# Automated generation of planar geometry olympiad problems

*(Bc. Patrik Bak)*

https://github.com/PatrikBak/GeoGen

**Supervisor:** doc. RNDr. Stanislav Krajči, PhD.
**Consultant:** Mgr. Michal Rolínek, PhD.

## Overall goal

Design and implement software that is able to generate planar geometry olympiad problems by extending an initial configuration with new geometrical objects and subsequently finding non-trivial theorems in the generated configurations.

## What was done in the bachelor thesis

1. We designed and implemented an abstract object representation of geometrical objects, configurations and theorems. Every algorithm then built on this representation. This representation should stay the same for the master thesis as well.
2. We designed and implemented algorithms that generate geometrical objects and configurations. These algorithms were independent from an actual geometrical representation of these objects. Because of the modular nature of the generation process there is a high chance that these algorithms will be used in a different context for the new algorithms of the master thesis.
3. We designed and implemented algorithms that find all possible theorems relevant to a configuration. We looked for 9 types of theorems (such as 'Concurrent objects', 'Collinear points', etc.). Each of these types had its own optimized algorithm that was able to generate all potential theorems that held true in the configuration. These theorems were then verified (see next point) and filtrated in order to exclude the irrelevant ones.
4. We designed and implemented a simple numerical verification system of theorems. This system used classical analytic geometry. A configuration was drawn randomly several times and the potential theorems were then verified numerically. Even if we could not be 100% sure that we had found real theorems, the chances were practically almost 100%.

## What is left for the master thesis

1. The main thing we have to do is to design and implement an algorithm for the detection of trivial theorems. Since the goal is to generate *olympiad* problems, we need to get rid of the ones that are either trivial by look, or turn out to be just a consequence of some higher known (though maybe not trivial) theorems.
2. Another important way to approach the olympiad level of problems is to make sure our theorems cannot be simply generalized further. For example if a point is the midpoint of a line segment, then the theorem should not hold true if it were an arbitrary point of this segment (olympiad level geometries avoid this style of misleading).
3. In order to improve the generated results even further, we might consider adding algorithms that can simplify a configuration so that it does not contain unnecessary objects. This was partially done in the bachelor thesis — there was an algorithm

that found out if a theorem 'used' every object of the configuration. The problem is that sometimes we can not exclude an object, because it is used in the definition of another object, but we could replace both of these objects with a single one (for example, the line perpendicular to a line segment at its midpoint is essentially its perpendicular bisector).

4. We also want to consider different approaches to generating problems. Right now the generation algorithms generate every possible configuration (to some given depth). A lot of them do not look like olympiad problems (even with relevant theorems), for example, when they repeat the same construction more than twice.
5. From an implementational point of view, we need to implement a database that would contain all the generated configurations, theorems and potential relations between them found by the new algorithms. The database is an inseparable aspect of the further work, since the algorithms will use it to gradually extend its knowledge base.
6. We need to think through a way to present the generated results. One option could be generating a code of the figure to be processed by an external program, such as Metapost. It is also a good idea to implement a simple user interface that will be used to run system tests and for tasks like manual marking of trivial theorems.

## The following steps

1. Prepare the current code for new changes. This includes a regular code clean-up, modularity improvement at a few places, fixes of some minor issues that were not important in the bachelor thesis, writing more tests, and adding more diagnostic options.
2. Figure out all possible algorithms for detecting trivial theorems. Implement one as an in-memory algorithm.
3. Figure out all possible algorithms for theorem generalization. Pick one and implement it as an in-memory algorithm.
4. Design and implement the database according to the picked algorithms. Integrate the database with the algorithms.
5. Run complex system tests. Analyze the output and according to its quality decide on further steps.

## References


[1] Grozdev, S., & Dekov, D. (2015). A Survey of Mathematics Discovered by Computers. *International Journal of Computer Discovered Mathematics*, 3-20. https://www.researchgate.net/publication/306426026_A_Survey_of_Mathematics_Discovered_by_Computers

[2] Alvin, C., Gulwani, S., Majumdar, R., & Mukhopadhyay, S. (2014, July). Synthesis of Geometry Proof Problems. In *AAAI* (pp. 245-252). http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/viewFile/8617/8427

[3] Singhal, R., Henz, M., & McGee, K. (2014, April). Automated Generation of Geometry Questions for High School Mathematics. In *CSEDU* (2) (pp. 14-25). http://www.cs.technion.ac.il/~janos/COURSES/THPR-2015/SinghalHenzMcGee.pdf

[4] Poulos, A. (2017). A Research on the Creation of Problems for Mathematical Competitions. *Teaching of Mathematics*, 20(1). http://elib.mi.sanu.ac.rs/files/journals/tm/38/tmn38p26-36.pdf