A Appendix

In this section, we present some lemmas solved by Thor only.

Case 1. The lemma *cols_upt_k _insert* is from the *QR Decomposition* entry[5] in the AFP.

```
lemma cols_upt_k_insert:
  fixes A::"'a^'n::{mod_type}^'m::{mod_type}"
  assumes k: "(Suc k)<ncols A"
  shows "cols_upt_k A (Suc k) = (insert (column
                  (from_nat (Suc k)) A) (cols_upt_k A k))"
  unfolding cols_upt_k_def
  apply (auto)
  apply (metis Suc_lessD from_nat_mono' from_nat_to_nat_id k
       less_Suc_eq_le less_le ncols_def to_nat_le)
  by (metis from_nat_mono' k less_imp_triv
       less_or_eq_imp_le ncols_def not_less_eq order_trans)
```

Here, *cols_upt_k A (Suc k)* returns the set of columns in the matrix *A* up to the natural number *k+1*, while *ncols A* counts the number of columns in the matrix *A*. In short, this lemma claims that the set of columns (in a matrix A) up to column index $k+1$ is equivalent to that of the same matrix up to column index k inserted with the $(k+1)^{\text{th}}$ column (of A). This will subject to the condition that $k+1$ is less than the number of columns in A. With Thor, the LM decided to unfold the goal with the definition of *cols_upt_k*, which is followed by an *auto* tactic to simplify the proof state. All remaining subgoals are then discharged by Sledgehammer.

Case 2. The lemma *size_del_max* is from the*Weight-Balanced Trees* entry[6] in the AFP.

```
lemma size_del_max: "t ≠ Leaf ⟹ size t = Suc(size(snd(del_max t)))"
  apply(induction t rule: del_max.induct)
  apply simp
  apply (clarsimp split: prod.splits)
  apply (smt (z3) size_rotateR size_wbt.simps(1))
  by simp
```

In this lemma, *t* is a weight-balanced tree, and the *size* function measures its size (as the name suggests) and *del_max* deletes the maximum node from it. Essentially, this lemma claims that when a weight-balanced its size will be reduced by one if we remove the largest node from it. For the proof, Thor intelligently performs structural induction with the induction rule *del_max.induct* and then simplifies the proof state a few times, which includes splitting products with the rule *prod.splits*. Finally, Thor concludes the remaining goals with Sledgehammer.

Case 3. The lemma *t_list_of_B_log_bound* is from the AFP entry named as *Priority Queues Based on Braun Trees*.[7]

```
lemma t_list_of_B_log_bound:
  "braun t ⟹ t_list_of_B t ≤ 3 * (nlog2 (size t + 1) + 1) * size t"
  apply (induction t rule: measure_induct_rule[where f=size])
  apply (case_tac x)
  apply simp
  using braun.simps(1) t_list_of_B_braun_simps(1) apply blast
  by (metis acomplete_if_braun height_acomplete order_refl
      size1_size t_list_of_B_induct)
```

Here, *size* measures the size of a Braun tree; *nlog2* stands for the function $\lambda x.\lceil \log_2(x) \rceil$; *t_list_of_B* is another measure of a Braun tree. Basically, this lemma describes the relationship between a normal tree size and a Braun-tree specific measure. The proof starts with an intelligent structural induction, progresses with case analysis, and is concluded with Sledgehammer on each of the remaining subgoals.

[5] QR_Decomposition/Gram_Schmidt.thy

[6] Weight_Balanced_Trees/Weight_Balanced_Trees.thy

[7] Priority_Queue_Braun/Sorting_Braun.thy

Case 4. The lemma *inj_imp_Ker0* is from the AFP entry named as *Matrices, Jordan Normal Forms, and Spectral Radius Theory*.[8]

```
lemma inj_imp_Ker0:
  assumes "inj_on T (carrier V)"
  shows "carrier (V.vs kerT) = {0_V}"
  apply (rule equalityI)
  apply (rule subsetI)
  apply (unfold ker_def, auto)
  by (metis V.module.M.zero_closed assms f0_is_0 inj_on_contraD)
```

Here, *T* is a linear map between two vector spaces. The lemma claims that if the *T* is injective on the carrier set of the space *V*, the kernel of *T* has to be a singleton set with the zero in *V*. In this proof, Thor naturally performs a sequence of introduction steps by applying the lemma *equalityI* and *subsetI*, before unfolds the definition of a kernel (i.e., *ker_def*) and uses *auto* to simplify the proof state. The final remaining goal is closed with Sledgehammer.

[8] Jordan_Normal_Form/Missing_VectorSpace.thy