Theorem Prover

ATLS 5241: Gregory Greenstreet, Brian Newsom
Sushma Akoju

Project Details

Project name: Theorem Prover

All contributing team members full names and emails: Sushma Akoju, sushma.akoju@colorado.edu

A link to or PDF of your presentation slides:

https://docs.google.com/presentation/d/1G3IEAArSBkxVCxcbiNA-CP7ZTPym5P4DYHeqA_or-DA/edit?usp=sharing

A link to your project: https://theorem-prover-4182022.uc.r.appspot.com/

Video: https://drive.google.com/drive/folders/11CYQdIHaEHFdWmHj36qu4WJO3829uT_0?usp=sharing

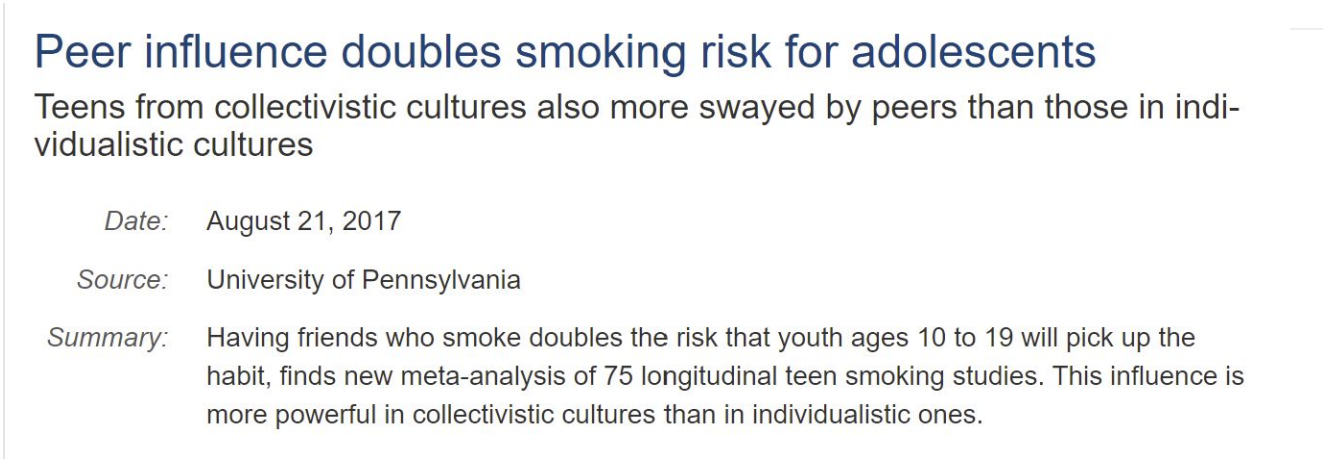
Repository: https://github.com/sushmaakoju/demo-ATLS5214

About the conceptual part of the Project

- Theorem Provers have been there all the time: Compilers and Interpreters.
- It was Theorem Proving concept that inspired Compilers.
- We extend Theorem Proving concept to solve real world problems Natural Language Problems.

Example description for this project

- Smoking causes cancer
- We need to stop people from smoking
- It's hard to do that since people are influenced by friends
- If friends keep smoking, they are likely to continue smoking



Peer influence doubles smoking risk for adolescents

Teens from collectivistic cultures also more swayed by peers than those in individualistic cultures

Date: August 21, 2017

Source: University of Pennsylvania

Summary: Having friends who smoke doubles the risk that youth ages 10 to 19 will pick up the habit, finds new meta-analysis of 75 longitudinal teen smoking studies. This influence is more powerful in collectivistic cultures than in individualistic ones.

