

**Rule 0.21** *[Avoid Nested Loops](#)*

<pre>[...] <b>contract</b> A {     [...]     <b>function</b> f(pds) {         [...]         <b>for</b>(init<sub>1</sub>; cond<sub>1</sub>; upd<sub>1</sub>) {             <b>for</b>(init<sub>2</sub>; cond<sub>2</sub>; upd<sub>2</sub>) {                 stmts             }         }         stmts'     }     [...] }</pre>	$=$	<pre>[...] <b>contract</b> A' {     [...]     <b>function</b> f(result, pds) {         [...]         <b>for</b>(init; cond; upd) {             stmts"         }         stmts'     }     [...] }</pre>
--	-----	--

where

The outer loop has initialization  $init_1$ , condition  $cond_1$ , and update  $upd_1$ ;  
 The inner loop has initialization  $init_2$ , condition  $cond_2$ , and update  $upd_2$ ;  
 $stmts$  represents the nested loop body statements;  
 $stmts'$  represents statements following the nested loops;  
 $stmts''$  represents the optimized single-loop body;  
 $result$  is a pre-computed parameter or auxiliary data structure used to avoid nesting;  
 $pds$  are the parameter declarations of function  $f$ .

provided

The nested loop computation can be restructured into a single loop using auxiliary data structures (e.g., mappings, arrays);

Alternatively, the result can be pre-computed off-chain and passed as a parameter;

The transformation maintains computational equivalence while reducing complexity from  $O(n \times m)$  to  $O(n + m)$  or  $O(n)$ ;

When using mappings for lookups, initialization occurs in a separate function or off-chain;

The optimized version produces the same final state as the nested loop version.

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

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