5-1-2019

INTELLIGENT PROBLEM SOLVER FOR DISCRETE STRUCTURES

Mohammed Ali

INTELLIGENT PROBLEM SOLVER FOR DISCRETE STRUCTURES

by

MOHAMMED ALI

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN SOFTWARE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2019

ACKNOWLEDGMENTS

Firstly, I would like to thank Dr. Conly, my senior project mentor, who has been an excellent guide for me throughout the development of the project. Next, I would like to thank my team, The Machine Team, for sticking with me through thick and thin over the hardships and successes we have undergone during the project. I would also like to thank the Honors College advisors/staff who were available and willing to help me all the time. They helped me understand every step that I had to do to complete my Honors degree in time.

Finally, I want to thank my family for supporting and believing in me. Thank you so much for the blessings you have showered on me during my time in college. I would not have gotten so far without you all looking after me and having my back even though we were thousands of miles away.

May 03, 2019

ABSTRACT

INTELLIGENT PROBLEM SOLVER FOR

DISCRETE STRUCTURES

Mohammed Ali, BS Software Engineering

The University of Texas at Arlington, 2019

Faculty Mentor: Christopher Conly

Online calculators have been on rise in the past few years. An online calculator for discrete mathematics is almost impossible to find. There are few online calculators that take up this challenge, but they have very limited functionality. This problem is mainly because discrete math problems have their own set of rules that apply only to that problem. We wrote a modified forward chaining algorithm on Wolfram Programming Lab that can solve discrete mathematics problems. We also created a beautiful front-end using React and Materials-UI framework. An application program interface (API) will be used to connect the backend with the front-end. The problem that the user types will be put in proper format and sent to the backend to be processed. The step-by-step solution is then returned that is displayed to the user.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Vision

The vision is to research and apply Knowledge Representation and Inference methods in Artificial Intelligence in order to design an intelligent educational web application that assists students in studying Discrete Structures. Through this project, the group aims to make scientific contributions about Knowledge Representation Methods.

1.2 Mission

The mission by the end of the project is to allow students to search for knowledge (search by the name of a specific definition, theory, or formula; search by a keyword or topic for relevant knowledge). Students can see a step-by-step solution for basic Discrete Structures problems. The intelligent problem solver lets the students define a problem (inputs with question) in a structured query. Given a defined problem, the system can provide students with approaches and step-by-step solutions. Overall, it will assist students in solving some harder discrete structures problems.

1.3 Background

Up to now, there have been few education sites that assist students with their academic studies such as Chegg, Khan Academy, Wolfram Alpha, and so on. However, all of them still have a limitation, they have real tutors supporting the students whenever they have a random question which is not listed in any solution manual. Inevitably, this reality causes some disadvantages:

- Increase in demand for tutors.
- Difficulty validating the tutor's experience.
- Difficulty updating new knowledge with all that many tutors.

In order to reduce the number of online supports for students, there are two ways currently applied in business:

- Introducing step-by-step solutions for predefined academic problems.
- Applying fees for real tutor support.

Keeping this in mind, there exists an increasing need for a system that can automatically assist the student without real tutors' expertise. This situation motivates an application that is capable of:

- Allowing the students to type in the prompt of a problem using a query language.
- Generating a step-by-step solution for the problem without human intervention.

1.4 Related Work

There have been few applications that provide step-by-step solutions for academic problems. Some Problem-Solving Engine service providers are Wolfram Alpha, QuickMath, CyMath, and so on. Until now, these Problem-Solving Engines only cover a limited amount of basic math contents such as functions, integration, etc. and are not capable of solving complex math problems or problems in other fields such as Chemistry, Physics, etc. The following are examples of the basic Problem-Solving Engines:

- Wolfram Alpha: http://www.wolframalpha.com/examples/pro-features/step-by-step-solutions/stepby-step-discrete-mathematics/
- CyMath: https://www.cymath.com/
- QuickMath: https://quickmath.com/webMathematica3/quickmath/equations/solve

In terms of research papers, there have been a few researches in Intelligent Problem Solvers in Education (IPSE). One of them has been published in the following chapter: Nhon Van Do (March 2nd, 2012). Intelligent Problem Solvers in Education: Design Method and Applications, Intelligent Systems Vladimir Mikhailovich Koleshko, IntechOpen, DOI: 10.5772/37115, which is available from: https://www.intechopen.com/books/intelligent-systems/intelligent-problem-solversin-education-design-method-and-applications

This chapter has demonstrated the design process of an IPSE using the Ontology of The Computational Object Knowledge Base model (COKB) discovered by Nhon Van Do. However, there have not been any major web or mobile applications that apply this new knowledge model. Therefore, the team aims to create an application to solve Discrete Structures problems using the COKB model.

