
Rule 0.26 *⟨Use Efficient Loop Increment⟩*

<pre> [...] contract A { [...] function f(pds) { [...] for(init; cond; i += 1) { stmts } stmts' } [...] } </pre>	=	<pre> [...] contract A' { [...] function f(pds) { [...] for(init; cond; ++i) { stmts } stmts' } [...] } </pre>
--	---	--

where

- i is the loop counter variable;
- $i += 1$ is the addition assignment increment operation;
- $++i$ is the pre-increment operator;
- $init$ and $cond$ are the loop initialization and condition expressions;
- $stmts$ represents the loop body statements;
- pds are the parameter declarations of function f ;
- $stmts'$ represents statements following the loop.

provided

- The loop uses $i += 1$ for incrementing the counter;
- The pre-increment operator $++i$ produces equivalent behavior;
- The increment operation occurs only in the loop update expression;
- No side effects depend on the specific increment method used;
- The pre-increment generates more efficient bytecode than addition assignment.

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

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