

AITP Methods for Maximizing Generality and Tractability of Algebras

Gonçalo Araújo
Universidade Nova de Lisboa
Lisbon, Portugal

Abstract

We value a mathematical theory for its generality and its tractability. For example, a magma (that is, a set with a binary operation) is very general because any theorem on it will have an impact on almost all algebras and possibly beyond, but the structure is so weak that deep theorems are very difficult to prove. The goal of my current work is to use Artificial Intelligence Theorem Proving (AITP) tools and AI techniques to devise new mathematical theories, maximizing generality and tractability. We expect to find several theories that will have a good ratio of generality/tractability that might be explored in the future. In addition, we will have many theories with a high level of generality where tractability might be proved once sufficiently strong AITP tools are available.

1 Introduction

Usually, a paper proposing a new class of mathematical objects starts by listing the areas that are particular cases of the new theory. The goal of my work is to highlight the generality of the theory.

Typically, an important new mathematical theory will have as particular examples a couple of important areas of mathematics. Regarding tractability, it is incumbent upon the paper itself to prove it by providing deep results and interesting connections to other parts of mathematics. The point to underline is that generality and tractability are mutually opposed parameters: an increase in generality usually implies a decrease in tractability; strong tractability usually means poor generality.

This research aims to establish a balance between theoretical generality and practical applicability. The resulting theories will find applications in diverse fields such as computer science, physics, engineering, and economics, enhancing our ability to model and solve real world problems.

2 Impact and Trustworthiness

Theorem provers such as Prover9, can be used to formally verify a range of mathematical properties, including properties of AI systems expressed as logic formulas. This process helps ensure that AI systems behave as intended under various conditions, which is one of the strongest methods for establishing their trustworthiness. Algebras of various generality are ubiquitous in formal verification.

3 Some Results with AITP Tools

It is well known that in every Inverse Semigroup, given by the axioms:

1. $x(yz) = (xy)z$; $f(f(x)) = x$; $xf(x)x = x$;
2. $xf(x)yf(y) = yf(y)xf(x)$; $f(xy) = f(y)f(x)$,

all its idempotent elements commute with each other. So one could think about generalizing those axioms and still have that same result. However, it is important to observe that, when generalizing a class of algebras, we have to check if the new definition is not equivalent to the original one.

By weakening those axioms, I generalized the concept of Inverse Semigroups, called Generalized Inverse Semigroups (GIS), given by the following axioms:

1. $x(yz) = (xy)z$; $f(f(f(f(x)))) = x$; $xf(x)xy = xy$;
2. $xf(x)yf(y) = yf(y)xf(x)$.

It is clear that $IS \subseteq GIS$. Some of the properties that I was able to prove, with the help of Prover9 were, for example, that all idempotent elements in GIS commute, $xef(x) \in E(GIS)$ for all $x \in GIS$ and $e \in E(GIS)$. In order to get those proofs, one must put the axioms of GIS in the assumptions and then the propriety we would like to try to prove in the goals.

