
<td>15</td>
<td>$i = i_1$</td>
</tr>
<tr>
<td>16</td>
<td>$0 \leq i+1 \land i+1 &lt; \text{length}(a) \land \exists x. (i+1 \leq x \land x &lt; \text{length}(a) \land a[x] = 0)$</td>
</tr>
</tbody>
</table>

Collapsed detail
Closure of lower branch

5: \( a[i] \neq 0 \)
6: \( \text{integer } i_1, i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0 \)
7: \( i \leq i_1 \)
8: \( a[i_1] = 0 \)
9: \( i < i_1 \lor i = i_1 \)
10: \( i < i_1 \)
11: \( 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \)
12: \( i = i_1 \)
13: \( a[i] = 0 \)
14: \( \neg (a[i] = 0) \)
15: \( \bot \)
16: \( 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \)

\( \land \text{elim } 3 \)
assumptions
\( \land \text{elim } 6.2 \)
\( \land \text{elim } 6.2 \)
\( A \leq B \Rightarrow A < B \lor A = B \) 7
assumption
\{cut\}
assumption
equality-substitution 12, 8
\( A \neq B \Rightarrow \neg (A = B) \) 5
\( \neg \text{elim } 13, 14 \)
contra (constructive) 15
\begin{verbatim}
14: 0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0
15: 0 \leq i
16: \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)
17: a[i] \neq 0
18: \text{integer } i_1, \; i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0
19: i \leq i_1
20: i_1 < \text{length}(a)
21: a[i_1] = 0
22: i < i_1 \lor i = i_1
23: i < i_1
24: 0 \leq i + 1
25: i + 1 < \text{length}(a)
26: i + 1 \leq i_1
27: i + 1 \leq i_1 \land i + 1 < \text{length}(a) \land a[i_1] = 0
28: \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
29: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
30: i = i_1
31: a[i] = 0
32: \neg (a[i] = 0)
33: \bot
34: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
35: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
36: 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)
\end{verbatim}
Modifiable Arrays

- Arrays are like functions
- Assigning to an array element is like creating a new function.
- The new function differs from the old in that one element may be different from before.

- Jape Notation: $a \oplus i \rightarrow v$ is the array that is like a except that the value of $a[i]$ is $v$.
- So $(a \oplus i \rightarrow v)[i] = v$, and $(a \oplus i \rightarrow v)[j] = a[j]$ if $j \neq i$. 
Jape’s Indexing rules

- \((a \oplus i \rightarrow v)[i] = v\), and
  \((a \oplus i \rightarrow v)[j] = a[j]\) if \(j \neq i\).
- Two rules below capture the two cases preceding.
  - The first rule simplifies an array modification expression when the index of the new array is provably **the same** as the index to which assignment was done.
  - The second rule simplifies in the case of a **different** index.
- The buttons indicate the direction of substitution.
Array Bounds Guarantees

- If an array index value is used in an assumption, the same index value can be used later on without requiring a bounds check.
- Sub-formula select a hypothesis using the desired index.

Result:

1: \( a[i] = 2 \)
2: \( 0 \leq i \land i < \text{length}(a) \)
Using the Array Rule

- Make sure the entire array sub-expression is sub-formula selected.

- It should match the form in the rule in the menu:

Here we identify:

- A with a
- E with i
- F with \( a[i]+1 \)
- [G] with \([i]\)

(so \( E = G \)).
Using the Equality Rule

- Two selections and a sub-formula selection are needed:
  - Selection an equality hypothesis and a goal.
  - Sub-formula select an instance of the LHS of the equality.
Result of the Equality Rule

The top simple equation can be justified by “obviously”.

\[
\begin{array}{l}
\ldots \\
5: \; 2+1 = 3 \\
6: \; a[i] + 1 = 3 \\
7: \; (a \oplus i \rightarrow a[i] + 1)[i] = 3 \\
8: \; (a \oplus i \rightarrow a[i] + 1)[i] = 3 \land 0 \leq i \land i < \text{length}(a) \\
9: \; a[i] = 2 \rightarrow (a \oplus i \rightarrow a[i] + 1)[i] = 3 \land 0 \leq i \land i < \text{length}(a) \\
10: \; \{(a \oplus i \rightarrow a[i] + 1)[i] = 3 \land 0 \leq i \land i < \text{length}(a)\} (a[i] := a[i] + 1) \{a[i] = 3\} \quad \text{array-element-assignment} \\
11: \; \{a[i] = 2\} (a[i] := a[i] + 1) \{a[i] = 3\} \\
\end{array}
\]
Summary: Jape proof with array modification