1.5 Approach

The team used Agile software development methodology as shown in Figure 1.1 for our project. The tasks that need to be accomplished were specified in the initial sprint meeting. Following the sprint meeting, the team members work on designing, developing, testing, and deploying the component that was decided in the sprint meeting. At the end of the sprint the team released the newly built component. This same process was repeated all over again for every component of the project.

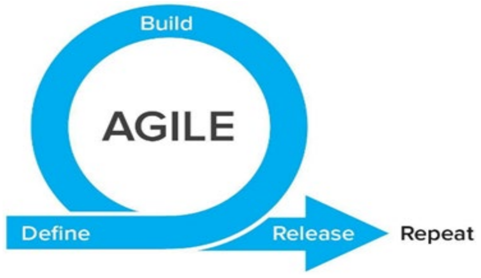


Figure 1.1: Agile Software Development Methodology

CHAPTER 2

REQUIREMENTS

The requirements stage involved the group members getting together and discussing the project to get a sense of what path the project should take. Our senior project mentor's opinion was helpful during this process.

2.1 Customer Requirements

This section contains requirements generated from discussion among the team members. These requirements are highly critical and are the indication of the functionality that the user accepts. None of these requirements shall be changed without mutual agreements among the team members.

2.1.1 Web Based GUI

The product shall have a Graphical User Interface for Users to interact with the display. The GUI will allow user to input Discrete Structure problem as text or upload picture of the problem. After the engine solves the problem, the user shall be able to view the step-by-step solution on the webpage.

Priority – Critical

2.1.2 Input Field for Problem

The product's GUI shall have an Input Field for the user to define a problem. The input field shall take input in plain text or shall take the uploaded picture of the problem as an input. Nonetheless, all the input problem must be related to Discrete Structure, otherwise, the input shall be invalid.

Priority – Critical

2.1.3 Solution Delivery to the User

The product shall deliver a step-by-step solution to the user defined valid Discrete Structure problem. The solution will be structured in the manner of Step 1, Step 2, Step 3, and so on.

Priority – Critical

2.1.4 Step Elaboration

The product shall allow a user to click a step on displayed solution, which will than cause the engine to retrieve information/topics/knowledge related to that step of solution and present it to the user. This further elaboration is aimed to allow the user to better understand as to how the problem is solved.

Priority – High

2.2 Packaging Requirements

Packaging requirements are those requirements that identify how the delivered product will be packaged for delivery to the end-user. In other words, it defines how the final product will "look" when finished and delivered. This product will be packaged into a web app also known as a website that users will be able to access through a web browser. No download will be necessary.

2.2.1 Web Application Requirements

There must be a web app hosted on some web server that the user can access through his/her web browser.

Priority – High

2.2.2 Web Server Requirements

There must be a web server that the web app can be hosted on so that users can access the website on the Internet.

Priority – High

<u>2.3 Performance Requirements</u>

The Performance of the project is measured by how robust the program is. It is also determined by the number of complex discrete structures problems that are successfully solved.

2.3.1 Processing

The software should take in the user input, process the information provided by the user and successfully give the step-by-step solution to every problem.

Priority – Critical

<u>2.4 Maintenance and Support Requirements</u>

2.4.1 Web Server Maintainability Requirement

Web server must be maintained and monitored daily to ensure the website is online. Unexpected crashes may occur, either because of issues with the web app itself, or the server host provider. Having the web server be monitored increases stability of the web site and ensures live access at any time for users.

Priority – High

2.4.2 Problem Expansion Requirement

Types of problem that can be solved through this web app should be expanded to user's needs over time. This will help keep a constant flow of users for the website.

Priority – Critical

CHAPTER 3

FEATURES AND FUNCTIONS

Based on the requirements that the team came up with, we decided that the web application should have the following features and functions:

- The user shall be able to submit problems and view solution.
- The user shall only define a problem related to Discrete Mathematics.
- Any input problems not related to Discrete Mathematics will be deemed invalid by the system.
- The user shall either define problem on the input field in the form of text or upload a clear picture of the problem.
- After receiving a valid input, the system shall compute and provide a detailed step-by- step solution to the user.
- All the steps shall be numbered using alpha-numeric texts, for e.g. Step 1, Step 2 and so on.
- If the user wishes, he/she can click on a step number and the system should retrieve specific knowledge/ topic related to that step and present it to the user as extended elaboration.

.

CHAPTER 4

ARCHITECTURAL DESIGN

Figure 4.1 is the overall structure of our system. As shown below, the system will have four components, each with specific set of functions and interaction with the other components. The dataflow and interactions among the components can be seen below, which will later be defined in detail. The User and the Knowledge-Engineer are not a component of the system but rather actors of the system.

