
Rule 0.29 *⟨Use Unchecked Arithmetic for Validated Operations⟩*

<pre>[...] contract A { [...] function f(pds) { [...] if (condition) { revert Error(); } result = expr₁ ± expr₂; stmts } [...] }</pre>	=	<pre>[...] contract A' { [...] function f(pds) { [...] if (condition) { revert Error(); } unchecked { result = expr₁ ± expr₂; } stmts } [...] }</pre>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

where

- result* is a variable storing the result of an arithmetic operation;
- expr₁* and *expr₂* are expressions involved in the arithmetic operation;
- ± represents arithmetic operators (+, −, *, /, %, etc.);
- condition* is a validation that prevents overflow/underflow;
- stmts* represents the sequence of statements following the arithmetic operation;
- pds* are the parameter declarations of function *f*.

provided

- Prior validation ensures that overflow/underflow cannot occur;
- The condition check guarantees safe arithmetic bounds before the operation;
- The arithmetic operation is immediately preceded by validation logic;
- Solidity version ≥ 0.8.0 is used (where overflow checks are enabled by default);
- The unchecked block only contains arithmetic operations that are provably safe;
- No external calls or state changes occur between validation and arithmetic operation.

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

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