1: $a[i] = 2$
2: $0 \leq i \land i < \text{length}(a)$
3: $0 \leq i$
4: $i < \text{length}(a)$
5: $2 + 1 = 3$
6: $a[i] + 1 = 3$
7: $(a \oplus i \rightarrow a[i] + 1)[i] = 3$
8: $(a \oplus i \rightarrow a[i] + 1)[i] = 3 \land 0 \leq i \land i < \text{length}(a)$
9: $a[i] = 2 \rightarrow (a \oplus i \rightarrow a[i] + 1)[i] = 3 \land 0 \leq i \land i < \text{length}(a)$
10: $\{ (a \oplus i \rightarrow a[i] + 1)[i] = 3 \land 0 \leq i \land i < \text{length}(a) \} \{ a[i] := a[i] + 1 \} \{ a[i] = 3 \}$
11: $\{ a[i] = 2 \} \{ a[i] := a[i] + 1 \} \{ a[i] = 3 \}$

infers in-bounds from usage in 1.

A=... rule

index rule

statement
How to get these rules to work in the GUI (It isn’t so obvious.)

- Looking at the 2nd provided array program example, we use sequence, then array-assignment twice (from the bottom up) to get to this point:

```
1: a[i]=0 → (a ⊕ i → a[i] + 1 ⊕ i → (a ⊕ i → a[i] + 1)[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)
2: {(a ⊕ i → a[i] + 1 ⊕ i → (a ⊕ i → a[i] + 1)[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)}
   (a[i] := a[i] + 1){(a ⊕ i → a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)}
3: {a[i] = 0}{a[i] := a[i] + 1}{(a ⊕ i → a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)}
4: {(a ⊕ i → a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)}{a[i] := a[i] + 1}{a[i] = 2}
5: {a[i] = 0}{a[i] := a[i] + 1}{a[i] := a[i] + 1}{a[i] = 2}
```

Provided:
DISTINCT a, i
How to use GUI (continued)

- The top line is pure logic, so we expand using \(\rightarrow\)Introduction and \(\land\)Introduction:

1: \(a[i]=0\)
2: \((a \oplus i \rightarrow a[i]+1 \oplus i \rightarrow a[i]+1)[i]+1)[i]=2\)
3: \(0 \leq i\)
4: \(i < \text{length}(a)\)
5: \((a \oplus i \rightarrow a[i]+1 \oplus i \rightarrow a[i]+1)[i]+1)[i]=2 \land 0 \leq i \land i < \text{length}(a)\)
6: \(a[i]=0 \rightarrow (a \oplus i \rightarrow a[i]+1 \oplus i \rightarrow a[i]+1)[i]+1)[i]=2 \land 0 \leq i \land i < \text{length}(a) \rightarrow \text{intro 1-5}\)
How to use GUI (continued)

- We then conclude the two array index bounds (lines 4, 5) by \( \land \) Elimination, giving:

\[
\begin{align*}
1: & \ a[i] = 0 \\
\ldots & \\
2: & \ (a \oplus i \rightarrow a[i] + 1 \oplus i \leftarrow (a \oplus i \rightarrow a[i] + 1)[i] + 1)[i] = 2 \\
3: & \ 0 \leq i \land i < \text{length}(a) \\
4: & \ 0 \leq i \\
5: & \ i < \text{length}(a) \\
6: & \ (a \oplus i \rightarrow a[i] + 1 \oplus i \leftarrow (a \oplus i \rightarrow a[i] + 1)[i] + 1)[i] = 2 \land 0 \leq i \land i < \text{length}(a)
\end{align*}
\]
How to use GUI (continued)

- We are left with a nested array-modification expression. Carefully sub-formula select the outer array-modification and apply the rule shown (since we have $a \oplus i \rightarrow \ldots[i]$). Do not have anything else (such as a goal) selected.

```
1: a[i]=0
   ...
2: (a \oplus i \rightarrow a[i]+1 \oplus i \rightarrow (a \oplus i \rightarrow a[i]+1)[i]+1)[i]=2
```

giving:

```
1: a[i]=0
   ...
2: (a \oplus i \rightarrow a[i]+1)[i]+1=2
3: (a \oplus i \rightarrow a[i]+1 \oplus i \rightarrow (a \oplus i \rightarrow a[i]+1)[i]+1)[i]=2
```

FROM $E \Rightarrow G$ INFER $(A \oplus E \Rightarrow F)[G]=F$ 2
How to use GUI (continued)

- Repeat the preceding process on the new formula:

\[
\begin{align*}
1: & \quad a[i] = 0 \\
2: & \quad (a \oplus i \cdots a[i] + 1)[i] + 1 = 2 \\
3: & \quad (a \oplus i \cdots a[i] + 1 \oplus i \cdots (a \oplus i \cdots a[i] + 1)[i] + 1)[i] = 2
\end{align*}
\]

giving:

\[
\begin{align*}
1: & \quad a[i] = 0 \\
2: & \quad a[i] + 1 + 1 = 2
\end{align*}
\]
How to use GUI (continued)

- Alternatively we could have selected the *inner* modification expression first:

```plaintext
1: a[i]=0
   ...  
2: (a\oplus i \rightarrow a[i]+1 \oplus i \rightarrow (a\oplus i \rightarrow a[i]+1)[i]+1)[i]=2
3: 0\leq i \land i<\text{length}(a)
```

**giving:**

```plaintext
1: a[i]=0  
   ...  
2: (a\oplus i \rightarrow a[i]+1 \oplus i \rightarrow a[i]+1)[i]=2
3: (a\oplus i \rightarrow a[i]+1 \oplus i \rightarrow (a\oplus i \rightarrow a[i]+1)[i]+1)[i]=2
```

assumption

FROM E\models G INFER (A\oplus E\rightarrow F)[G]=F
FROM E\models G INFER A[G]
How to use GUI (continued)

- (Note that this is different from two slides ago). Then simplify that result:

| 1:  | a[i]=0                     |
|     | ...                       |
| 2:  | (a⊕i→a[i]+1⊕i→a[i]+1+1)[i]=2 |
| 3:  | (a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2 |
| 4:  | 0≤i ∧ i<length(a)         |

giving (as before):

| 1:  | a[i]=0                     |
|     | ...                       |
| 2:  | a[i]+1+1=2                |
| 3:  | (a⊕i→a[i]+1⊕i→a[i]+1+1)[i]=2 |
| 4:  | (a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2 |

assumption

FROM E=G INFERENCE (A⊕E→F)[G]=F 2
FROM E=G INFERENCE (A⊕E→F)[G]=F 3
How to use GUI (continued)

• Use plain equality substitution to simplify the new goal. Note that both the goal and the equation defining the substitution are selected, and the sub-formula for which substitution is to occur is sub-formula selected (3 selections).

\[ a[i] = 0 \]
\[ a[i] + 1 + 1 = 2 \]

\[ a[i] = 0 \]
\[ a[i] + 1 + 1 = 2 \]

\[ a[i] = 0 \]
\[ a[i] + 1 + 1 = 2 \]

\[ a[i] = 0 \]
\[ a[i] + 1 + 1 = 2 \]
Consecutive Array Modification

```
1: a[i]=0
2: 0+1+1=2
3: a[i]+1+1=2
4: (a⊕i→a[i]+1)[i]+1=2
5: (a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2
6: 0≤i ∧ i<length(a)
7: 0≤i
8: i<length(a)
9: (a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2 ∧ 0≤i ∧ i<length(a)
10: a[i]=0→(a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2 ∧ 0≤i ∧ i<length(a) → intro 1–9
11: {(a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2 ∧ 0≤i ∧ i<length(a)}
   (a[i]:=a[i]+1){(a⊕i→a[i]+1)[i]=2 ∧ 0≤i ∧ i<length(a)}
12: {a[i]=0}{a[i]:=a[i]+1}{(a⊕i→a[i]+1)[i]=2 ∧ 0≤i ∧ i<length(a)}
13: {(a⊕i→a[i]+1)[i]=2 ∧ 0≤i ∧ i<length(a)}{a[i]:=a[i]+1}{a[i]=2}
14: {a[i]=0}{a[i]:=a[i]+1}{a[i]:=a[i]+1}{a[i]=2}
```

