fn truncate_domain

Michael Shoemate

September 9, 2025

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

Proves soundness of `truncate_domain` in `mod.rs` at commit f5bb719 (outdated[1]).

1 Hoare Triple

Precondition

Caller Verified

Truncation keys are stable row-by-row transformations of the data.

Function

```
def truncate_domain(
    domain: DslPlanDomain,
    truncation: Truncation,
) -> DslPlanDomain:
    match truncation:
        case Truncation.Filter(_):
            for m in domain.margins:
                # After filtering you no longer know partition lengths or keys.
                m.invariant = None  #
            # Upper bounds on the number of rows and groups remain valid.
            return domain
        case Truncation.GroupBy(keys, aggs):
            for agg in aggs:
                # True means resize is allowed
                check_infallible(agg, True)  #

            def with_truncation(lf):
                return lf.group_by(keys).agg(aggs)

            def without_invariant(m):
                m.invariant = None
                return m

            # derive new output domain
            return FrameDomain.new_with_margins(
                [  #
                    Seriesdomain.new_from_field(f)
                    for f in domain.simulate_schema(with_truncation)
                ],
                margins=[
```

[1] See new changes with `git diff f5bb719..69324fc rust/src/transformations/make_stable_lazyframe/truncate/mod.rs`

```
                    # discard invariants as multiverses are mixed
                    without_invariant(m.clone())
                    for m in domain.margins
                    # only keep margins that are a subset of the truncation keys
                    if m.by.is_subset(HashSet.from_iter(keys))
                ],
            )
```

Postcondition

Theorem 1.1 (Postcondition)**.** Define `functionas` the function that takes in a dataset `arg` and returns a dataset with `truncation` applied to it. `truncation` is either a filter or groupby operation.

- If `truncation` is a filter, then `function` returns a subset of rows in the data.
- If `truncation` is a groupby operation, then `function` groups by `truncation.keys` and the identifier, and aggregates by `truncation.aggs`.

For every element x in `domain`, `function`(x) is in the returned domain or raises a data-independent runtime exception.

Proof. First consider the case where `truncation` is a filter. Since filtering is a contractive mapping, the output domain is a subset of the input domain. However, any invariants on the group length and keys are no longer valid. This is reflected in line 9.

Now consider the case where `truncation` is a groupby operation. Since each group becomes one row in the resulting dataset, an arbitrary black-box function may be applied to each group, so long as erors from the black-box function (the aggregates) are data-independent. Line 15 ensures that the aggregates are infallible.

The schema of members of the output domain is determined by the keys and aggregates of the groupby operation, which is computed on line 26. In this case, only margins that are a subset of the grouping keys remain valid, as any bounds on group lengths are not preserved through the black-box aggregation function. Among preserved margins, invariants are discarded. □