Rule 0.31 ⟨*Redundant Control Flow Removal (Break to Early Return)*⟩

```
[...]
contract A {
 [...]
 function f(pds)  {
  [...]
  T result = defaultVal;
  for(init; cond; update) {
   if(breakCond) {
    result = value;
    break;
   }
   stmts
  }
  return result;
 }
 [...]
}
```

=

```
[...]
contract A′ {
 [...]
 function f(pds)  {
  [...]
  for(init; cond; update) {
   if(breakCond)
    return value;
   stmts
  }
  return defaultVal;
 }
 [...]
}
```

where

breakCond is a boolean condition that triggers early loop termination;

T is the return type of function f;

result is a local variable of type T used to store the return value;

value is the expression assigned to *result* when *breakCond* is true;

defaultVal is the default return value when the loop completes normally;

init, *cond*, and *update* are the loop initialization, condition, and update expressions;

stmts represents the remaining statements in the loop body;

 is the return type declaration.

provided

The only purpose of *result* is to store a value for return after the loop;

result is not read or modified elsewhere in the function after its initialization;

The **break** statement is the only statement after the assignment to *result* within the if block;

No statements exist after *stmts* and before the end of the loop iteration;

No cleanup or finalization code exists between the loop and the return statement;

breakCond, *value*, and *defaultVal* have no side effects that affect program state;

stmts does not modify *result*.

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.

Let *retVal* and *retVal′* be the values returned by $A.f()$ and $A'.f()$, respectively.

Then, the coupling invariant is

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