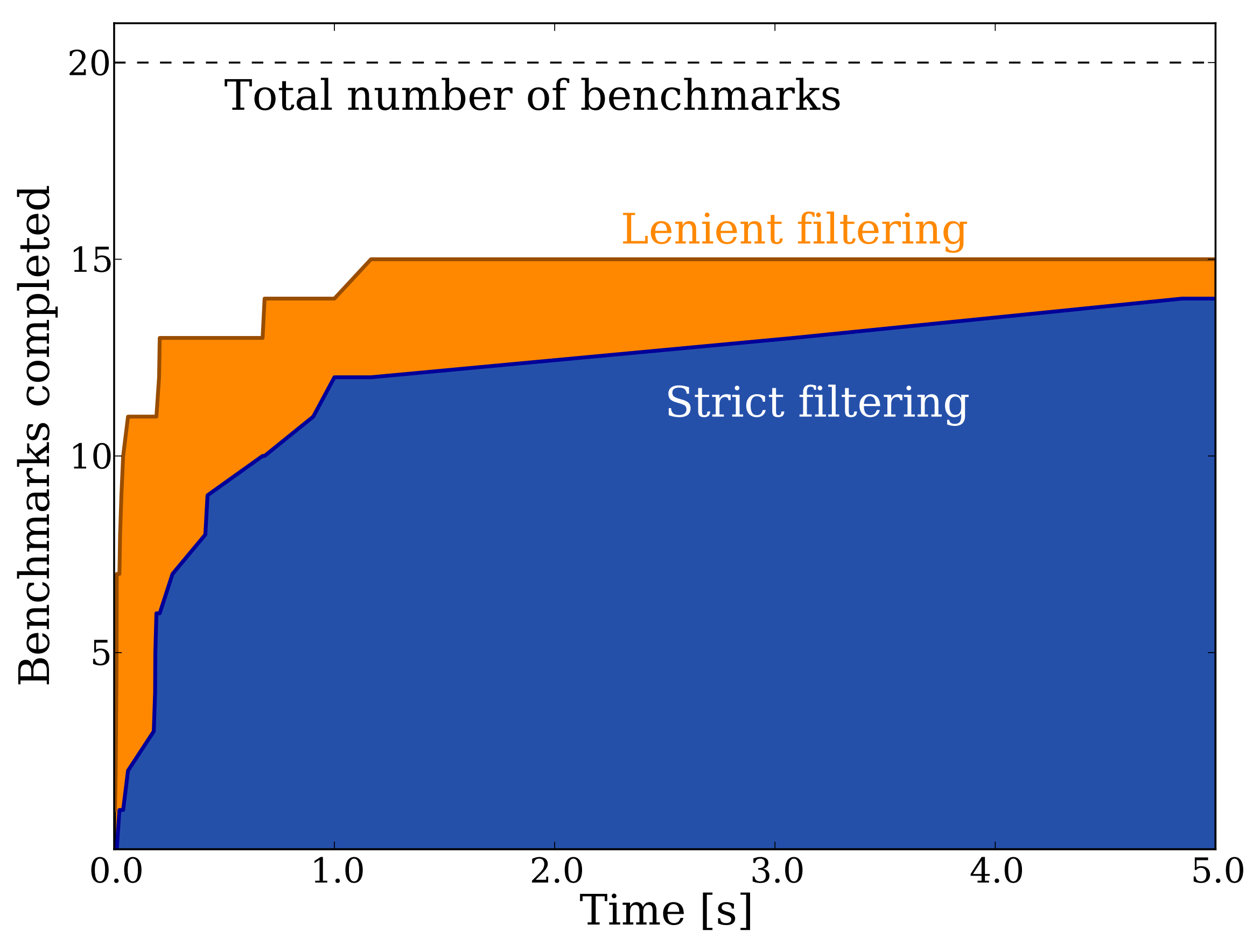
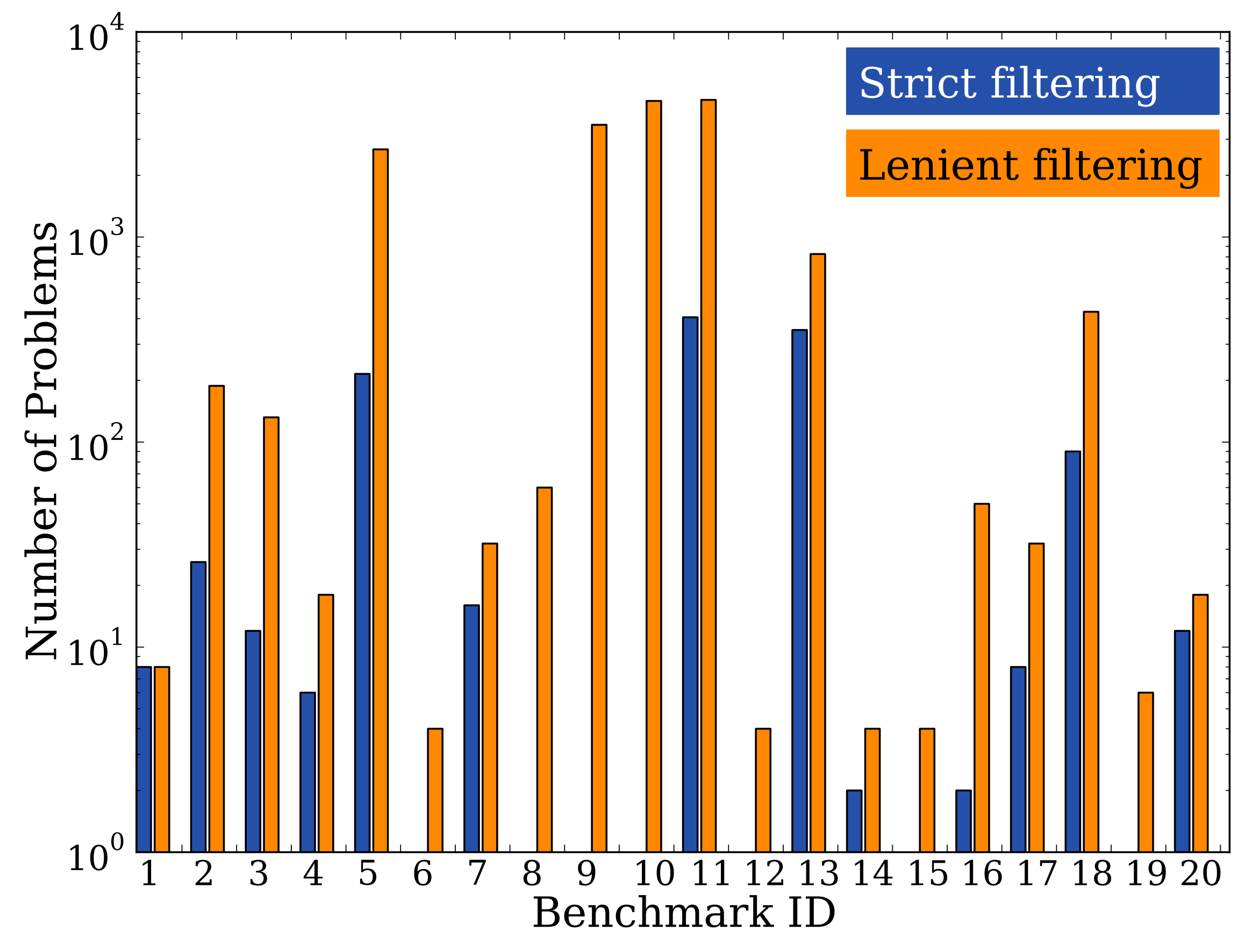
Problem Generation for DFA Construction