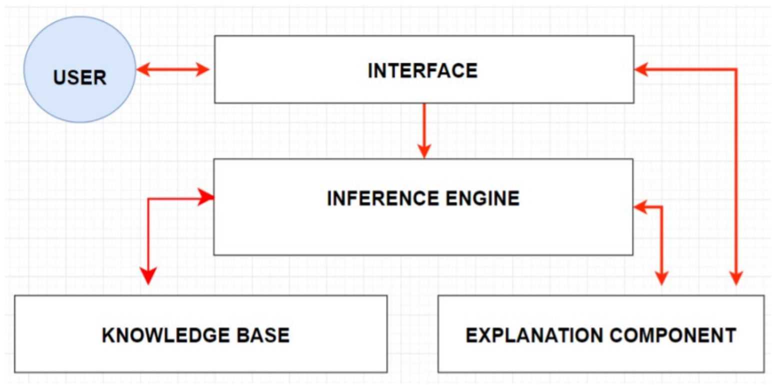


Figure 4.1: Basic Design of the System

4.1 Interface

It is the primary point of contact between the IPS (Intelligent Problem Solver) and the user. This component controls the GUI that is responsible for getting the problem specification (input) from the user, sending it to the server (Inference Engine / Explanation

Component) and displaying the result back from the server (Inference Engine/ Explanation component) in a step-by-step solution.

4.2 Knowledge Base

This component is in-charge for storing the knowledge (Classes, Functions Concepts, Rules, Operators, etc.) in structured files that the system can retrieve and process. It is the core component of the system that provides the information necessary for the Inference Engine to solve the user input problem using an established theory or theorem.

4.3 Inference Engine

This component can be considered as the brain of the system. It is where the forward chaining algorithm is implemented to generate new facts from the knowledge base starting with the known facts and inference rules. Specification from the user is sent to the Inference Engine which then consults with the Knowledge Base and implements forward chaining to generate the output. The output is then tunneled to the Explanation Component.

4.4 Explanation Component

End User expects a step-by-step solution rather than a single line of result. Thus, the raw output from the Inference Engine is sent to the Explanation Component, where the Explanation engine makes sense of how the problem was solved in a stepwise manner. It also formats each step in human readable form and sends it to the Interface Component to display to the user.

CHAPTER 5

DETAILED DESIGN

The following is the detailed design of our system. The lines show the interaction between the systems and subsystems. The numbering alongside the lines shows the order in which these interactions takes place.

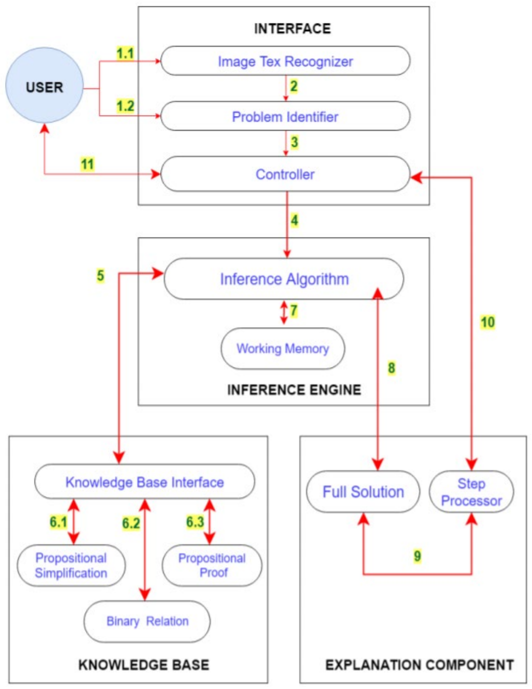


Figure 5.1: Detailed Design of the System

5.1 Inference Engine Layer

5.1.1 Inference Algorithm Subsystem

The inference engine system is based on algorithms and storing the results. This subsystem of Inference Layer basically uses a modification of forward chaining algorithm to calculate the final result of the user provided problem with the help of the knowledge base.

5.1.1.1 Subsystem Software Dependencies

This subsystem is using the Wolfram lab API for the algorithm. The way it works is that the algorithm is written on Wolfram's servers by the team. An API call is made in the web application to access the written algorithm. The algorithm will run through a sequence of steps and when it finds the solution returns the solution.

5.1.1.2 Subsystem Programming Languages

The program language used for this subsystem is Wolfram's programming language.

5.1.1.3 Subsystem Data Processing

The algorithm that the team developed is based on the forward chaining algorithm used heavily in Artificial Intelligence (AI) based fields. It processes the problems and comes up with the intermediate results and eventually the final result.

5.1.2 Working Memory Subsystem

This subsystem of the Inference Layer basically stores the intermediate results.

5.1.2.1 Subsystem Software Dependencies

This subsystem uses Google's Firebase, a powerful document-based database to store data and be able to retrieve it quickly and efficiently.

5.1.2.2 Subsystem Programming Languages

Although we are using a database, this database is a noSQL database. Therefore, SQL is not being used as a language. The data is accessed or queried, through JavaScript.

5.1.2.3 Subsystem Data Structures

The data structures being used are document based, unlike the typical relational model used in most SQL databases. Since Firebase is a document-based database, it does not use any kind of relational model. The document-based model is similar to JSON, so all the data structures utilized in Firebase will follow the JSON format.

5.1.2.4 Subsystem Data Processing

The only data processing that takes place in this subsystem is taking the steps that are generated and storing them in the actual database. The data is not analyzed and processed at an extreme level, but it is looked at and retrieved frequently.

5.2 Explanation Layer

5.2.1 Full Solution Window and Step Processor Subsystem

The full solution is the entire solution to the user's problem. While, the full solution just stores the whole solution, the step processor links to the full solution and divides it into steps. In other words, the step processor divides the whole solution so that it makes sense to the user.

5.2.1.1 Subsystem Software Dependencies

This subsystem uses Wolfram lab API, a powerful mathematical processor to be able to retrieve and compute the solutions quickly and efficiently.

5.2.1.2 Subsystem Programming Languages

This processor uses the Wolfram Alpha programming language. The way the data is accessed or queried, is through JavaScript.

5.2.1.3 Subsystem Data Structures

The document-based model for each solution is JSON, so all the data structures queried from the Wolfram API will follow the JSON format.

5.2.1.4 Subsystem Data Processing

The only data processing that could be considered as data processing is taking the intermediate steps that we generate from the wolfram API and storing it.

5.3 Interface Layer

5.3.1 Image Recognizer Subsystem

This subsystem takes a picture uploaded by the user and identifies the text within that picture. The text will be letters and symbols that make up discrete mathematics/logic.

5.3.1.1 Subsystem Software Dependencies

Latex viewer which recognizes latex and a Mathpix API which helps with text recognition (in the picture uploaded by the user) is used.

5.3.1.2 Subsystem Programming Languages

JavaScript is used to make calls to the APIs

5.3.1.3 Subsystem Data Structures

The results from the API calls are returned as JSON objects. As a key-value pair, each property of the data can be viewed and checked to see what the text was and if the recognition worked properly.

5.3.1.4 Subsystem Data Processing

The images need to be processed to Latex.

5.3.2 Problem Identifier Subsystem

The Problem Identifier is the subsystem that has the power to recognize what the actual problem is from the user provided text (from an image or from the user's keyboard entry). More specifically, it will recognize the problem (the discrete math problem the user wants to solve) and separates the hypothesis and goal. It also makes sure the input is in the correct format.

5.3.2.1 Subsystem Programming Languages

JavaScript is used to put together the user input to verify if the user input is a problem. In addition, JavaScript is also used to check if the problem is solvable and in the correct format.

5.3.2.2 Subsystem Data Structures

JSON is an object format for the data structures used in this subcomponent.

5.5.6 Subsystem Data Processing

Data processing is heavily used in this subsystem. The results from the Image Text recognizer or the user keyboard input is translated into a format usable by our backend program.

5.3.3 Controller Subsystem

The Controller sub-component of the Interface will assist in control of the Interface and allow user's input to reach the backend.

5.3.3.1 Subsystem Programming Languages

JavaScript is used to oversee the other two subcomponents (Problem identifier subsystem and Image Text Recognizer Subsystem).

5.4 Knowledge Base Layer

5.4.1 Knowledge Domain Subsystem

The Knowledge Base stores all the inference rules of all the knowledge domains in Discrete Structures. We can utilize these rules stored whenever a problem is given, instead of calculating it for each individual problem. This makes the application much quicker as it is simply accessing rules and applying them rather than starting all the way from scratch.

5.4.1.1 Subsystem Software Dependencies

The dependencies include Google's Firebase, and Wolfram.

5.4.1.2 Subsystem Data Structures

Document based data. For example, in JSON, there are key-value pairs, which is how the data is arranged in this database. When retrieving results, they can be saved as JSON objects which can be parsed out easily. JSON format is preferred as it is standard for transferring data and very compatible with the application.

5.4.1.3 Subsystem Data Processing

Data processing will include the processing of the results that the database (or the API call to the database) returns. As the format is expected to be JSON, the processing will mostly comprise of recognizing key value pairs and saving them to the corresponding variables. The data stored in the variables is then used by the Inference Engine to determine the next step to get to the solution.

CHAPTER 6

SYSTEM INNER WORKING

The team will be using Wolfram Programming lab for our backend where we will be storing the algorithm to solve the discrete mathematics problems. It also stores all the knowledge base and the steps that are to be taken to solve the problem. The algorithm being deployed to solve the discrete mathematics problem is a modified form of a forward-chaining algorithm. The algorithm searches for a rule in the knowledge base considering the facts provided by the user. If there is some sort of rule present that simplifies the problem, the rule is applied, and the result is stored in the solution steps. Storing the steps makes sure that the algorithm can avoid any steps that initiate a loop during the problem-solving. A rule when applied resulting in one of the solutions already present in previous steps means that there is a possibility of a loop. Therefore, the algorithm avoids such pitfalls and makes sure there are no infinite loops that deter the algorithm from solving the problem. If the algorithm does not find any rule that can be applied further to get to the final goal, the system displays could not solve the problem message. If the algorithm reaches the final goal while applying the rules, the final step is stored by the explanation component and then the step-by-step solution is displayed to the user.

The front-end will be made using React and open source material-UI framework. An application program interface (API) will be used to connect the backend with the frontend. The problem that the user types will be put in proper format by the front and sent to the backend to be processed. The step by step solution is then returned which is displayed to the user

CHAPTER 7

WEB APPLICATION FRONT-END

Since the author worked more on the front-end, the following is the description as to how the web application can be used by the user. Figure 7.1 is the current home page of the system. The user can type in the problem in the text box or select the Propositional Logic or the Binary Relations button.

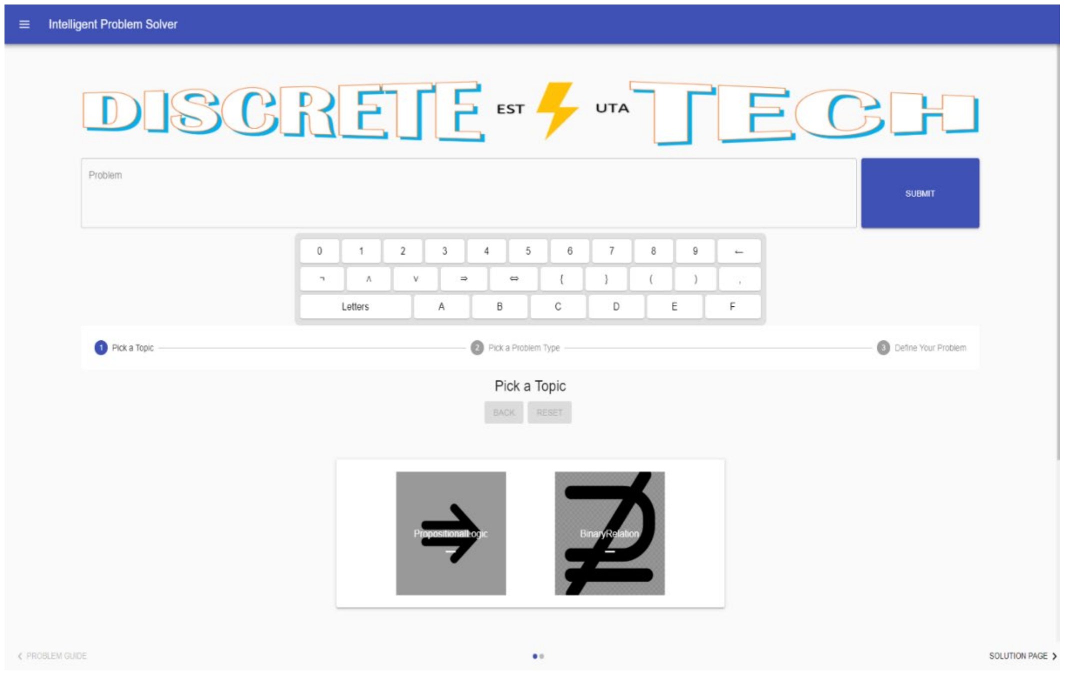
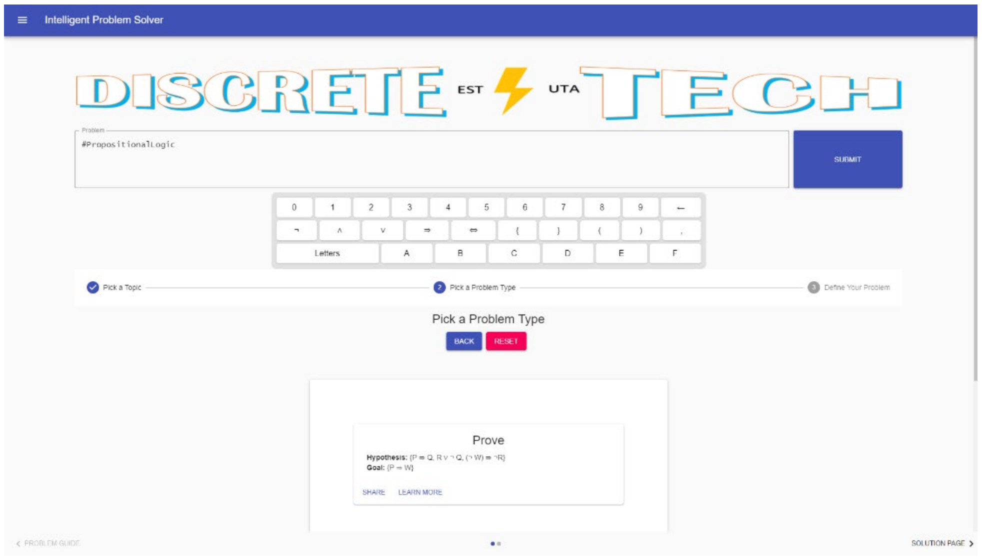


