
Rule 0.8 *(Use Fixed-Size Arrays Instead of Dynamic Arrays)*

<pre> [...] contract A { T[] arr; [...] function f(pds) { [...] arr.push(value); [...] } [...] } </pre>	=	<pre> [...] contract A' { T[N] arr; uint length; [...] function f(pds) { [...] require(length < N); arr[length] = value; length++; [...] } [...] } </pre>
---	---	---

where

arr is a dynamic array of type $T[]$ in contract A ;

T is the element type of the array;

N is a compile-time constant representing the maximum array size;

length is a state variable tracking the current number of elements in the fixed-size array;

value is the element being added to the array;

pds are the parameter declarations of function f .

provided

The maximum number of elements that will be stored in the array is known and equals N ;

All array access operations in A respect the bound $length < N$ in A' ;

Array operations in A (e.g., *push*, *pop*) are replaced with explicit index operations and bounds checking in A' ;

The variable *length* is properly maintained to reflect the current number of valid elements;

Array accesses use *length* for bounds validation in A' .

Invariant:

Let s_i and s'_i be the initial state of A and A' , respectively.

Let s_f and s'_f be the state reached by A and A' , respectively, after $A.f()$ and $A'.f()$ are executed from s_i and s'_i , respectively.

Then, the coupling invariant is

$$\forall s_i, s'_i . (s_i = s'_i) \rightarrow (s_f = s'_f)$$