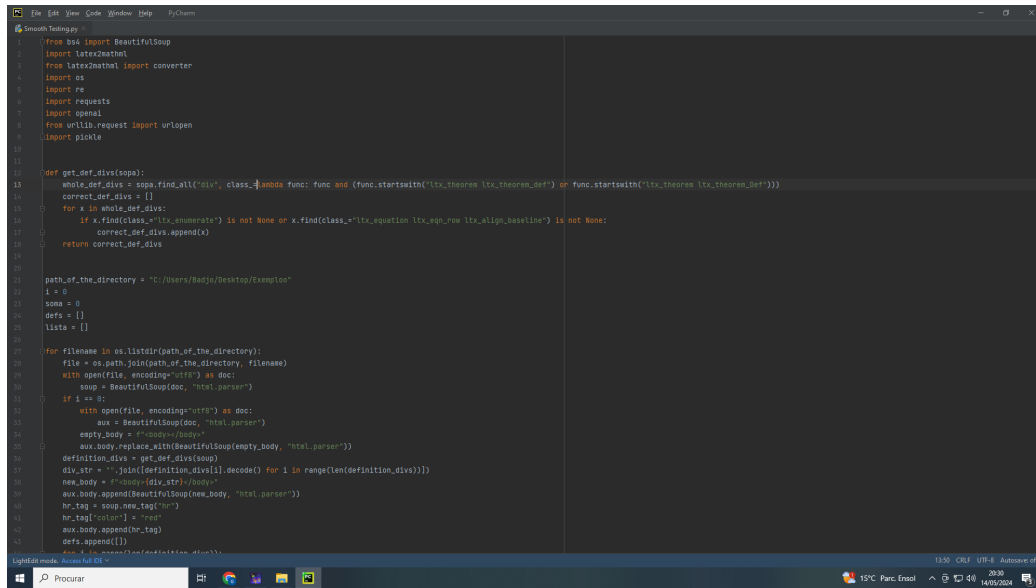
References

- [1] Max Tegmark, Steve Omohundro: Provably safe systems: The only path to controllable AGI. CoRR abs/2309.01933 (2023).
- [2] M. L. Dalla Chiara and R. Giuntini, Quantum Logic, <https://arxiv.org/abs/quant-ph/0101028>, (2001).
- [3] Keqin Liu, A class of group-like object, <https://arxiv.org/abs/math/0311396>, (2003).
- [4] M. Pedicini and F. Quaglia, PELCR: Parallel Environment for Optimal Lambda-Calculus Reduction, <https://arxiv.org/abs/cs/0407055>, (2004).
- [5] Tim Boykett, Orderly Algorithm to enumerate central groupoids and their graphs, <https://arxiv.org/abs/math/0407070>, (2004).
- [6] Nick Papanikolaou, Logic Column 13: Reasoning Formally about Quantum Systems: An Overview, <https://arxiv.org/abs/cs/0508005>, (2005).
- [7] V. A. Shcherbacov, On the structure of left and right F-, SM- and E-quasigroups, <https://arxiv.org/abs/0811.1725>, (2008).
- [8] Jack Spielberg, Groupoids and C*-algebras for categories of paths, <https://arxiv.org/abs/1111.6924>, (2014).
- [9] Cornejo, J.M., Sankappanavar, H.P. Order in Implication Zroupoids. *Stud Logica* 104, 417–453 (2016). <https://doi.org/10.1007/s11225-015-9646-8>
- [10] Borowiec, Andrzej and Dudek, Wiesław A. and Duplij, Steven, Bi-Element Representations of Ternary Groups, <https://arxiv.org/abs/math/0306210>, (2006).
- [11] Andrzej Borowiec and Wieslaw A. Dudek and Steven Duplij, Basic concepts of ternary Hopf algebras, <https://arxiv.org/abs/math/0306208>, (2003).
- [12] Tim Boykett, Rectangularity, <https://arxiv.org/abs/1108.1658>, (2011).

- [13] Dudek, Wieslaw A. and Thomys, Janus, On some generalizations of BCC-algebras, <https://arxiv.org/abs/1205.1440>, (2012).
- [14] Ulrich Kraehmer and Friedrich Wagemann, Racks, Leibniz algebras and Yetter-Drinfel'd modules, <https://arxiv.org/abs/1403.4148>, (2014).
- [15] Dejan Delic and Aklilu Habte, A new algorithm for constraint satisfaction problems with Maltsev templates, <https://arxiv.org/abs/1709.08311>, (2017).
- [16] McCune, W. Prover9 Automated Theorem Prover. <http://www.cs.unm.edu/~mccune/prover9/>

Appendices

A Some of the Technology Developed so Far



```
1 from bs4 import BeautifulSoup
2 import latex2mathml
3 from latex2mathml import converter
4 import os
5 import re
6 import requests
7 import openal
8 from urllib.request import urlopen
9 import pickle
10
11
12 def get_def_divs(soup):
13     whole_def_divs = soup.find_all("div", class_=lambda func: func and (func.startswith("ltx_theorem ltx_theorem_def") or func.startswith("ltx_theorem ltx_theorem_def")))
14     correct_def_divs = []
15     for x in whole_def_divs:
16         if x.find(class_="ltx_enumerate") is not None or x.find(class_="ltx_equation ltx_eqn_row ltx_align_baseline") is not None:
17             correct_def_divs.append(x)
18     return correct_def_divs
19
20 path_of_the_directory = "C:/Users/Bazjo/Desktop/Exemplo"
21 i = 0
22 soma = 0
23 defs = []
24 lista = []
25
26 for filename in os.listdir(path_of_the_directory):
27     file = os.path.join(path_of_the_directory, filename)
28     with open(file, encoding="utf-8") as doc:
29         soup = BeautifulSoup(doc, "html.parser")
30         if i == 0:
31             with open(file, encoding="utf-8") as doc:
32                 aux = BeautifulSoup(doc, "html.parser")
33                 empty_body = "<body></body>"
34                 aux.body.replace_with(BeautifulSoup(empty_body, "html.parser"))
35             definition_divs = get_def_divs(soup)
36             div_str = "<".join([definition_divs[i].decode() for i in range(len(definition_divs))])
37             new_body = f"<body>{div_str}</body>"
38             aux.body.append(BeautifulSoup(new_body, "html.parser"))
39             hr_tag = soup.new_tag("hr")
40             hr_tag["style"] = "width: 100%; border: 0.5px solid black;"
41             aux.body.append(hr_tag)
42             defs.append(i)
43             i += 1
44             soma += 1
45             lista.append(filename)
```

Figure 1: Illustration of part of a program that would extract the mathematical definitions from a html document, in order to gather varieties and quasi-varieties.

```
output: (axiom_46_1) \n

input: [defs_dicts[47]["Axiom1"]] \n (defs_dicts[47]["Axiom2"]) \n
output: (axiom_47_1) \n (axiom_47_2) \n

"""
# print(prompt)
# openai.api_key = "sk-07ay5d88aGtp8e3bFFAT3B1bkF39pBuld8mgdyKFdyA88g"
response=openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
    messages=[
        { "role": "user", "content": prompt}
    ],
    temperature=0
)
response1 = response.choices[0].message.content
with open(f'C:/Users/Utilizador/Desktop/prover9_dicts/axioms{i}.pickle', 'wb') as file:
    pickle.dump(response1, file)

for i in range(0, 0):
    print(i)
    n = len(defs_dicts[i])
    with open(f'C:/Users/Utilizador/Desktop/prover9_dicts/axioms{i}.pickle', 'rb') as file:
        aux = pickle.load(file)
        for j in range(1, n-1):
            print(defs_dicts[i][f"Axiom{j}"])
            print("\n \n \n")
            print(aux)
            print("\n \n \n")
            print("NEXT DEFINITION")
```

Figure 2: Illustration of part of a program that, using ChatGPT, would be able to convert the axioms of those classes of algebras into first-order logic formulas so we could introduce those axioms in Prover9.