Recall the **while** rule

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

$$\{ I \land P \} \ S \ \{ I \}$$

$$\{ I \} \ \textbf{while}(\ P \ ) \ S \ \{ I \land \neg P \}$$
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.

```latex
\ldots
{i = Ki \land j = Kj \land i \geq 0}(k := 0)
\begin{align*}
1: & \{i \geq 0 \land k + i \times j = K_i \times K_j\} \\
& \text{while } i \neq 0 \text{ do } k := k + j; i := i - 1 \text{ od}
\end{align*}
{k = K_i \times K_j}
```

(What does this code do?)

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.

1: \{i=K_i \land j=K_j \land i \geq 0\}(k:=0)  
   \{i \geq 0 \land k+i \times j=K_i \times K_j\}  
   \ldots

\hline
   \{i \geq 0 \land k+i \times j=K_i \times K_j\}

2: while i \neq 0 do k:=k+j; i:=i-1 od  
   \{k=K_i \times K_j\}

\hline
   \{i=K_i \land j=K_j \land i \geq 0\}(k:=0)  
   \{i \geq 0 \land k+i \times j=K_i \times K_j\}

3: while i \neq 0 do k:=k+j; i:=i-1 od  
   \{k=K_i \times K_j\}

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: \( \{ i = 10 \} \) while \( i > 0 \) do \( i := i - 1 \) od \( \{ i = 10 \land \neg (i > 0) \} \) while 5,8,9-14
16: \( i = 10 \land \neg (i > 0) \rightarrow i = 0 \)
17: \( \{ i = 10 \} \) while \( i > 0 \) do \( i := i - 1 \) od \( \{ i = 0 \} \)

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

- Can the following be proved?

```
\ldots
1: \{y=0 \land i=0 \land n \geq 0\} (j:=1) \{y=i \times i \land i \leq n \land i \geq 0\}
   \text{while } i < n \text{ do } y:=y+j \text{; } j:=j+2 \text{; } i:=i+1 \text{ od}\{y=n \times n\}
```

Provided:

DISTINCT i, j, n, y
The loop invariant is not strong enough to enable induction

- This is more likely provable.

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

Provided:

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

\[
\cdots
\]

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\}

Provided:

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

Initialization

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

First assignment in loop body

12: $\{y = 0 \land i = 0 \land n \geq 0\} (j = 1) \{y = i \times i \land i \leq n \land i \geq 0 \land j = 2 \times i + 1\}$
13: $y = i \times i \land i \leq n \land i \geq 0 \land j = 2 \times i + 1 \land i < n$
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: $y = i \times i \land i \leq n \land i \geq 0 \land j = 2 \times i + 1 \land i < n \land y + j = (i + 1) \times (i + 1) \land i + 1 \leq n \land i + 1 \geq 0 \land j + 2 = 2 \times (i + 1) + 1$
24: $\{y + j = (i + 1) \times (i + 1) \land i + 1 \leq n \land i + 1 \geq 0 \land j + 2 = 2 \times (i + 1) + 1\} (y = y + j) \{y = (i + 1) \times (i + 1) \land i + 1 \leq n \land i + 1 \geq 0 \land j + 2 = 2 \times (i + 1) + 1\}$ variable-assignment

assumption
$\land$ elim 1
$\land$ elim 1
$\land$ elim 1
obviously, from 3, 2
obviously, from 4, 3
obviously, from 3
obviously, from 3
$\land$ intro 5, 6, 7, 8
$\land$ intro 1, 9
variable-assignment
consequence(L) 10, 11
assumption
$\land$ elim 13
$\land$ elim 13
$\land$ elim 13
$\land$ elim 13
obviously, from 16, 14
obviously, from 17
obviously, from 15
obviously, from 16
$\land$ intro 18, 19, 20, 21
$\land$ intro 13−22
variable-assignment
The Completed Proof (lines 25-51 of 51)