assumption

obviously

equality-substitution 1, 2

FROM E=G INFERENCE (A⊕E→F)[G]=F 3

FROM E=G INFERENCE (A⊕E→F)[G]=F 4

bounded 1

∧ elim 6

∧ elim 6

∧ intro 5, 7, 8

array-element-assignment

consequence(L) 10, 11

array-element-assignment

sequence 12, 13

---

Provided:

DISTINCT a, i
The following are more detail on an earlier example.

\[
\begin{align*}
\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0) & \land (i = 0) \\
0 \leq i & \land i < \text{length}(a) \land (\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)) & \text{while } a[i] \neq 0 \text{ do } i := i + 1 \text{ od } a[i] = 0
\end{align*}
\]

- Look at part of the invariant here.
- Note that the lower bound on \(x\) is a function of the index \(i\).
- This is important, because it says that the element such that \(a[x] = 0\) is yet to be found.
- We need this invariant to prove termination.
- The loop test will stop when \(a[i] = 0\).

- The expansion order is tricky.
**Key Step #1: Split i \leq i1**

2. \(0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \rightarrow 0 \leq i < \text{length}(a)\)

3. \(0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0\)

4. \(0 \leq i\)

5. \(i < \text{length}(a)\)

6. \(\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)\)

7. \(a[i] \neq 0\)

8. integer i1

9. \(i \leq i1 \land i1 < \text{length}(a) \land a[i1] = 0\)

10. \(i \leq i1\)

11. \(i < i1 \lor i = i1\)

12. \(i1 < \text{length}(a)\)

13. \(a[i1] = 0\)

14. \(0 \leq i1 + 1 \land i1 + 1 < \text{length}(a) \land \exists x. (i1 + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)

15. \(0 \leq i1 + 1 \land i1 + 1 < \text{length}(a) \land \exists x. (i1 + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)

16. \(0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \rightarrow 0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)

Then use ∨Elimination.
Key Step #2: Aim for a contradiction in the $i = i_1$ case.

Now introduce a backward $\neg$Elimination.
Key Step #2, continued:
Key Step #2, continued:
Key Step #2, continued: unify

```
0 ≤ i + 1 ∧ i + 1
i = i + 1

_B1
...

¬ _B1
...

0 ≤ i + 1 ∧ i + 1
0 < i + 1 ∧ i + 1
```

Type a formula to unify with _B1

```
a[i] = 0
```
Key Step #2, continued: use comparison menu to justify \( \neg (a[i] = 0) \)
Key Step #2, concluded: substitute to justify $a[i] = 0$

1. $\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0) \land (i = 0) \land 0 \leq i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)$

2. $0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land 0 \leq i < \text{length}(a)$

3. $0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] = 0$

4. $0 \leq i$

5. $i < \text{length}(a)$

6. $\exists x. (i \leq x < \text{length}(a) \land a[x] = 0)$

7. $a[i] = 0$

8. integer $i_1$

9. $i = i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0$

10. $i = i_1$

11. $i < i_1 \lor i = i_1$

12. $i < \text{length}(a)$

13. $a[i_1] = 0$

14. $i < i_1$

15. $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)$

16. $i = i_1$

17. $a[i] = 0$

18. $\neg (a[i] = 0)$

19. $\bot$

20. $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)$
Status following key step #2

1: $\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0) (i := 0) (0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0))$

\[\ldots\]

2: $0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \rightarrow 0 \leq i \land i < \text{length}(a)$

3: $0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0$

4: $0 \leq i$

5: $i < \text{length}(a)$

6: $\exists x. (i \leq x < \text{length}(a) \land a[x] = 0)$

7: $a[i] \neq 0$

8: integer $i$

9: $i \leq i \land i < \text{length}(a) \land a[i] = 0$

10: $i \leq i$

11: $i < i \land i = i$

12: $i < \text{length}(a)$

13: $a[i] = 0$

14: $i < i$

\[\ldots\]

