

# fn then\_index\_or\_default

Michael Shoemate

This proof resides in “**contrib**” because it has not completed the vetting process.

Proves soundness of the implementation of `then_index_or_default` in `mod.rs` at commit `f5bb719` (outdated<sup>1</sup>).

This postprocessor indexes into a vector or returns the default value of the type if the index does not exist.

## 1 Hoare Triple

### Precondition

#### Compiler-Verified

- Generic `T` implements trait `Default`

#### User-Verified

None

### Pseudocode

```
1 def then_index_or_default(  
2   index: usize,  
3 ) -> Function[Vec[T], T]:  
4   return Function.new(lambda x: x[index] if index < len(x) else T.default())
```

### Postcondition

**Theorem 1.1.** For every setting of the input parameters (`index`, `T`) to `then_index_or_default` such that the given preconditions hold, `then_index_or_default` raises an exception (at compile time or run time) or returns a valid postprocessor. A valid postprocessor has the following property:

1. (Data-independent errors). For every pair of elements  $x, x'$  in `input_domain`, `function(x)`, `function(x')` either neither or both raise an error. If both raise an error, then they both raise the same error.

*Proof.* The function is infallible, so the function satisfies the data-independent errors property. Therefore the postcondition is satisfied.  $\square$

---

<sup>1</sup>See new changes with `git diff f5bb719..c7d74e0 rust/src/transformations/scalar_to_vector/mod.rs`