Figure 7.1: Home Page of the Web Application

Once the user selects the Propositional Logic, the user views the page shown in Figure 7.2



```
#PropositionalLogic
```

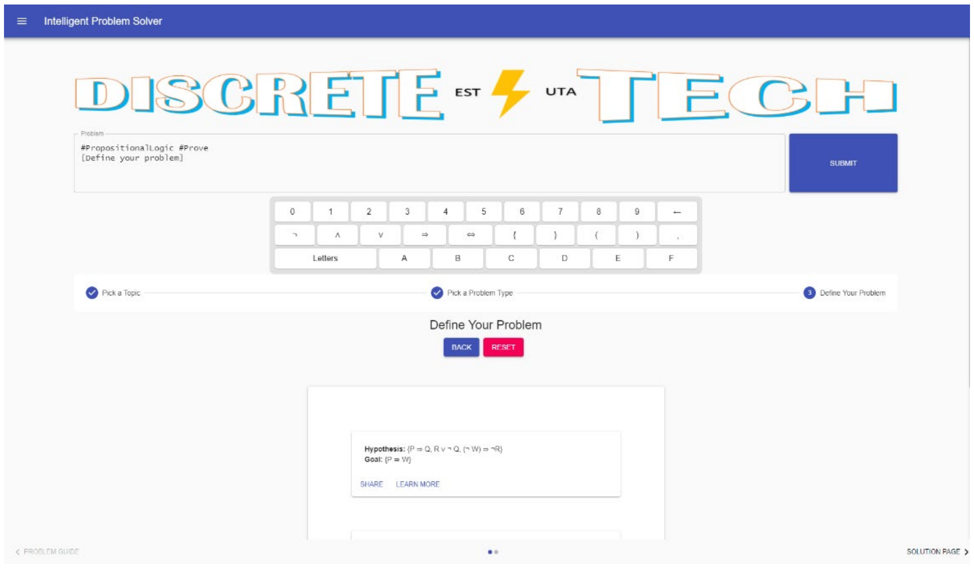
Pick a Problem Type

Prove

Hypothesis: {P ⇒ Q, R ∨ ¬ Q, (¬ W) ⇒ ¬R}
Goal: {P ⇒ W}

Figure 7.2: Home/PropositionalLogic

The user currently has an option of selecting the proof option for solving propositional logic. Currently, the web application only supports proof problems for propositional logic. In the future, the team plans to work on adding more type of problems. Once the user selects proof, the user views the page shown in Figure 7.3. The user can see few sample problems which the user can select to solve or type in their own problem.



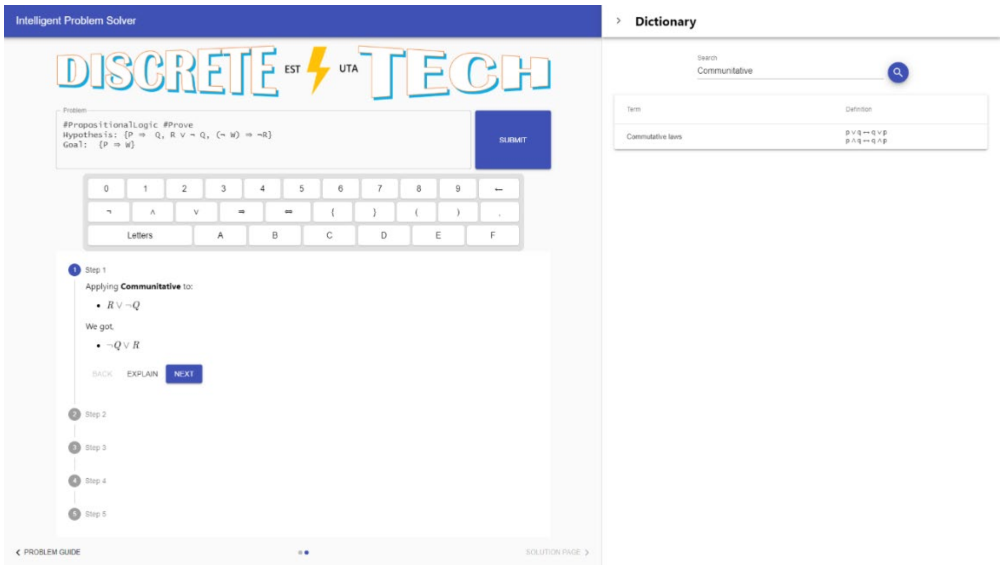
```
#PropositionalLogic #Prove
[Define your problem]
```