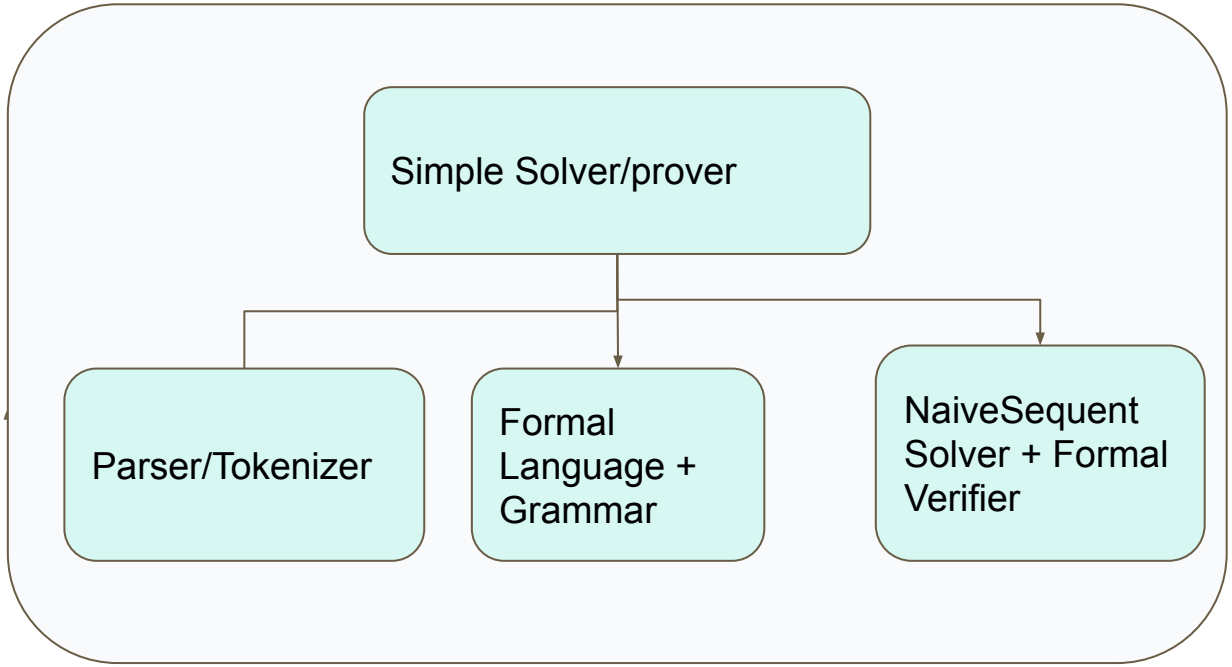
https://www.sciencedaily.com/releases/2017/08/170821102718.htm

Natural Language Beliefs and Logic Statements

- Let us say, we have following beliefs from previous slide:
  - Smoking causes cancer
  - Friends have similar habits
  - Let Alice and Bob be two friends
  - Alice Smokes and has Cancer
  - Alice has Smoking habit
  - Bob has Smoking habit
  - Can Bob get Cancer?

Simple Theorem Prover (Entscheidungsproblem)

- General Substitution
- General Unification
- **Entscheidungsproblem** -
- Universally validity and defines
- Using Hilbert's version



Reference: https://www2.karlin.mff.cuni.cz/~stovicek/math/decidability.pdf

About Universal Validity

Hilbert: Is there an algorithm which, given an effectively described theory, such as Peano Arithmetics, and a sentence ξ in the theory decides, whether ξ is or is not provable from the axioms?

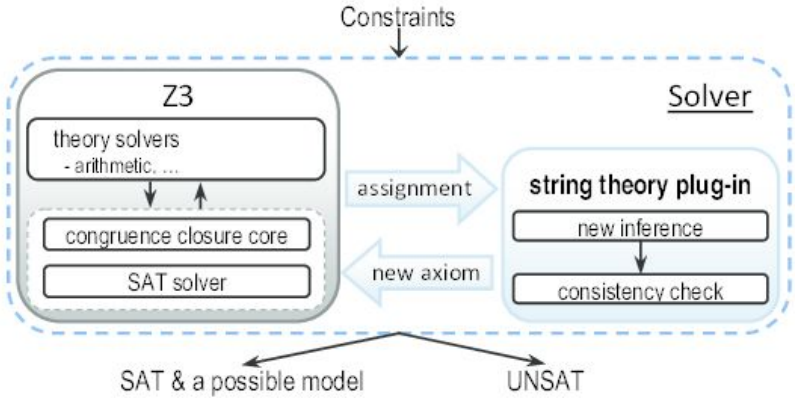
Halting problem: Is there an algorithm (program) Halt(P, F) which, given a source code P of another program and its input file F, decides whether P halts on the input F ?

