1 Praktikum - Spezifikation und Verifikation

1.1 Introduction

The goal of this practical course was to prove the correctness of the `cc_explain` algorithm, which was defined in my Bachelor's thesis [Ghi22]. This function takes as input two elements of the congruence closure and it returns the labels of the proof forest that explain why the two elements are congruent. The `cc_explain` function uses an additional union-find that stores which edges in the proof forest have already been examined by the function. Because of the additional union-find, it was difficult to use the induction hypothesis in order to directly prove the correctness of the function.

Therefore, I proved it by defining the auxiliary function `cc_explain2` [`CC_Explain2_Definition.thy`], which is identical to `cc_explain`, except that it does not have an additional union-find. I proved the correctness and termination of `cc_explain2` and the equivalence of `cc_explain` and `cc_explain2`. The termination proof is based on an invariant, which still needs to be proven. See Subsection 1.2.1.

1.2 Termination of CC_Explain2

The proof of termination for `cc_explain` was based on the presence of the additional union-find, therefore it was necessary to find a different argument for the termination of `cc_explain2`. Based on an idea by Corbineau [Cor01], I added a timestamp for each edge in the proof forest, that shows in which order the edges were added to the proof forest [`CC_Definition2.thy`]. I extended the `congruence_closure` record with a list that contains the timestamps and the current timestamp [`time`]. Then I extended the congruence closure algorithm, so that it adds the corresponding timestamps to the edges and proved its equivalence to the original congruence closure algorithm [`merge_merge_t_equivalence`].

1.2.1 Invariants

I defined two invariants of the congruence closure algorithm, about the validity of the two new fields of the `congruence_closure_t` record. The `time_invar` defines that all timestamps in the proof forest are between 0 and the current timestamp (non-inclusive).

The invariant `timestamps_invar` still needs to be proven. [`CC_Explain2_Termination.thy`]. The only point in the congruence closure where the timestamps are modified, is in `propagate_step_t`. It needs to be shown that the `timestamps_invar` holds after `add_edge` was applied to the proof forest, `add_label` was applied to the labels list of the proof forest and `add_timestamp` was applied to the timestamps list [`timestamps_invar_step`]. From this, it follows that `timestamps_invar` is an invariant of `merge_t` [`timestamps_invar_merge_t`]. In theory, it should be true, because the edges on the path between two elements stay the same and only one edge is added with a timestamp that is larger than all the current timestamps in the proof forest. It is a bit complicated to prove, because the `add_label` function reverses the direction of some edges, therefore the `lowest_common_ancestor` of some elements might change. For more information about the `add_label` function, see [Ghi22].

1.2.2 Termination

Using these invariants, I proved that the multiset of the timestamps in the pending list decreases in each recursive call of `cc_explain2` [`recursive_calls_mset_less`]. Using induction on the multiset of the timestamps of the pending list, I proved the termination of `cc_explain2` [`cc_explain_aux2_domain`].

1.3 Correctness of Explain2

Given that `cc_explain2` does not have an additional union-find, it was possible to directly prove its correctness using the termination proof, the induction on `cc_explain2` and the invariant of the congruence closure algorithms defined for my bachelor's thesis [Ghi22]. [`cc_explain_aux2_correctness`]

1.4 Equivalence of Explain and Explain2

In order to prove the equivalence of `cc_explain` and `cc_explain2`, it needs to be shown that it is redundant to reconsider edges that have already been considered. To express that, I defined an invariant of `cc_explain` [`equations_of_l_in_pending_invar`].

I defined the `additional_uf_labels_set` of the additional union find, which is the set of labels of the edges that are present in the additional union-find. This set coincides

with the output of the `cc_explain` function. Additionally, the `additional_uf_pairs_set` is the set of pairs (a_1, b_1) and (a_2, b_2) for each edge in the additional union-find that is labeled with $F(a_1, a_2) = a$ and $F(b_1, b_2) = b$.

The invariant states that all pairs in the `additional_uf_pairs_set` are either in pending or have been in pending previously and have already been considered by the function, which means that the output of `explain_along_path2` is in the `additional_uf_labels_set` and the pending list is in the `additional_uf_pairs_set`.

With this invariant, it was possible to prove the equivalence of `cc_explain` and `cc_explain2`. I proved a generalized statement, using the induction rule on the timestamps of `xs2`:

```
lemma cc_explain_aux_cc_explain_aux2_equivalence:
assumes "set xs2 ⊆ set xs ∪ additional_uf_pairs_set"
      "subseq xs xs2"
shows "cc_explain_aux2 cc xs2 ⊆
cc_explain_aux cc l xs ∪ additional_uf_labels_set"
```

where `l` is the additional union-find. This way, we could use the induction hypothesis even though the pending list of `cc_explain2` may contain more elements than the one of `cc_explain`. We can show that the additional elements in `xs2` that are not in `xs` are redundant by using the invariant.

Bibliography

[Ghi22] R. Ghidini. *Formalisation of a Congruence Closure Algorithm in Isabelle/HOL*. BA thesis. 2022.

[Cor01] P. Corbineau. "Autour de la clôture de congruence avec Coq." MA thesis. Université Paris-Sud, 2001.