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land i &lt; n)</td>
<td>consequence(L) 23, 24</td>
</tr>
<tr>
<td>26</td>
<td>(y = (i+1) \times (i+1) \land i + 1 \leq n \land i + 1 \geq 0 \land j + 2 = 2 \times (i+1) + 1)</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>27</td>
<td>(y = (i+1) \times (i+1) \land i + 1 \leq n \land i + 1 \geq 0 \land j = 2 \times (i+1) + 1)</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>28</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1)</td>
<td>sequence 25, 26, 27</td>
</tr>
<tr>
<td>29</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land i &lt; n)</td>
<td>assumption</td>
</tr>
<tr>
<td>30</td>
<td>(i &lt; n)</td>
<td>(\land) elim 29</td>
</tr>
<tr>
<td>31</td>
<td>(n - i &gt; 0)</td>
<td>obviously, from 30</td>
</tr>
<tr>
<td>32</td>
<td>(y = i \cdot i + 1 \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land n - i &gt; 0)</td>
<td>(\lor) intro 29-31</td>
</tr>
<tr>
<td>33</td>
<td>integer (K_m)</td>
<td>assumption</td>
</tr>
<tr>
<td>34</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land n \land i = K_m)</td>
<td>assumption</td>
</tr>
<tr>
<td>35</td>
<td>(n - i = K_m)</td>
<td>(\land) elim 34</td>
</tr>
<tr>
<td>36</td>
<td>(n - (i+1) &lt; K_m)</td>
<td>obviously, from 35</td>
</tr>
<tr>
<td>37</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land n \land i = K_m \land n - (i+1) &lt; K_m)</td>
<td>(\lor) intro 34-36</td>
</tr>
<tr>
<td>38</td>
<td>(n - (i+1) &lt; K_m)</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>39</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land n \land i = K_m \land y = j + 2 \cdot n - (i+1) &lt; K_m)</td>
<td>consequence(L) 37, 38</td>
</tr>
<tr>
<td>40</td>
<td>(n - (i+1) &lt; K_m)</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>41</td>
<td>(n - (i+1) &lt; K_m)</td>
<td>variable-assignment</td>
</tr>
<tr>
<td>42</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land n \land i = K_m \land y = j + 2 \cdot n - (i+1) \land n - i &lt; K_m)</td>
<td>sequence 39, 40, 41</td>
</tr>
<tr>
<td>43</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land n \land i = K_m \land y = j + 2 \cdot n - (i+1) \land i &lt; n) while (i &lt; n) do (y = j + 2 \cdot n - (i+1) \land i + 1 ) od (y = i \cdot i + 1 \land i + 1 \land j \leq 2 \cdot x + i + 1 \land (i &lt; n))</td>
<td>while 28, 32, 33-42</td>
</tr>
<tr>
<td>44</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land (i &lt; n))</td>
<td>assumption</td>
</tr>
<tr>
<td>45</td>
<td>(y = i \cdot i)</td>
<td>(\land) elim 44</td>
</tr>
<tr>
<td>46</td>
<td>(i + n)</td>
<td>(\land) elim 44</td>
</tr>
<tr>
<td>47</td>
<td>(n + i &gt; 0)</td>
<td>obviously, from 47, 46, 45</td>
</tr>
<tr>
<td>48</td>
<td>(y = n \cdot n)</td>
<td>(\lor) intro 44-48</td>
</tr>
<tr>
<td>49</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land (i &lt; n)) while (i &lt; n) do (y = j + 2 \cdot n - (i+1) \land i + 1 ) od (y = n \cdot n)</td>
<td>consequence(R) 43, 49</td>
</tr>
<tr>
<td>50</td>
<td>(y = i \cdot i + 1 \leq n \land i \geq 0 \land j = 2 \cdot x + i + 1 \land (i &lt; n)) while (i &lt; n) do (y = j + 2 \cdot n - (i+1) \land i + 1 ) od (y = n \cdot n)</td>
<td>Ntuple 12, 50</td>
</tr>
</tbody>
</table>
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.

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

\begin{align*}
&i := 0; \\
&j := n; \\
&\text{while } i < n \text{ do} \\
&\quad \text{if } a[i] = 0 \text{ then } j := i \\
&\quad \text{else skip} \\
&\text{fi} \\
&i := i+1 \\
&\text{od} \\
&\{j < n \rightarrow a[j] = 0\}
\end{align*}

What invariant do we need?
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 \( n \).

\[
\begin{align*}
\{n \geq 0 \land \text{length}(a) = n\} & \quad (i := 0; j := n) \quad \{i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)\} \\
\text{while } i < n \text{ do if } a[i] = 0 \text{ then } j := i \text{ else skip fi; } i := i + 1 \text{ od} \quad \{j < n \rightarrow a[j] = 0\}
\end{align*}
\]

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: \(n \geq 0 \land \text{length}(a) = n \land 0 \leq n \land 0 \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0)\)
12: \(\{0 \leq n \land 0 \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0)\}(i := 0)(i \leq n \land i \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0)\)
13: \(\{i \geq n \land i \geq 0 \land \text{length}(a) = n \land (n < n \rightarrow a[n] = 0)\}(j := n)(i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0))\)
14: \(\{i \geq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)\}(i := n)(i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)\)
15: \(\{n \geq 0 \land \text{length}(a) = n\}(i := 0)(j := n)(i \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)\)
Read-Only Example (lines 16-37)

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. $(a[i] = 0)$
30. $i + 1 \leq n$
31. $i + 1 \geq 0$
32. $i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (i < n \rightarrow a[i] = 0)$
33. $\neg (a[i] = 0) \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (i < n \rightarrow a[i] = 0)$
34. $0 \leq i$
35. $i < \text{length}(a)$
36. $a[i] = 0 \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (i < n \rightarrow a[i] = 0)$
37. $\neg (a[i] = 0) \land i + 1 \leq n \land i + 1 \geq 0 \land \text{length}(a) = n \land (j < n \rightarrow a[j] = 0)$

assumption
elim 16
elim 16
elim 16
elim 16
assumption
obviously, from 20
obviously, from 17
assumption
hyp 21
intro 24–25
intro 22,23,18,26
intro 21–27
assumption
obviously, from 20
obviously, from 17
intro 30,31,18,19
intro 29–32
obviously, from 17
obviously, from 20,18
intro 28,33,34,35
intro 16–36
Read-Only Example (lines 38-47)

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

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

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

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

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

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

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

45: i < n

46: n - i > 0

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

\begin{itemize}
\item variable-assignment
\item skip
\item choice 38,39
\item consequence(1) 37,40
\item variable-assignment
\item sequence 41,42
\item assumption
\item \& elim 44
\item obviously, from 45
\item \rightarrow intro 44–46
\end{itemize}
Read-Only Example (lines 48-74)

48: \[\text{integer } K_m\]
49: \[1 \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \land a[j] = 0) \land i < n \land n_i = K_m\]
50: \[i \geq 0\]
51: \[i < n\]
52: \[n_i = K_m\]
53: \[a[i] = 0\]
54: \[n-(i+1) < K_m\]
55: \[\neg(a[i] = 0)\]
56: \[n-(i+1) < K_m\]
57: \[0 \leq i\]
58: \[a[i] = 0 \land n-(i+1) < K_m\]
59: \[a[i] = 0 \land n-(i+1) < K_m\]
60: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
61: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
62: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
63: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
64: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
65: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
66: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
67: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
68: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
69: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
70: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
71: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
72: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
73: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]
74: \[\neg(a[i] = 0) \land n-(i+1) < K_m\]

Assumption

\(1 \leq n \land i \geq 0 \land \text{length}(a) = n \land (j < n \land a[j] = 0) \land i < n \land n_i = K_m\)

\(i \geq 0\)

\(i < n\)

\(n_i = K_m\)

\(a[i] = 0\)

\(n-(i+1) < K_m\)

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

\(n-(i+1) < K_m\)

\(0 \leq i\)

\(a[i] = 0 \land n-(i+1) < K_m\)

\(a[i] = 0 \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)

\(\neg(a[i] = 0) \land n-(i+1) < K_m\)
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 ...

```
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: 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)
```

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

```
3: 0≤i ∧ i<length(a) ∧ ∃x. (i≤x ∧ x<length(a) ∧ a[x]=0) ∧ a[i]≠0
4: 0≤i
5: i<length(a)
6: ∃x. (i≤x ∧ x<length(a) ∧ a[x]=0)
7: a[i]≠0
8: integer i1, i≤i1 ∧ i1<length(a) ∧ a[i1]=0
   ...
9: 0≤i+1 ∧ i+1<length(a) ∧ ∃x. (i+1≤x ∧ x<length(a) ∧ a[x]=0)
10: 0≤i+1 ∧ i+1<length(a) ∧ ∃x. (i+1≤x ∧ x<length(a) ∧ 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: 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

