Rule 0.16 ⟨*Use Short-Circuiting for Conditional Expressions*⟩

```
[...]
contract A {
  [...]
  function f(pds) {
    [...]
    temp = $expr_2$;
    if ($expr_1$ op temp) {
      stmts
    }
    stmts'
  }
  [...]
}
```

=

```
[...]
contract A' {
  [...]
  function f(pds) {
    [...]
    if ($expr_1$ op $expr_2$) {
      stmts
    }
    stmts'
  }
  [...]
}
```

where

$expr_1$ is a less expensive boolean expression (cheap check);

$expr_2$ is a more expensive boolean expression (expensive check);

op is a short-circuiting logical operator ($\wedge\wedge$ for AND or $\vee\vee$ for OR);

temp is a temporary variable storing the result of $expr_2$;

pds are the parameter declarations of function f;

stmts and *stmts'* represent statement sequences.

provided

For $op = \wedge\wedge$: $expr_1$ should be the cheaper condition that fails early;

For $op = \vee\vee$: $expr_1$ should be the cheaper condition that succeeds early;

The expression $expr_2$ is only evaluated when necessary due to short-circuit evaluation;

The expressions $expr_1$ and $expr_2$ have no side effects that affect each other;

The order of evaluation does not affect the correctness of the program;

$expr_2$ is more expensive (in gas) than $expr_1$.

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

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