Define Your Problem

Hypothesis: {P ⇒ Q, R ∨ ¬ Q, (¬ W) ⇒ ¬R}
Goal: {P ⇒ W}

Figure 7.3: Home/PropositionalLogic/Proof

If the user selects any of the sample problems, the problem input textbox is filled with the sample problem. On the other hand, the user can ignore the sample problems and type in his own problem. Once the problem is typed in the textbox and the user presses the submit button, the user sees the step-by-step solution as shown in Figure 7.4. The user can also press the "Explain" button which opens the dictionary side drawer.



```
#PropositionalLogic #Prove
Hypothesis: {P ⇒  Q, R ∨ ¬ Q, (¬ W) ⇒ ¬R}
Goal:  {P ⇒ W}
```

Step 1

Applying **Communitative** to:

- $R \vee \neg Q$

We got,

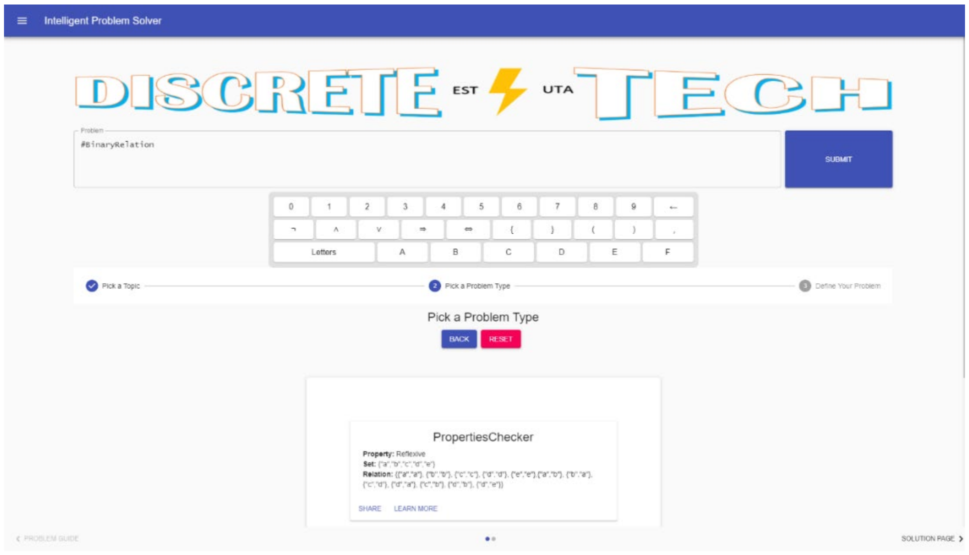
- $\neg Q \vee R$

Communitative

Term	Definition
Commutative laws	$p \vee q \leftrightarrow q \vee p$ $p \wedge q \leftrightarrow q \wedge p$

Figure 7.4: Home/PropositionalLogic/Proof/Solution

If the user selects Binary Relations instead of Propositional Logic in Figure 7.1, the user views the page shown in Figure 7.5. Currently for Binary relations, the web application can figure out the properties of a binary relations (Is the relation Reflexive, Symmetric, Antisymmetric or Transitive). Again, here the user has an option to either type in their own problem or select one of the sample problems to solve.



```
#BinaryRelation
```

