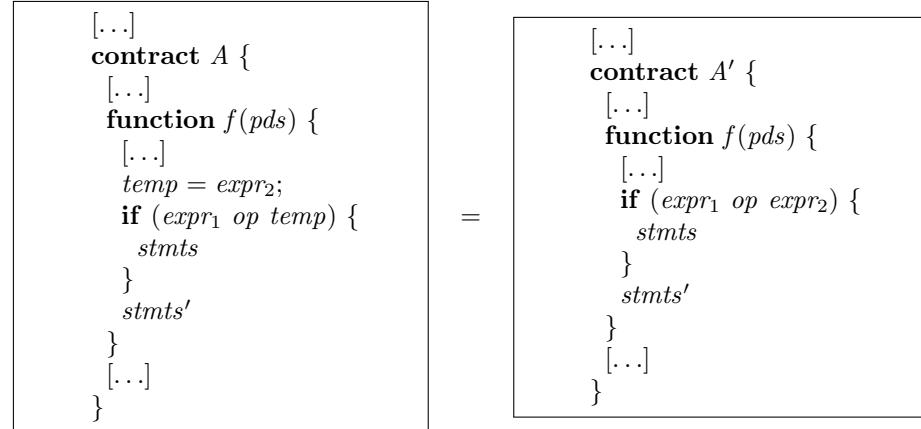

Rule 0.16 *(Use Short-Circuiting for Conditional Expressions)*


where

expr₁ is a less expensive boolean expression (cheap check);
expr₂ 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 *expr₂*;
pds are the parameter declarations of function *f*;
stmts and *stmts'* represent statement sequences.

provided

For *op* = $\wedge\wedge$: *expr₁* should be the cheaper condition that fails early;
For *op* = $\vee\vee$: *expr₁* should be the cheaper condition that succeeds early;
The expression *expr₂* is only evaluated when necessary due to short-circuit evaluation;
The expressions *expr₁* and *expr₂* have no side effects that affect each other;
The order of evaluation does not affect the correctness of the program;
expr₂ is more expensive (in gas) than *expr₁*.

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

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