Rule 0.5 ⟨*Struct Packing*⟩

```
[...]
contract $A$ {
  [...]
  struct $S$ {
    $T_1$ $\textit{field}_1$;
    $T_2$ $\textit{field}_2$;
    ...
    $T_n$ $\textit{field}_n$;
  }
  [...]
}
```

=

```
[...]
contract $A'$ {
  [...]
  struct $S$ {
    $T_{\pi(1)}$ $\textit{field}_{\pi(1)}$;
    $T_{\pi(2)}$ $\textit{field}_{\pi(2)}$;
    ...
    $T_{\pi(n)}$ $\textit{field}_{\pi(n)}$;
  }
  [...]
}
```

where

S is a struct type defined in the contract;

$\textit{field}_i$ are the fields of the struct with types T_i for $i = 1, \ldots, n$;

π is a permutation of $\{1, 2, \ldots, n\}$ that reorders the fields;

T_i represents types of varying sizes (e.g., **uint256**, **uint8**, **bool**, **address**).

provided

The permutation π optimally packs fields to minimize the number of storage slots used;

Fields with combined size ≤ 32 bytes are grouped together in the reordered struct;

The reordering does not affect the logical semantics of the contract;

All field accesses in the contract remain semantically equivalent after reordering.

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

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