CSE 579: Knowledge Representation Portfolio

Shreyas Chandrashekhar Kirtane
ASU ID: 1225453736
skirtan1@asu.edu

***Abstract*—Knowledge representation is a field in artificial intelligence that deals with how knowledge can be stored, structured and manipulated within a computer system with a goal to develop formalisms and methodologies to encode information in a way that makes it useful for reasoning, decision-making and problem solving tasks. This report is a portfolio for the course project for CSE 579, Knowledge representation course. This report gives detailed solution to the automated warehousing problem developed as the course project using concepts from Answer set programming using clingo. Finally, the contributions made as a part of this project and the lessons learned are described**

***Index Terms*—artificial intelligence, reasoning, answer set programming, clingo**

I. Introduction

ASP or Answer set programming is a declarative programming paradigm used to solve combinatorial search problems. In ASP, a problem is described using rules and facts. It extends traditional logic programming models by allowing non-monotonic reasoning. It works by finding stable models for a problem. A stable model is a solution to which remains consistent to the facts and rules described by the problem. This core idea of stable model allows for non trivial reasoning deductions using logic programming [1]. The automated warehousing problem is a problem modelled to automate warehouse operations using flat robots. This problem is taken from ASP challenge 2019. The problem consists of a warehouse represented by a grid where flat robots are placed. There are shelves placed throughout the grid containing various products. There are also various picking stations where orders are made. The robots must move throughout the grid picking up shelves which have products required by the order in desired quantities and deliver it to the picking station of the order. A robot can only pickup one shelf at a time and cannot move through nodes which house other shelves. Such constraints are placed which model the physical world. Robot collisions are not allowed. Furthermore, there are nodes designated as highways, where the robots cannot put down a shelf. Finally, it is desired that a plan must be found which details the movement and actions of the robots which satisfy the constraints and fulfill the orders. Multiple robots are allowed to perform actions at a time. The minimum time step which satisfies all the orders is required to be found by the solution of the problem. This is a provable non deterministic polynomial problem.

This is where ASP shines. ASP has been widely used in evaluating NP-Hard problems. The knowledge representation language of ASP is very expressive in a precise mathematical sense; in its general form, allowing for disjunction in rule heads and nonmonotonic negation in rule bodies, ASP can represent every problem in the complexity class Σ_2^P and Π_2^P (under brave and cautious reasoning, respectively) [2]. Thus, ASP is strictly more powerful than SAT-based programming, as it allows us to solve problems which cannot be translated to SAT in polynomial time [3]. Problems in diagnosis and planning without complete knowledge like the automated warehousing problem fall in Σ_2^P and Π_2^P classes, [3]–[5]. Problems like automated warehousing problem can be encoded in ASP [3], [6], [7]. Clingo is a solver that is used to evaluate the ASP programs and find stable models. ASP and clingo will be utilised to solve the automated warehousing problem.

II. Description of Solution

The project consisted of 3 milestones. The first two milestones consisted of ASP problems with varying difficulties intended for practise and getting familiar with clingo and asp programming.

This project is based on concepts combining reasoning with that of actions. Actions are usually described as functions which change the state of an environment. The majority of activities are deterministic and alter the state of the world. It is possible to express an action's consequences and prerequisites quantitatively by using reasoning with actions. This can assist in diagnosing and debugging the issues as well as determining the viability of our solution. The worlds can be in exponential states. The states of the world are called fluents. The states of fluents can be changed using actions. This combination of fluents and actions can be represented as a graph also known as a finite state machine. The nodes of the graph represent the states. And the edges represent transition between states which can be caused by actions.

This theory however, remains incomplete without inclusion of the common sense law of inertia. It states that in absence of any actions the states of the fluent must remain the same as the previous state.

To solve the automated warehousing problem in ASP, the program is structured into different components, which include declaration of the objects, constraints related to the state of fluents, preconditions; effects and constraints of actions and domain independent axioms. The following subsections describe these components in detail.