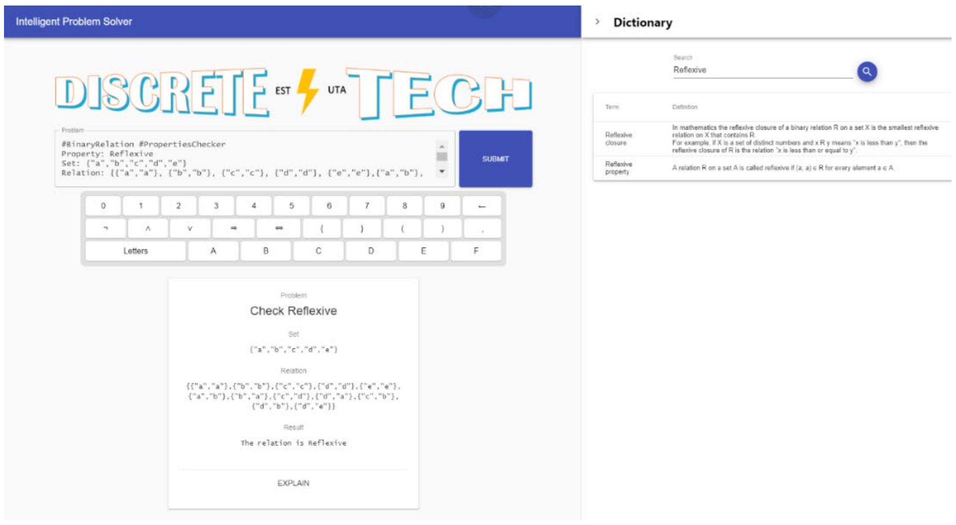
Pick a Problem Type

PropertiesChecker

Property: Reflexive
Set: {"a","b","c","d","e"}
Relation: {{"a","a"}, {"b","b"}, {"c","c"}, {"d","d"}, {"e","e"},{"a","b"}, {"b","a"}, {"c","d"}, {"d","a"}, {"c","b"}, {"d","b"}, {"d","e"}}

Figure 7.5: Home/BinaryRelations

Once the user enters the binary relation and presses submit the user views the solution in the format shown in Figure 7.6. The user can press explain to see the dictionary definitions of the topics/rules the algorithm used to solve the problem. Note: The user can search any topic in the dictionary whenever the user wants to.



```
#BinaryRelation #PropertiesChecker
Property: Reflexive
Set: {"a","b","c","d","e"}
Relation: {{"a","a"}, {"b","b"}, {"c","c"}, {"d","d"}, {"e","e"},{"a","b"},
```

Problem
Check Reflexive

Set
{"a","b","c","d","e"}

Relation
{{"a","a"},{"b","b"},{"c","c"},{"d","d"},{"e","e"},{"a","b"},{"b","a"},{"c","d"},{"d","a"},{"c","b"},{"d","b"},{"d","e"}}

Result
The relation is Reflexive

Reflexive

Term	Definition
Reflexive closure	In mathematics the reflexive closure of a binary relation R on a set X is the smallest reflexive relation on X that contains R. For example, if X is a set of distinct numbers and x R y means "x is less than y", then the reflexive closure of R is the relation "x is less than or equal to y".
Reflexive property	A relation R on a set A is called reflexive if (a, a) ∈ R for every element a ∈ A.

Figure 7.6: Home/BinaryRelations/Solution

CHAPTER 8

CONCLUSION

The system components efficiently interact internally with each other to find the solution for the user provided question. The most important component of the system, the Inference Engine, is successful in figuring out the answer to the user provided Discrete Structures problems. It also works seamlessly with the front-end to display the step-by-step solution to the user in a beautiful and interactive manner. Overall, our system fulfills our vision to design an intelligent educational web application that assists students in studying Discrete Structures.

REFERENCES

Ben-Arieh, D., & Moodie, C. L. (1987). Knowledge based routing and sequencing for discrete part production. *Journal of Manufacturing Systems*, *6*(4), 287-297

Liu, S., and Bobrow, J. E., "An Analysis of a Pneumatic Servo System and Its Application to a Computer-Controlled Robot," *ASME Journal of Dynamic Systems, Measurement, and Control*, 1988, Vol 110 pp 228-235.

Nhon Van Do (March 2nd, 2012). Intelligent Problem Solvers in Education: Design Method and Applications, Intelligent Systems Vladimir Mikhailovich Koleshko, IntechOpen, DOI: 10.5772/37115. Available from: https://www.intechopen.com/books/intelligent-systems/intelligent-problem-solversin-education-design-method-and-applications

Zeigler, B. P. (1987). Hierarchical, modular discrete-event modelling in an object-oriented environment. *Simulation*, *49*(5), 219-230.

BIOGRAPHICAL INFORMATION

Mohammed Ali was an international student at the University of Texas Arlington (UTA). He enrolled in UTA in Fall 2015 where he pursued an Honors Bachelor of Science in Software Engineering. After graduating in the Spring 2019, he will focus on doing his graduate degree in Computer Science with a focus in software security. During his time as an undergraduate student at UTA, he was extensively involved in research projects. He plans to continue being involved in research projects during his graduate studies.