```
3: 0 ≤ i ∧ i < length(a) ∧ ∃ x. (i ≤ x ∧ x < length(a) ∧ a[x] = 0) ∧ a[i] ≠ 0
4: 0 ≤ i
5: i < length(a)
6: ∃ x. (i ≤ x ∧ x < length(a) ∧ a[x] = 0)
7: a[i] ≠ 0
8: integer i1, i ≤ i1 ∧ i1 < length(a) ∧ a[i1] = 0
9: i ≤ i1
10: i1 < length(a)
11: a[i1] = 0
12: [highlighted] i < i1 ∨ i = i1
   ...
13: 0 ≤ i + 1 ∧ i + 1 < length(a) ∧ ∃ x. (i + 1 ≤ x ∧ x < length(a) ∧ a[x] = 0)
14: 0 ≤ i + 1 ∧ i + 1 < length(a) ∧ ∃ x. (i + 1 ≤ x ∧ x < length(a) ∧ a[x] = 0)
```
Strategy: \( \land \)-Elimination

\begin{verbatim}
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 } i1, i \leq i1 \land i1 < \text{length}(a) \land a[i1] = 0
9: i \leq i1
10: i1 < \text{length}(a)
11: a[i1] = 0
12: i \land i1 \lor i = i1
13: i < i1
... 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)
15: i = i1
... 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)
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)
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)
\end{verbatim}
Upper branch: $\exists$-introduction

\[
i < i_1 \lor i = i_1
\]

\[
i < i_1
\]

\[
\ldots
\]

\[
0 \leq i + 1
\]

\[
\ldots
\]

\[
i + 1 < \text{length}(a)
\]

\[
\ldots
\]

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

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

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: \(i < i_1 \lor i = i_1\)
13: \(i < i_1\)
14: \(0 \leq i + 1\)
15: \(i + 1 < \text{length}(a)\)
16: \(i + 1 \leq i_1\)
17: \(i + 1 \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0\)
18: \(\exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
19: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
Lower branch

```plaintext
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 \(i_1\), \(i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0\)
9: \(i \leq i_1\)
10: \(i_1 < \text{length}(a)\)
11: \(a[i_1] = 0\)
12: \(i < i_1 \lor i = i_1\)
13: \(i < i_1\)
14: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
15: \(i = i_1\)
16: \(0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0)\)
```
Closure of lower branch

5: \( a[i] \neq 0 \)
6: integer \( i1 \), \( i \leq i1 \) \& \( i1 < \text{length}(a) \) \& \( a[i1] = 0 \)
7: \( i \leq i1 \)
8: \( a[i1] = 0 \)
9: \( i < i1 \) \lor \( i = i1 \)
10: \( i < i1 \)
11: \( 0 \leq i + 1 \) \& \( i + 1 < \text{length}(a) \) \& \( \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \)
12: \( i = i1 \)
13: \( a[i] = 0 \)
14: \( \neg (a[i] = 0) \)
15: \( \perp \)
16: \( 0 \leq i + 1 \) \& \( i + 1 < \text{length}(a) \) \& \( \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \)

\( \land \text{elim 3} \)
assumptions
\( \land \text{elim 6.2} \)
\( \land \text{elim 6.2} \)
\( A \equiv B \equiv A \land B \lor A = B \)
7
assumption
\{cut\}
assumption
equality-substitution 12, 8
\( A \equiv B \equiv \neg (A = B) \)
5
\( \neg \text{elim 13, 14} \)
contra (constructive) 15
\[ \begin{align*}
0 & \leq i \land i < \text{length}(a) \land \exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \land a[i] \neq 0 \\
0 & \leq i \\
\exists x. (i \leq x \land x < \text{length}(a) \land a[x] = 0) \\
a[i] & \neq 0 \\
\text{integer } i_1, i \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0 \\
i & \leq i \\
i_1 & < \text{length}(a) \\
a[i_1] & = 0 \\
i & < i_1 \lor i = i_1 \\
i & < i_1 \\
0 & \leq i + 1 \\
i + 1 & < \text{length}(a) \\
i + 1 & \leq i \\
i + 1 & \leq i_1 \land i_1 < \text{length}(a) \land a[i_1] = 0 \\
\exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \\
0 & \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \\
i & = i_1 \\
a[i] & = 0 \\
\neg (a[i] = 0) \\
\bot \\
0 & \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \\
0 & \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \\
0 & \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x \land x < \text{length}(a) \land a[x] = 0) \\
\end{align*} \]
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.