A. Sort and Object declaration

This section of the asp program is used to describe the world of the problem. This section is also used to extract object declarations from the init statements provided as part

of the sample test cases with the challenge problem. We have been provided with 5 sample input cases which can be used to test the solution to the problem. The objects in the automated warehouse problem include objects like: picking station(P), product(I), order(O), robot(R), shelf(S) and highway (C). In addition, we also create an object called node (N), which converts the grid location described by X and Y coordinates into a one dimensional structure. We also define state of these objects or fluents. These definitions include *nodeLocation, robotAt, shelfAt, productCount and productRequirement*. *nodeLocation* is an object used to convey the conversion between node Id N and its location (X, Y) on the grid. *robotAt* and *shelfAt* represent the locations of robots and shelf at a given time. The shelf can be on a node or a robot, but not on both at the same time. *productCount* is used to state the product and its count on a shelf at a given time. *productRequirement* is used to represent the number and the product required for a particular order at a given time.

Finally, the end state of the system is described. All orders must be fulfilled at the final time step. This means that the product requirement of any product for any order must not be greater than 0. We state this notion using constraints. They help the clingo solver to ignore unwanted solution branches. All solutions must satisfy the constraint to be a valid solution.

B. State constraints

State constraints are used to model the natural intuition and limits of the physical world. We don't want to allow solutions which violate the rules of the physical world to be selected. For example, robot are limited to picking up only a single shelf at a time. Since these robots are flat when they are empty. Similarly, two robots cannot be at the same node at a given time. Similarly, a shelf cannot be on a node and a robot at the same time. A shelf contains an assortment of products, however it can only contain a single value for a particular type of product, i.e same product cannot have two different quantities on a single shelf. The same is also true for product requirement as well.

C. Preconditions, effects and constraints of actions

This section of the asp program describes the various actions that can affect the state of fluents in the problem world. Each of the actions have some constraints, affect the fluent in some way and require some conditions to be true in order for the action to occur.

Movement is an action that can be performed by the robots, it describes the robot, the node at which the robot ends up at after moving and the final time. There are constraints to these action such that two robots cannot end up at the same node at the end of a time step. Also robots are only allowed to move horizontally and vertically 1 unit in a time step such that the Manhattan distance between the previous location and the new location is 1. Also the robots cannot swap location in a time step.

The action for picking up a shelf is called shelfPickup, it describes the shelf, robot and the timestamp at the end of the action. After performing this action the shelf is picked up by the robot. Since this action affects the state of two fluents, there are preconditions to both of these fluents which must be satisfied before this action can be performed. For the shelf, the shelf must not be on a node at time T-1. As for the robot the robot must not be carrying a shelf at time T-1 for it to pickup a shelf. Similar action is described for putting down a shelf as shelfPutdown. For this action the preconditions involve the shelf being on a robot and the robot not being on a highway node for it to put down the shelf on a node. After putdown action the effective location of a shelf doesn't change with that of a robot.

The action corresponding to delivering products from shelf to picking station of an order is called fulfillOrder. It is associated with the robot, order, product and includes the quantity of product being delivered and the time stamp. For delivering a product the condition that the robot must be at the same location as that of the picking station must be satisfied. The shelf must have the product being delivered and the quantity being delivered must be available with the shelf with the robot. The robot cannot deliver a product not required by the the order, or a product not available with the shelf. After this action the amount of the product delivered required by the order reduces by the quantity of product delivered by the shelf. The amount of product available with the shelf also reduces.

D. Domain independent axioms

Domain independent axioms are used to describe the general properties and behaviours of the domain, which are true regardless of the specific problem instance. These axioms are true regardless of the particular interpretation or assignment of values to variable within the domain. These axioms are used in encoding common-sense reasoning and domain knowledge into ASP systems. This helps with weeding out unwanted results and ensure that the ASP follows common rules of physics and human intuition. It is ensured that the initial state of the fluents are exogenous or out of control of the solver. Furthermore, the rules which state the fluents must have unique values, i.e fluent can only have one state at a time, are also described under domain independent axioms. The second axioms is that actions are exogenous. This helps the system in taking all possible combinations of actions at all the time stamps depending on which branches do not break the constraints of the system. Laws related to the common sense inertia are also encoded in domain independent axioms. For example, for a fluent if any action which can change it's state are not performed at a time step, the state of the fluent must remain same in the next time step. For non-static fluents, if it keeps on changing state, unless an action which can stop the non-stable behaviours is performed. This is consistent with the laws of motion in physics.

