Bachelor Thesis

Visualizing Dynamic Programming on Tree Decompositions

Martin Röbke

born: 04.03.1995 in Dresden, Germany

matriculation number: 3949819

martin.roebke@tu-dresden.de

Technische Universität Dresden
Faculty of Computer Science
International Center For Computational Logic

Supervisor: Dr. Johannes Fichte

Second evaluator: Prof. Dr. rer. nat. Stefan Gumhold

Dresden, June 7, 2020

Abstract

The thesis is about a practical and lightweight implementation for visualizing dynamic programming on tree decompositions. I created the python-packagetdvisu for the purpose of visualizing, teaching and analyzing the solving process of MSOL-problems using tree decomposition. Intended audience: Developer of dynamic programming on tree decompositions for debugging. Researcher of such algorithms for comparisons and visualizations. Teachers or students wanting some automatic visualization of their examples and the dynamic-solving-process. As two reference implementations of dynamic programming on tree decompositions the projects GPUSAT and dpdb were chosen.

Contents

1. Introduction

Graphs are increasingly interesting in scientific work. The idea for this project comes from my supervisor Dr. Johannes Fichte, who works on and with many projects such as the ones listed above on solving monadic second order logic (MSOL [1]) problems using highly parallelized architectures like graphics processing units or state of the art databases. One early implementation is published in [2] where for different real world examples the results looked promising These projects are very competitive ????REF????? for solving even large instances of those problems.

intro. mit motivation und related work, state of the art, advancements.

Visualization Pipeline

Stand Umsetzung, Tools: Slack, Trello, GitHub, Presentations

2. Background

2.1. Boolean satisfiability problem

https://en.wikipedia.org/wiki/Boolean_satisfiability_problem

SAT was the first known NP-complete problem, as proved by Stephen Cook at the University of Toronto in 1971 [3]

literal ≡ boolean variable v or its negation
clause ≡ finite set of literals, interpreted as the disjunction
unit ≡ clause with |c|=1
CNF formula ≡ set of clauses, interpreted as their conjunction
var ≡ set of variables contained in the clause or clause set C
assignment ≡ α:var(C) $\rightarrow$ {0,1}
satisfiedclause ≡ if $\exists$v $\in$ var(c), v$\in$c and α(v)=1 or $\neg$v$\in$c and α(v)= 0. Otherwise falsified
satisfiedform ≡ each clause in the formula is satisfied by assignment

2.2. MSOL

See also figure 2.

2.3. DIMACS Format

2.4. Tree Decomposition

2.5. Courcelle's Theorem

> Every graph property definable in monadic second-order logic (MSO) is decidable in linear time on graphs of bounded treewidth.
> Courcelle, Bruno (1990)[1]

For all $k \in \mathbb{N}$ and MSO-sentences F is the decision problem for a given graph G, whether $G \models F$ is true, in time $2^{p(tw(G))} \cdot |G|$ with a polynom p decidable.

- *drawback:* still expensive ($2^{p(twG)}$, $2^{2^{(\#Q)}}$, large constants)

The workflow then looks like we see in figure 1.

[1] Courcelle, Bruno "The monadic second-order logic of graphs. I. Recognizable sets of finite graphs", Information and Computation, 85 (1990) no. 1: 12-75

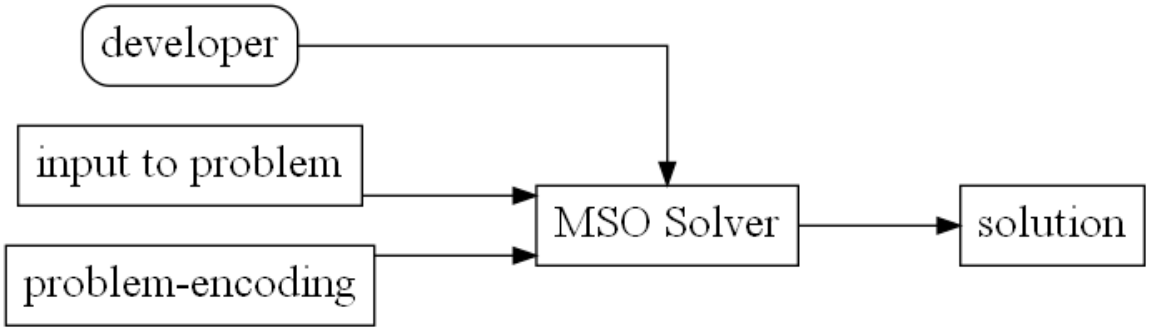

Figure 1: Implementation of the theorem

3. Concept

What I do and why I did it Steps in Trello, Issues, Commits Research: language (python - explain) graph-construction (graphviz vs networkX), examples (diploma at first).

4. My Visualization Project

Github Objectives htd hier oder auslassen? Files / Classes / Methods Current perspective Checking with www.deepcode.ai

4.1. Integration in GPUSAT

Programm Umsetzung Beispiel

4.2. Integration in dpdb

Programm Umsetzung Beispiel

5. Application and Images

6. Summary and Outline

What is achieved? What worked good, what bad?

References

[1] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*. Cambridge: Cambridge University Press, 1 ed., 2012.

[2] A. Langer, F. Reidl, P. Rossmanith, and S. Sikdar, "Evaluation of an mso-solver," *Proc. of ALENEX 2012*, 01 2012.

[3] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, (New York, NY, USA), p. 151–158, Association for Computing Machinery, 1971.

A. Images

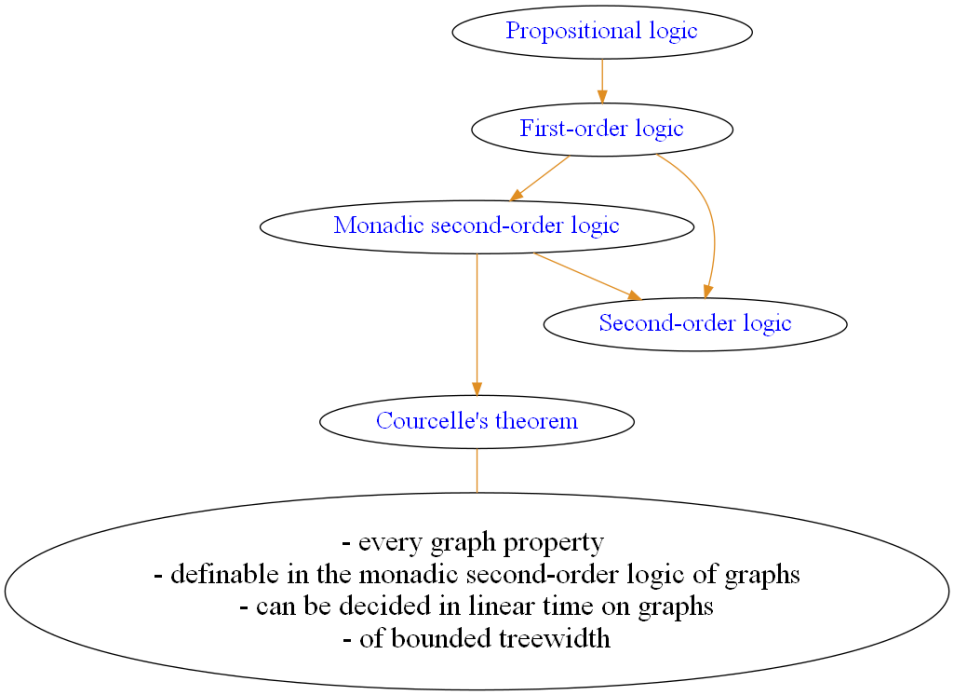

Figure 2: From propositional logic to monadic second order logic