```
1: a[i]=2
2: 0\leq i \land i<\text{length}(a)
3: 0\leq i
4: i<\text{length}(a)
5: a[i]+1=3
6: (a[i] \iff a[i]+1)[i]=3
```

```
A=A
A=.. 
..=B
obviously
boundedness from (in)equality
```

FROM E=G INFERENCE (A \iff E \iff F) [5]
Result of the Equality Rule

\[
\begin{array}{l}
\ldots \\
5: & 2+1=3 \\
6: & a[i]+1=3 \\
7: & (a\oplus i\rightarrow a[i]+1)[i]=3 \\
8: & (a\oplus i\rightarrow a[i]+1)[i]=3 \land 0 \leq i \land i < \text{length}(a) \\
9: & a[i]=2 \rightarrow (a\oplus i\rightarrow a[i]+1)[i]=3 \land 0 \leq i \land i < \text{length}(a) \\
10: & \{(a\oplus i\rightarrow a[i]+1)[i]=3 \land 0 \leq i \land i < \text{length}(a)\}(a[i]:=a[i]+1)\{a[i]=3\} \text{ array-element-assignment} \\
11: & \{a[i]=2\}(a[i]:=a[i]+1)\{a[i]=3\} \text{ consequence(L) 9,10}
\end{array}
\]

The top simple equation can be justified by “obviously”.
Summary: Jape proof with array modification

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>( a[i]=2 )</td>
</tr>
<tr>
<td>2:</td>
<td>( 0 \leq i \land i &lt; \text{length}(a) )</td>
</tr>
<tr>
<td>3:</td>
<td>( 0 \leq i )</td>
</tr>
<tr>
<td>4:</td>
<td>( i &lt; \text{length}(a) )</td>
</tr>
<tr>
<td>5:</td>
<td>( 2 + 1 = 3 )</td>
</tr>
<tr>
<td>6:</td>
<td>( a[i]+1 = 3 )</td>
</tr>
<tr>
<td>7:</td>
<td>( (a \oplus i \rightarrow a[i]+1)[i] = 3 )</td>
</tr>
<tr>
<td>8:</td>
<td>( (a \oplus i \rightarrow a[i]+1)[i] = 3 \land 0 \leq i \land i &lt; \text{length}(a) )</td>
</tr>
<tr>
<td>9:</td>
<td>( a[i]=2 \rightarrow (a \oplus i \rightarrow a[i]+1)[i] = 3 \land 0 \leq i \land i &lt; \text{length}(a) )</td>
</tr>
<tr>
<td>10:</td>
<td>( { (a \oplus i \rightarrow a[i]+1)[i] = 3 \land 0 \leq i \land i &lt; \text{length}(a) } { a[i]:=a[i]+1 } { a[i]=3 } )</td>
</tr>
<tr>
<td>11:</td>
<td>( { a[i]=2 } { a[i]:=a[i]+1 } { a[i]=3 } )</td>
</tr>
</tbody>
</table>

**infers in-bounds from usage in 1.**

A=... rule

index rule

assumption
bounded 1
\& elim 2
\& elim 2
obviously
equality-substitution 1,5
FROM \( E \Rightarrow G \) INFERENCE (\( A \oplus E \rightarrow F \)) [G] = F 6
\& intro 7,3,4
→ intro 1–8
array-element-assignment
consequence(L) 9,10
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:

```plaintext
1: a[i]=0 → (a ⊕ a[i] + 1) ⊕ (a ⊕ a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)
2: (a ⊕ a[i] + 1) ⊕ (a ⊕ a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)
   (a[i] := a[i] + 1) ⊕ (a ⊕ a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)
array-element-assignment
   array-element-assignment
3: {a[i] = 0} a[i] := a[i] + 1 ∧ (a ⊕ a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a)
   consequence(L) 1, 2
4: (a ⊕ a[i] + 1)[i] = 2 ∧ 0 ≤ i ∧ i < length(a) {a[i] := a[i] + 1} {a[i] = 2}
   array-element-assignment
5: {a[i] = 0} a[i] := a[i] + 1 ∧ a[i] := a[i] + 1 {a[i] = 2}
   sequence 3, 4
```
How to use GUI (continued)

- The top line is pure logic, so we expand using →Introduction and ∧Introduction:

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

5: (a⊙i→a[i]+1⊙i→(a⊙i→a[i]+1)[i]+1)[i]=2 ∧0≤i∧i<length(a)

6: a[i]=0→(a⊙i→a[i]+1⊙i→(a⊙i→a[i]+1)[i]+1)[i]=2 ∧0≤i∧i<length(a) → intro 1–5
How to use GUI (continued)

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

\begin{align*}
1: & \quad a[i] = 0 \\
2: & \quad (a \oplus i \rightarrow a[i] + 1 \oplus i \rightarrow (a \oplus i \rightarrow a[i] + 1)[i] + 1)[i] = 2 \\
3: & \quad 0 \leq i \wedge i < \text{length}(a) \\
4: & \quad 0 \leq i \\
5: & \quad i < \text{length}(a) \\
6: & \quad (a \oplus i \rightarrow a[i] + 1 \oplus i \rightarrow (a \oplus i \rightarrow a[i] + 1)[i] + 1)[i] = 2 \wedge 0 \leq i \wedge 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 \(a \oplus i \rightarrow ...[i]\)). Do not have anything else (such as a goal) selected.

\[
\{ a[i]=0 \mid a[i]==a[i]+1 \mid a[i]==a[i]+1 \mid a[i]=2 \ [1] \}
\]

\[
\text{Indexing}
FROM E=G INFERS (A \oplus E \rightarrow F)[G]=F
\]

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

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

giving:

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

FROM E=G INFERS (A \oplus E \rightarrow F)[G]=F 2

assumption
How to use GUI (continued)

• Repeat the preceding process on the new formula:

\[
a[i] = 0 \\
\ldots \\
(a \oplus i \rightarrow a[i] + 1)[i] + 1 = 2 \\
(a \oplus i \rightarrow a[i] + 1 \oplus i \rightarrow (a \oplus i \rightarrow a[i] + 1)[i] + 1)[i] = 2 \\
\]

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

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

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

**giving:**

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

**assumption**

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

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1:</td>
<td>a[i]=0</td>
<td></td>
</tr>
<tr>
<td>2:</td>
<td>(a⊕i→a[i]+1⊕i→a[i]+1+1)[i]=2</td>
<td></td>
</tr>
<tr>
<td>3:</td>
<td>(a⊕i→a[i]+1⊕i→(a⊕i→a[i]+1)[i]+1)[i]=2</td>
<td></td>
</tr>
<tr>
<td>4:</td>
<td>0≤i∧i&lt;length(a)</td>
<td></td>
</tr>
</tbody>
</table>

giving (as before):
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).

giving:

1: \( a[i] = 0 \)
2: \( a[i] + 1 + 1 = 2 \)

assumption

equality-substitution 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: \( 0 \leq i < \text{length}(a) \)
7: \( 0 \leq i \)
8: \( i < \text{length}(a) \)
9: \( (a+i->a[i]+1+i->(a+i->a[i]+1)[i]+1)[i]=2 \land 0 \leq i < \text{length}(a) \)

0: \( a[i]=0->(a+i->a[i]+1+i->(a+i->a[i]+1)[i]+1)[i]=2 \land 0 \leq i < \text{length}(a) \) \rightarrow \text{intro 1-9}

1: \{a[i]=0\}(a[i]:=a[i]+1)\{(a+i->a[i]+1)[i]=2 \land 0 \leq i < \text{length}(a)\}

array-element-assignment

2: \{a[i]=0\}(a[i]:=a[i]+1)\{(a+i->a[i]+1)[i]=2 \land 0 \leq i < \text{length}(a)\}

array-element-assignment

3: \{(a+i->a[i]+1)[i]=2 \land 0 \leq i < \text{length}(a)\}(a[i]:=a[i]+1)(a[i]=2)

4: \{a[i]=0\}(a[i]:=a[i]+1,a[i]:=a[i]+1)(a[i]=2)

sequence 12,13

---

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

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

- 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.
Key Step #1: Split $i \leq i_1$

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

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 $i_1$

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

10: $i \leq i_1$

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

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

13: $a[i_1] = 0$

...  

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

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

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

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

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

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

