Rule 0.4 ⟨*Refactoring Loops with Repeated Computations*⟩

```
[...]
contract A {
  [...]
  function f(pds) {
    [...]
    for(init; cond; update) {
      stmts[expr]
    }
    stmts'
  }
  [...]
}
```

=

```
[...]
contract A' {
  [...]
  function f(pds) {
    [...]
    T local = expr;
    for(init; cond; update) {
      stmts[local]
    }
    stmts'
  }
  [...]
}
```

where

expr is an expression that is loop-invariant (yields the same result in every iteration);

T is the type of the computed expression *expr*;

local is a local variable of type T used to cache the computation result;

stmts[*expr*] represents statements inside the loop that use *expr*;

stmts[*local*] represents the same statements with *expr* replaced by *local*;

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

stmts′ represents statements following the loop.

provided

expr does not depend on the loop variable or any value modified within the loop;

expr is side-effect free (does not modify state or call external functions);

The value of *expr* remains constant throughout all loop iterations;

No variable in *expr* is modified between the assignment to *local* and the loop execution.

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

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