Turing: There is no such algorithm. Therefore, the halting problem is undecidable.

-> There is no such Universally valid algorithm

Z3 Solver and Prover (Microsoft Research)

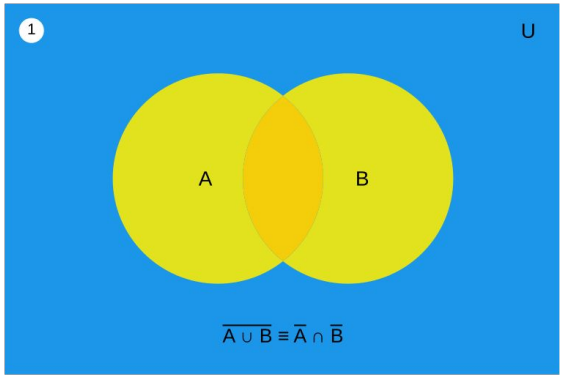
- Built from advanced concepts of Satisfiability Theories (Modulo)
- Proof by Refutation
- Z3 first tries to prove theorem is wrong from set of axioms
- If it fails to prove theorem is wrong from set of facts, then theorem is true.

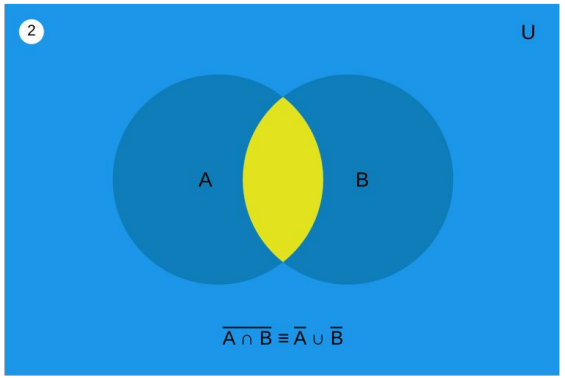


De Morgan's Law

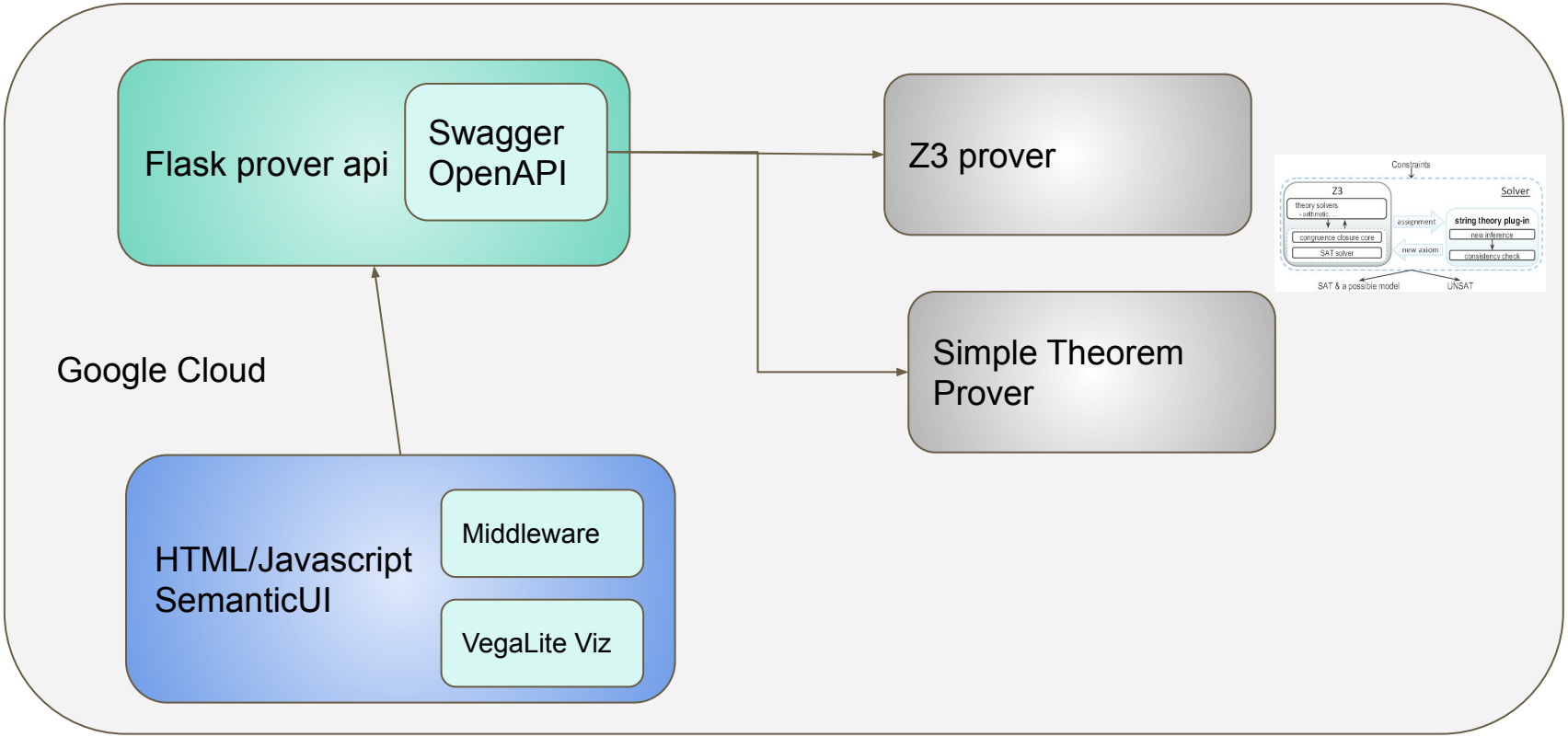
The negation of a disjunction is the conjunction of the negations

The negation of a conjunction is the disjunction of the negations





Architecture



Challenges & Solutions

- Included complex methods for api methods
- Used Textbox instead of dropdown - they are not universal provers!
- Vega is not converted to HTML using D3
- Dataset NOT received as it is pending approval from researcher

Solutions:

- Use simple abstract api methods
- Use dropdown
- Use VegaEmbed to simply populate Vega visualizations directly into HTML

