

# Benchmarking ADT struct representation vs Vector struct representation

This notebook compares the representation of structures on Boogie level as:

- Vectors of the universal `$Value` type. Values for all fields are represented in boxed (`$Value`) representation. Selecting and updating fields amounts to vector indexing. On select/update values need to be unboxed/boxed.
- Abstract data types. Values of fields are stored in unboxed representation unless their type is generic. Equality on universal values has to be implemented by a large case distinction of the multiple ADT variants. However, equality is extensional unless a struct contains a transient field of vector type, which breaks extensionality.

## Preparation

Load the prover-lab crate. This may take *long* (minutes) the first time the Jupyter server is started because it compiles a lot Rust sources.

```
In [2]: :sccache 1
        :dep prover-lab = { path = "../.." }
```

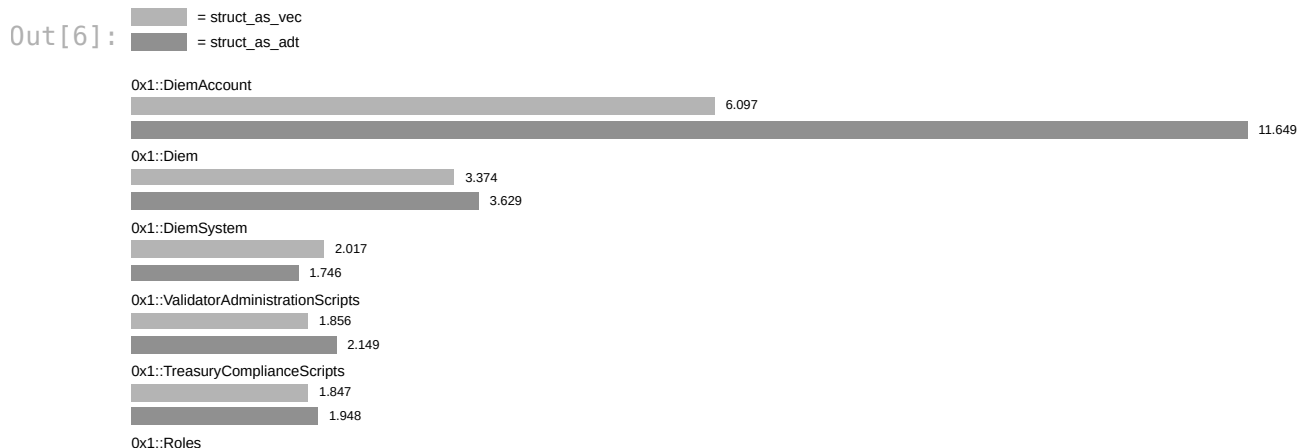
Out[2]: sccache: true

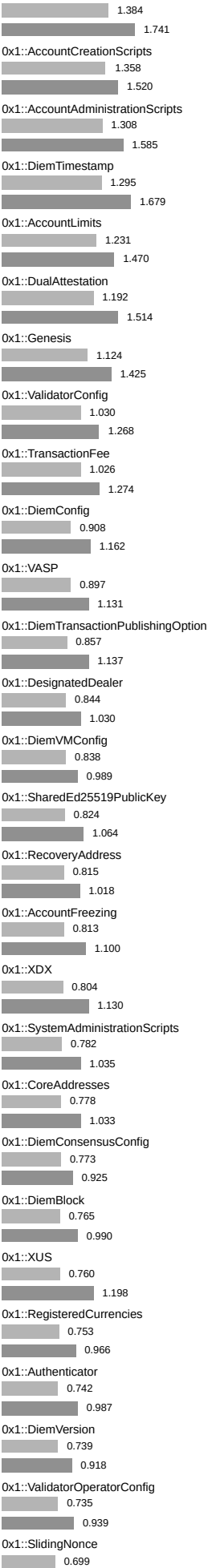
Make functions from the benchmark module available:

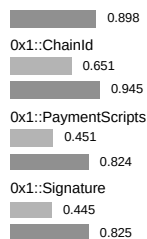
```
In [3]: use prover_lab::benchmark::*;
```

## Module Verification Time

```
In [6]: let mut struct_as_vec_mod = read_benchmark("struct_as_vec.mod_data");
        let mut struct_as_adt_mod = read_benchmark("struct_as_adt.mod_data");
        struct_as_vec_mod.sort(); // Will also determine order of other samples.
        plot_benchmarks(&[&struct_as_vec_mod, &struct_as_adt_mod])
```







## Top 20 by Function

In [7]:

```
let mut struct_as_vec_fun = read_benchmark("struct_as_vec.fun_data"?;
let mut struct_as_adt_fun = read_benchmark("struct_as_adt.fun_data"?;
struct_as_vec_fun.sort(); // Will also determine order of other samples.
struct_as_vec_fun.take(20);
plot_benchmarks(&[&struct_as_vec_fun, &struct_as_adt_fun])
```

Out[7]:



In [ ]: