Rule 0.26 ⟨*Cache Array Length in Loops*⟩

```
[...]                                              [...]
contract A {                                       contract A′ {
  [...]                                              [...]
  function f(pds) {                                  function f(pds) {
    [...]                                              [...]
    for(init; i < arr.length; upd) {        =          uint len = arr.length;
      stmts                                            for(init; i < len; upd) {
    }                                                    stmts
    stmts′                                             }
  }                                                    stmts′
  [...]                                              }
}                                                    [...]
                                                   }
```

where

arr is an array whose length is accessed in the loop condition;

arr.length is the array length property accessed in each loop iteration;

len is a local variable that caches the array length;

init and *upd* are the loop initialization and update expressions;

i is the loop index variable;

stmts represents the loop body statements;

pds are the parameter declarations of function f;

stmts′ represents statements following the loop.

provided

The array length is accessed in the loop condition on every iteration;

The array *arr* is not modified within the loop in a way that changes its length;

No operations within the loop (e.g., **push**, **pop**) alter the array size;

Caching the length does not affect the correctness of the loop;

The cached length remains valid throughout loop execution.

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

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