Google Cloud deployment & documentation

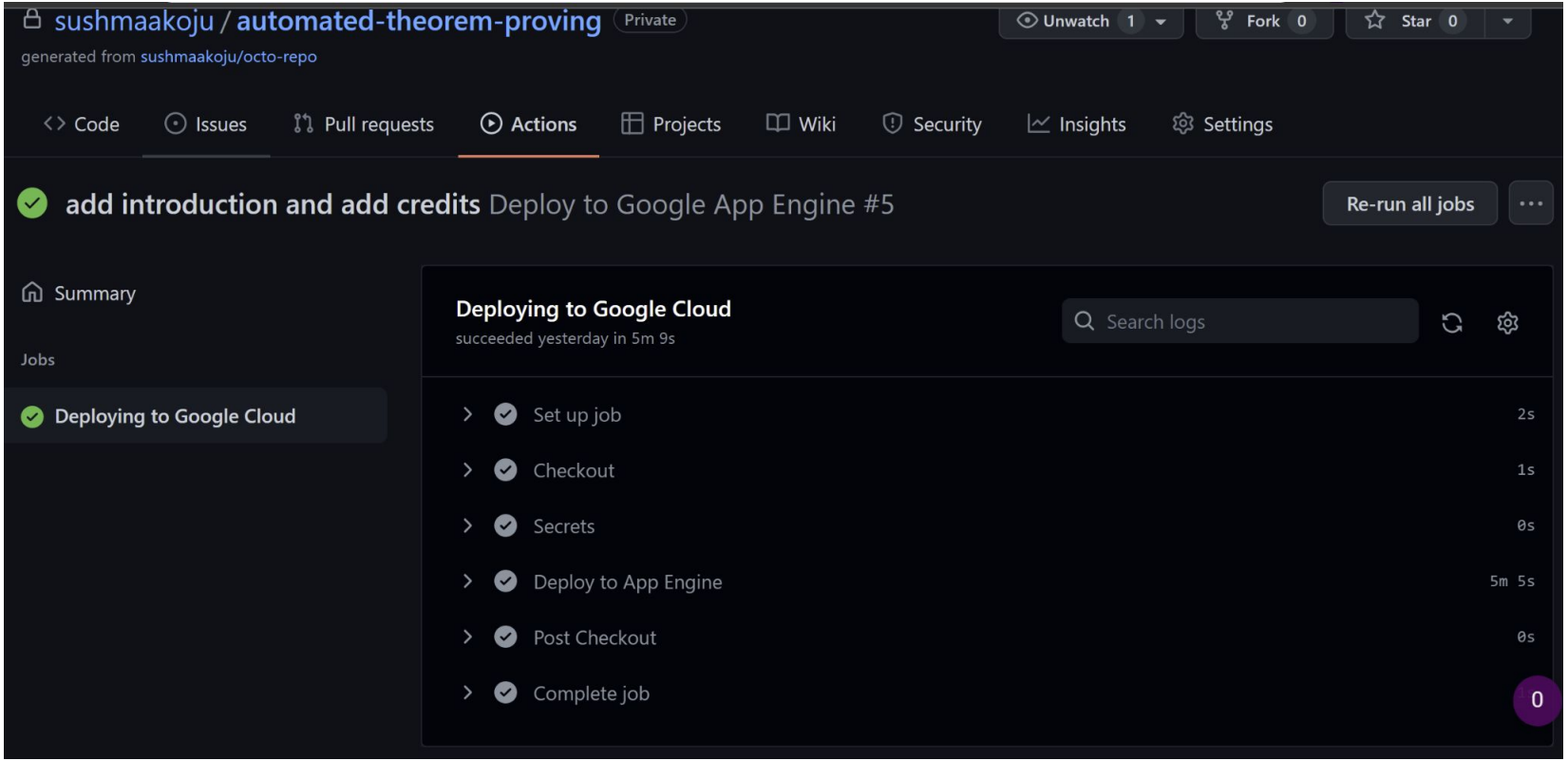
Deployed with help and guidance from Sagar - due to similarity of architecture.

I wrote this document to write down details of what I learnt from Google Cloud:

