Ergodic Theory of Groups: Week 11

Prof. Dr. C. Löh/J. Witzig June 24, 2020

Reading assignment (for the lecture on June 30)**.** We now introduce cost of groups and apply our results on cost of measured equivalence relations.

- Read Chapter 3.3.1 *Cost of groups.*
- Read Chapter 3.3.2 *Weak containment and ergodic decomposition.*

  This section is (unfortunately) a bit technical. However, the results are important for the following sections. Of course, instead of working through this section, you could solve the fixed price problem – then you don't have to worry about this technical section …

Reading assignment (for the lecture on July 1)**.**

- Read Chapter 3.3.3 *The fixed price problem.*
- Read Chapter 3.3.4 *The cost of the profinite completion.*

  In this context, it might be helpful to also compute the rank gradient of finitely generated free groups directly through the Nielsen-Schreier theorem.

  (Optional) Compute the rank gradients of fundamental groups of oriented closed connected surfaces via algebraic topology/group homology.

Next week, we will conclude Chapter 3 with some applications of ergodic theory (in the form of cost) to rank gradients. After that, we will start with a very different aspect of measured group theory, namely flexibility and amenability.

Implementation (graphings)**.** Read the third Isabelle fragment: *Graphings* (Appendix A.4) on graphings in a minimal setting without measurability etc.; you might also want to interact with it (it depends on RelRel_Iso.thy):

http://www.mathematik.uni-r.de/loeh/teaching/erg_ss2020/3006/Graphing.thy

We might discuss this fragment in one of the lectures.

Exercises (for the session on July 3)**.** The following exercises (which all are solvable with the material read/discussed in week 10) will be discussed.

Please turn over

Exercise 10.1 (rank gradients and subgroups)**.** Let Γ be a finitely generated infinite residually finite group. Which of the following statements are true? Justify your answer with a suitable proof or counterexample.

1. If $\Lambda \subset \Gamma$ is a finite index subgroup of Γ, then $\operatorname{rg} \Lambda = [\Gamma : \Lambda] \cdot \operatorname{rg} \Gamma$.
2. If $\Lambda \subset \Gamma$ is a non-trivial subgroup of Γ of infinite index, then $\operatorname{rg} \Gamma = 0$.

Exercise 10.2 (cost of subrelations)**.**

1. Let $(\mathcal{R}, \mu)$ be a measured finite equivalence relation on a standard Borel space X with $\mu(X) < \infty$ and let $\mathcal{R}' \subset \mathcal{R}$ be a standard equivalence relation. Show that
$$\operatorname{cost}_\mu \mathcal{R}' \leq \operatorname{cost}_\mu \mathcal{R}.$$
2. Does the same estimate also hold if we drop the finiteness assumption on $\mathcal{R}$? Justify your answer with a suitable proof or counterexample!

Exercise 10.3 (rank gradient of products)**.** Let Γ and Λ be finitely generated infinite residually finite groups. Complete the proof of Proposition 3.2.40 that
$$\operatorname{rg} \Gamma \times \Lambda = 0.$$

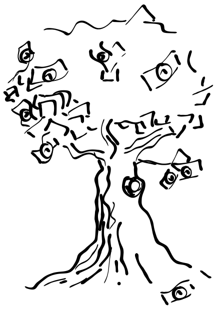
Exercise 10.4 (implementation: ranks of products)**.** We consider the file
http://www.mathematik.uni-r.de/loeh/teaching/erg_ss2020/2406/Rank_Exercise.thy

This depends on Product_GenSet.thy (which you should thus keep in the same directory). Complete this file, i.e.:

1. Give pen-and-paper completions of the proofs of the statements *optimal_gen_set_ex*, *gen_sets_product_size*, *rank_product_subadditivity*.
2. Implementation: Complete the proof of *optimal_gen_set_ex*.
3. Implementation: Complete the proof of *gen_sets_product_size*.
4. Implementation: Complete the proof of *rank_product_subadditivity*.

Bonus problem (exact cost)**.** Let $c \in \mathbb{R}_{\geq 0}$. Show that there exists a measured equivalence relation $(\mathcal{R}, \mu)$ on a standard Borel space X such that μ is a probability measure on X without atoms and that satisfies
$$\operatorname{cost}_\mu \mathcal{R} = c.$$

Hints. Dissect a suitable free product and don't lose 1.9 billion on the way!



Submission before July 1, 8:00, via email to johannes.witzig@ur.de or through git.