Rule 0.25 ⟨*Cache Array Member Variables*⟩

```
[...]
contract A {
  [...]
  function f(pds) {
    [...]
    for(init; cond; upd) {
      stmts[arr[i]]
    }
    stmts'
  }
  [...]
}
```

=

```
[...]
contract A' {
  [...]
  function f(pds) {
    [...]
    for(init; cond; upd) {
      T cache = arr[i];
      stmts[cache]
    }
    stmts'
  }
  [...]
}
```

where

arr is an array (storage or memory) accessed within the loop;

$arr[i]$ is an array element accessed multiple times in the loop body;

cache is a local variable of type T (reference type for storage, value type for memory) that caches $arr[i]$;

T is the type of the array elements;

$stmts[arr[i]]$ represents loop body statements that access $arr[i]$ multiple times;

$stmts[cache]$ represents the same statements with $arr[i]$ replaced by *cache*;

init, *cond*, and *upd* are the loop initialization, condition, and update expressions;

pds are the parameter declarations of function f;

$stmts'$ represents statements following the loop.

provided

The array element $arr[i]$ is accessed multiple times within the same loop iteration;

For storage arrays, use **storage** keyword to cache references; for memory arrays, cache values;

The cached reference or value maintains consistency throughout the iteration;

No operations within the loop invalidate the cached reference (e.g., array resizing);

The caching does not introduce race conditions or affect correctness.

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

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