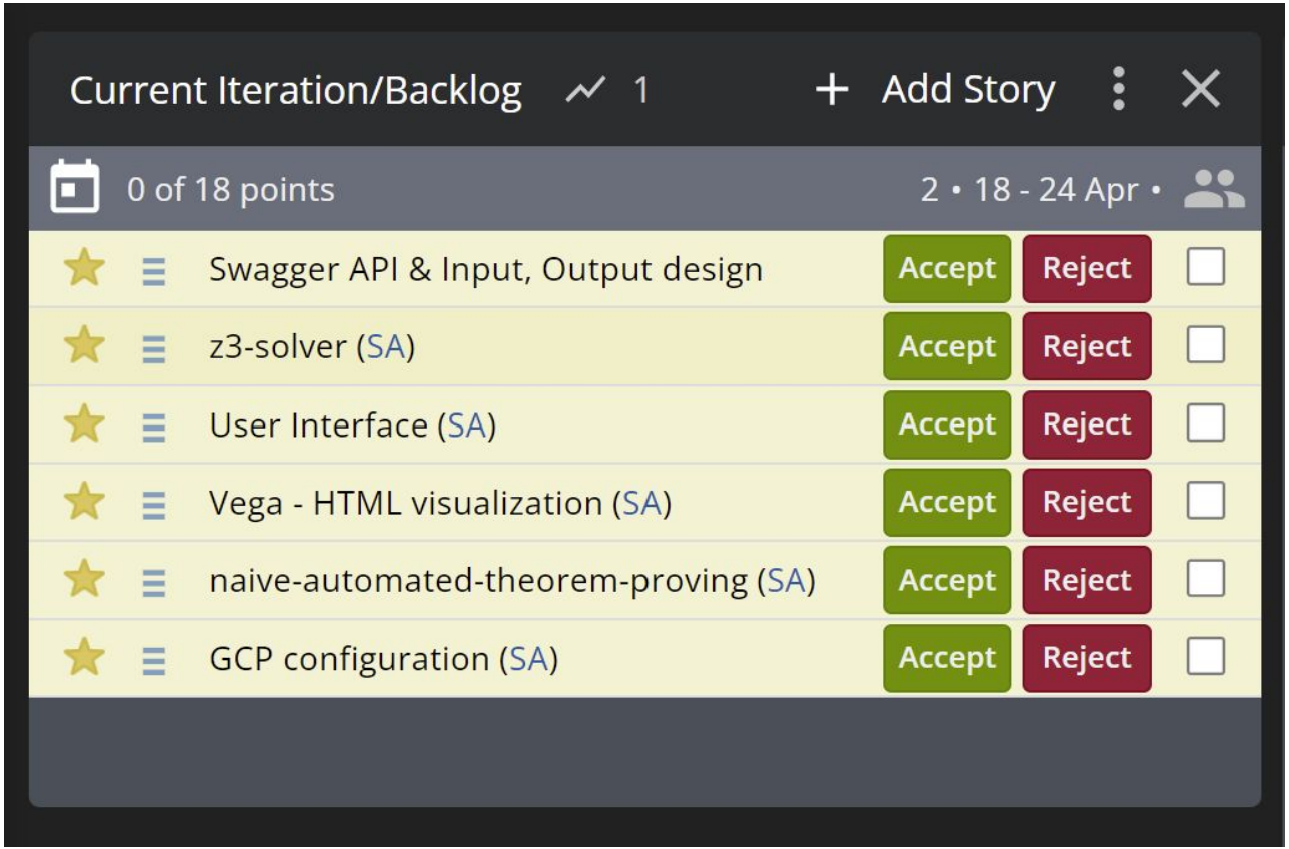
https://docs.google.com/document/d/1WvGXkunjrA8_EkvV5E-KLgv-t5DTDvTVogJZCgk5mSA/edit?usp=sharing

Reviewed by Sagar.