Alexander Weinert
alexander.weinert@rwth-aachen.de

Problem

Deterministic Finite Automata (DFA) are an important concept taught to nearly every CS student. Most courses support the material taught with a set of tasks of the form "*Construct a DFA that recognizes the language L*". However, there is usually at most one exercise for each concept that can be used when solving such a problem. We solve this problem by generating a set of problems of this form that are of both **similar form** and **similar difficulty** to a given problem.





Results

Runs on testcases taken from [1] terminate in around **1 to 2 seconds** and usually generate around **50 to 200 problems**. We inspected the resulting problems and found them to be of **similar difficulty to the original ones**. The number and quality of the resulting problems as well as the time for generating them depends heavily on the complexity and formulation of the original problem.

startsWith 'a' and |indOf 'ab'| % 2=1

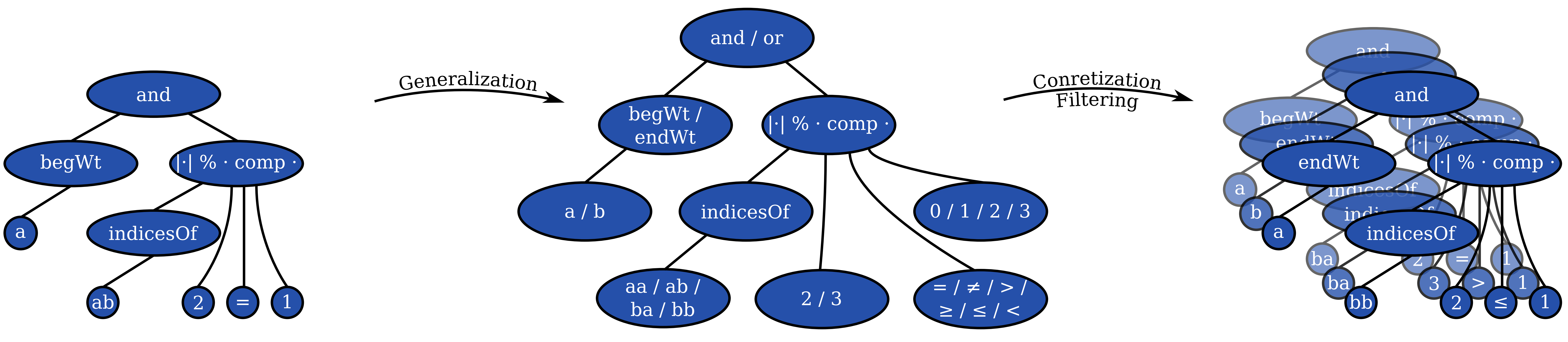
---

startsWith 'a' and |indOf 'ba'| % 2=1
endsWith 'b' or |indOf 'ab'| % 2=1
endsWith 'b' or |indOf 'aa'| % 2=1
endsWith 'a' and |indOf 'ba'| % 2<1
startsWith 'b' or |indOf 'bb'| % 2=1
startsWith 'b' and |indOf 'bb'| % 2=1
endsWith 'b' and |indOf 'aa'| % 2=1
endsWith 'a' or |indOf 'ba'| % 2<1
startsWith 'a' or |indOf 'ab'| % 2=1
startsWith 'a' or |indOf 'ba'| % 1>0

Examples of original formula and generated formulas

Idea

Our approach is based on the idea of Singh et al. for creating algebra problems from [2]. We first translate the original problem into a logical formula. Following this, we **generalize** this formula to a choice-formula that represents multiple formulas of the same structure. We then **concretize** this generalization using a SMT-solver. As a final step, we **filter** the problems that are too difficult or too easy.



Original Problem **Choice-formula** **Set of generated Problems**

Method

Our algorithm takes a Mosel-formula as input. This formula describes the regular language for which an automaton is to be constructed [3].
We then transform the formula's AST into the AST of the corresponding Choice-formula, which describes a set of Mosel-formulas.
This choice formula is then translated into SMT-constraints and handed off to a solver. Each solution of these constraints is equivalent to a new Mosel-formula. The translation to SMT-constraints allows us to control the instantiation more precisely. This allows us to keep the generated formulas similar to the original one.
Since Mosel-formulas correspond to the regular languages, we can then construct the DFA corresponding to each generated formula. By comparing the two DFAs of the generated and the original formula, we can then filter those formulas that result in a DFA that is too different from the original one.

Acknowledgements

The author would like to thank Prof. Sanjit Seshia, the students of CS298-98 and Loris D'Antoni for continuous feedback on this project as well as the authors of automatatutor.com, whose code was used as the basis for the implementation.

References

[1] J. Hopcroft et al., Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 2011
[2] R. Singh et al., Automatically Generating Algebra Problems, AAAI-12
[3] R. Alur et al., Automated Grading of DFA Constructions, IJCAI-13