10: $i \leq i$

11: $i < i \lor i = i$

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

13: $a[i] = 0$

14: $i < i$

...  

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: $\perp$

20: $0 \leq i + 1 \land i + 1 < \text{length}(a) \land \exists x.(i + 1 \leq x < \text{length}(a) \land a[x] = 0)$
Key Step #2, concluded:
substitute to justify $a[i] = 0$
Status following key step #2

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

\[ \vdots \]

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

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

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

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

\[ a[i] = 0 \]

\[ \vdots \]

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

\[ i := i \]

\[ a[i] = 0 \]

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

\[ \bot \]

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

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

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

\[ \exists \text{elim } 6,8,21 \]

\[ \land \text{elim } 3 \]

\[ \land \text{elim } 3 \]

\[ \land \text{elim } 3 \]

\[ \land \text{elim } 3 \]

\[ A \leq B \land A \leq B \lor A = B \land \text{elim } 9 \]

\[ \land \text{elim } 9 \]

\[ \land \text{elim } 9 \]

\[ \text{assumption} \]

\[ \text{equality-substitution } 16,13 \]

\[ A \leq B \land \neg (A = B) \land \text{elim } 17,18 \]

\[ \lor \text{elim } 11,14-15,16-20 \]

\[ \land \text{elim } 6,8,21 \]
1. \( \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0) \)
2. \( 0 \leq 0 \)
3. integer i2
4. \( 0 \leq i2 \land i2 < \text{length(a)} \land a[i2] = 0 \)
5. \( 0 \leq i2 \)
6. \( i2 < \text{length(a)} \)
7. \( 0 < \text{length(a)} \)
8. \( 0 < \text{length(a)} \)
9. \( \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0) \)
10. \( \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0) \rightarrow 0 \leq 0 \land 0 < \text{length(a)} \land \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0) \)
11. \( \{\text{0} \leq i < \text{length(a)} \land \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0)\} (i := 0) \{\text{0} \leq i \land i < \text{length(a)} \land \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0)\} \)
12. \( \exists x. (\text{0} \leq i \land i < \text{length(a)} \land a[i] = 0) (i := 0) \{\text{0} \leq i \land i < \text{length(a)} \land \exists x. (\text{0} \leq x \land x < \text{length(a)} \land a[x] = 0)\} \)
13. \( 0 \leq i < \text{length(a)} \land \exists x. (\text{0} \leq x < \text{length(a)} \land a[x] = 0) \)
14. \( 0 \leq i < \text{length(a)} \)
15. \( 0 \leq i \land i < \text{length(a)} \land \exists x. (\text{0} \leq x < \text{length(a)} \land a[x] = 0) \rightarrow 0 \leq i \land i < \text{length(a)} \)

assumption
obviously
assumption
assumption
\& elim 4
\& elim 4
obviously, from 6.5
\exists elim 1,3–7
\& intro 2.8.1
\rightarrow intro 1–9
variable-assignment
consequence(L) 10,11
assumption
\& elim(L) 13
\rightarrow intro 13–14
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 i1
21: \[i \leq i1 \land i1 < \text{length}(a) \land a[i1] = 0\]
22: \[i \leq i1\]
23: \[i1 \land i = i1\]
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 \land i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0)

41: \{ 0 \leq i + 1 < \text{length}(a) \land \exists x. (i + 1 \leq x < \text{length}(a) \land a[x] = 0) \mid i = i + 1 \mid 0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \}

42: 0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0) \land a[i] = 0 \land i = i + 1 \mid 0 \leq i < \text{length}(a) \land \exists x. (i \leq x < \text{length}(a) \land a[x] = 0)

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

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

45: \text{length}(a) - i > 0

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

47: \text{integer} \ K_m

48: 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 = K_m

49: \text{length}(a) - i = K_m

50: \text{length}(a) - (i + 1) < K_m

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

52: \{ \text{length}(a) - (i + 1) < K_m \mid i = i + 1 \mid \text{length}(a) - i < K_m \}

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 = K_m \mid i = i + 1 \mid \text{length}(a) - i < K_m \}

54: \{ 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 i = i + 1 \land \text{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] = 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) \land a[i] \neq 0 \rightarrow i = i + 1 \land \text{od} \land a[i] = 0 \}

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