CI-CD

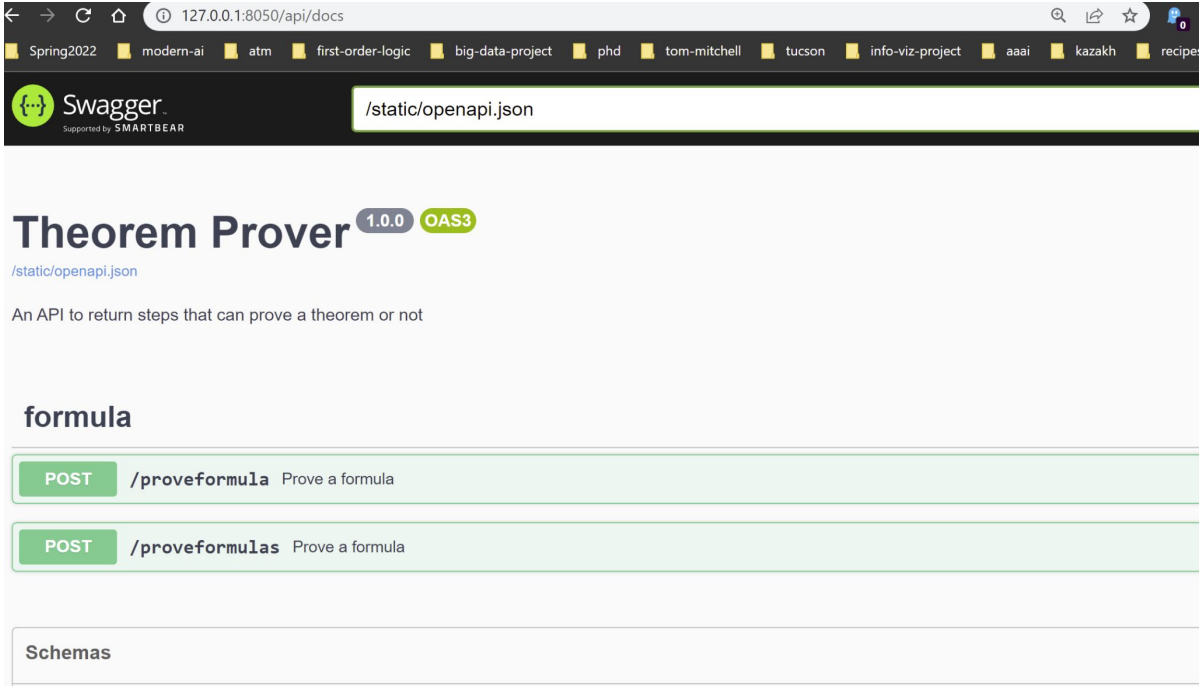

sushmaakoju / automated-theorem-proving
Private
generated from sushmaakoju/octo-repo
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
add introduction and add credits Deploy to Google App Engine #5
Re-run all jobs
Summary
Jobs
Deploying to Google Cloud
Deploying to Google Cloud
succeeded yesterday in 5m 9s
Search logs
Set up job 2s
Checkout 1s
Secrets 0s
Deploy to App Engine 5m 5s
Post Checkout 0s
Complete job 0