15: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)$

16: $i = i$

17: $a[i] = 0$

18: $\neg (a[i] = 0)$

19: $\bot$

20: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)$

21: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)$

22: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)$
Completed Proof (lines 1-15)

1. $\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0)$
2. $0 \leq 0$
3. integer 2
4. $0 \leq 2 \land 2 < \text{length}(a) \land a[2] = 0$
5. $0 \leq 2$
6. $2 < \text{length}(a)$
7. $0 < \text{length}(a)$
8. $0 < \text{length}(a)$
9. $0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0$
10. $\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0) \rightarrow 0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0$
11. $\{0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0\} \Rightarrow 0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0$
12. $\{\exists x. (0 \leq x \land x < \text{length}(a) \land a[x] = 0)\} \Rightarrow 0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0$
13. $\{0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0\} \Rightarrow 0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0$
14. $0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0$
15. $0 \leq 0 \land 0 < \text{length}(a) \land a[0] = 0 \rightarrow 0 \leq 0 \land 0 < \text{length}(a)$
Completed Proof (lines 16-39)

16: \(0 \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0\)
17: \(0 \leq i\)
18: \(\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0)\)
19: \(a[i] \neq 0\)
20: integer \(i\)
21: \(i \leq i1 \land i1 < \text{length}(a) \land a[i1] = 0\)
22: \(i = i1\)
23: \(i < i1 \lor i = i1\)
24: \(i1 < \text{length}(a)\)
25: \(a[i1] = 0\)
26: \(i < i1\)
27: \(0 \leq i + 1\)
28: \(i + 1 < \text{length}(a)\)
29: \(i + 1 \leq i1\)
30: \(i + 1 \leq i1 \land i1 < \text{length}(a) \land a[i1] = 0\)
31: \(\exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
32: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
33: \(i = i1\)
34: \(a[i] = 0\)
35: \(\neg (a[i] = 0)\)
36: \(\perp\)
37: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
38: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
39: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
Completed Proof (lines 40-60)

40: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \rightarrow 0 \leq i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)\) — intro 16–39

41: \(\{0 \leq i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)\}\{i := i + 1\}\{0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0)\}\) — variable-assignment consequence(l) 40, 41

42: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] = 0\) — assumption

43: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] = 0\) — \& elim 43

44: \(i < \text{length}(a)\) — obviously from 44

45: \(\text{length}(a) - i > 0\) — intro 43–45

46: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \rightarrow \text{length}(a) - i > 0\)

integer Km

47: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \land \text{length}(a) - i = \text{Km}\)

48: \(\text{length}(a) - i = \text{Km}\)

49: \(\text{length}(a) - (i + 1) < \text{Km}\)

50: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \land \text{length}(a) - i = \text{Km} \rightarrow \text{length}(a) - (i + 1) < \text{Km}\)

51: \(\{\text{length}(a) - (i + 1) < \text{Km}\}\{i := i + 1\}\{\text{length}(a) - i < \text{Km}\}\)

52: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \land \text{length}(a) - i = \text{Km} \rightarrow \text{length}(a) - (i + 1) < \text{Km}\)

53: \(\{0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \land \text{length}(a) - i = \text{Km}\}\{i := i + 1\}\{\text{length}(a) - i < \text{Km}\}\)

54: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0)\) while a[i] \neq 0 do i := i + 1 od

\(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land \neg(a[i] = 0)\)

55: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land \neg(a[i] = 0)\)

56: \(\neg(a[i] = 0)\)

57: \(a[i] = 0\)

58: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land \neg(a[i] \neq 0) \rightarrow a[i] = 0\)

59: \(0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0)\) while a[i] \neq 0 do i := i + 1 od a[i] = 0

60: \(\exists x. (0 \leq x < \text{length}(a) \land a[x] = 0)\) while a[i] \neq 0 do i := i + 1 od a[i] = 0 — Ntuple 12, 59

A = B \& \neg(A = B) 56

\rightarrow intro 55–57

consequence(R) 54, 58