Pivotal Tracker

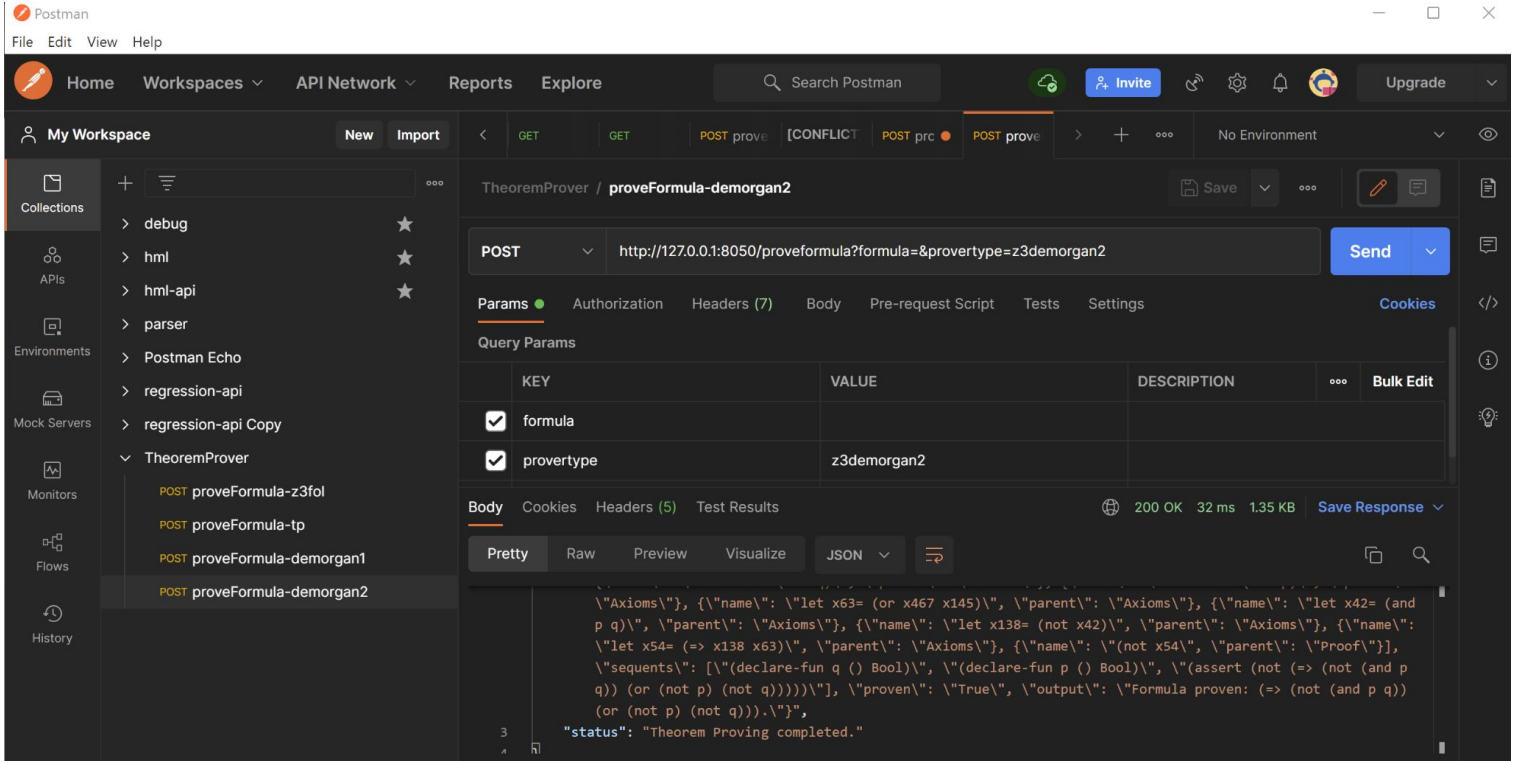


Swagger (OpenAPI) standard

https://theorem-prover-4182022.uc.r.appspot.com/api/docs



Postman API Testing



Simple Theorem Prover : Additional theory

1) Define class for Variables/symbols

2) define Functions/methods to Instantiate:

"Not", "And", "Or", "Implies", "ForAll", "ThereExists"

3) define keywords, to separate tokens, identify tokens

Valid tokens: ['not', 'implies', 'and', 'or', 'forall', 'exists']

4) Parse and typecheck (we need to know if types are valid)

5) Substitution: to substitute ground terms (such as Alice, Bob)

3) First each of function/operation, create a Sequent

Note: A sequent is a conditional or unconditional assertion.

4) Unify each sequent. -> 3 unification strategies:

a) Does sequent consist of a list? then unify list

b) Find all unifiable pairs

What we mean by unification?

f(g(x, b), f(x, z)) = f(y, f(g(a, b), c)) can be written as

x = g(g(a, b), b), y = g(a, b), z = c

For more in-depth reading: please do, refer this: https://www.cs.le.ac.uk/events/mgs2009/courses/struth/slides.pdf.

I would be glad to discuss further.

Unification pseudo code: Naive version

Goal: Identify two symbolic expressions.

Method: Replace certain subexpressions (variables) by other expressions

Refer: Syntactic Unification for Inferential Logic.

https://github.com/aimacode/aima-pseudocode/blob/master/md/Unify.md

Credits

Thank to Sagar for help with Google Cloud.

Thank to Saumya for sharing the guidelines.

Thank you Brian and Greg for allowing me to work on this project.