E. Optimizing and output directive

Clingo returns all the objects in the world which satisfy the constraint at the final time step. However, it is necessary to limit the output to only display required information which

was obtained using the reasoner. Therefore the actions are encoded into a single occurs operation, which describes the action and its arguments and the time step it was performed at. Clingo will also return all the solutions which satisfy the constraint. To optimize the solution for a single parameter, such as minimizing or maximizing for a specific parameter, an optimization directive with the name of the parameter can be added. For, the automated warehousing problem we can add a minimizing optimizing directive on the time step parameter.Also to limit the displayed output, we add an output directive which displays only occurs statements for the solution. This provides us with an action plan with actions performed at each time step leading to the minimum time it took to complete the orders.

III. Results

Clingo programs can be run from the terminal using the clingo command. Since the initial state of the fluents are exogenous, to specify a specific scenario, an input file can be used to describe the initial states. To append multiple code files, clingo only requires the name of files in the order in which they should be appended. So to run we can run the solution file and input file together to get the solution for the scenario. We also need to assign a variable t which specifies the upper bound for the time step t. This is done to get added efficiency by limiting the search space. The higher the value of t, the more probability that a scenario will have a stable model solution. On the other hand, it is computationally expensive to have large enough values of t.
The challenge of the problem describes a sample test case which should give the result 13. The initial state of the problem is described in Fig 1. It consists of two robots and two picking stations with 6 shelves. In addition the challenge consists of 5

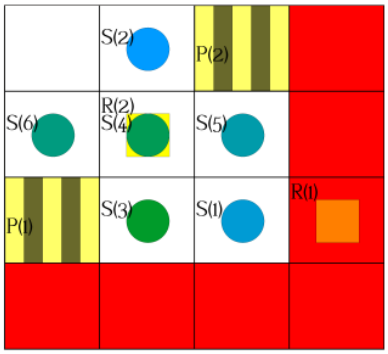

Fig. 1. Sample problem instance

other sample problems of different arrangements. The results obtained for these instances is encapsulated in table from fig 2. Furthermore, clingo has support for features like parallel processing, which can utilize multiple cores on a system efficiently to get the solution faster. As shown in the table the ASP program successfully finds the minimum time span for each of the sample instance files. The effectiveness of ASP at solving such problem is signified by ASP. With incomplete knowledge, it is able to deduce facts to improve performance while solving combinatorial search problems.

TABLE I
Test case results

Test instance	Time
inst1.asp	T=13
inst2.asp	T=13
inst3.asp	T=7
inst4.asp	T=10
inst5.asp	T=8

IV. My contributions

As this project was undertaken independently, I carried out all aspects of the implementation on my own. Throughout the process, I consulted the course material and tutorials online to improve my understanding of Knowledge reasoning formalism and ASP concepts. I learned various features of the clingo reasoner and completed this project in incremental steps:

1) I used the course material to learn how to express basic logic concepts in clingo.
2) I completed the First two milestones which helped me get familiar with optimization directives and other features of the clingo asp solver.
3) I tried re-implementing basic action tasks like monkey banana and blocks world in clingo to get a good understanding of action reasoning problems.
4) I perused through the project description and the ASP challenge.
5) I understood the requirements of the problem and the input and output format.
6) I started by converting the input into object definitions and creating objects which might be needed to solve the problem. I then defined the final desired state of the objects.
7) After defining the objects I implemented the state constraints and started working on actions. After defining the actions their preconditions and effects, I worked through implementing domain independent axioms.
8) After this I added the optimization directives and output directives.
9) At this point, I started debugging the program and fixing problems with the initial test case.
10) After correcting the bugs and constraints I tested the code with different test cases and performed runs several times to record the results.
11) At last I added comments and improved readability of the code.

V. Lessons learned

Working on the project has taught me the importance of Knowledge reasoning in intelligent systems. I have learned about the formalism used to represent knowledge in computer systems and extract meaningful reasoning from the knowledge base to accomplish a task. The initial milestones improved my skills with clingo programming. I understood how to describe

a problem statement in clingo and how clingo can be used to solve various combinatorial search problems and decision problems. I learned how it can be used as a scheduler and Sudoku solver

By working through the project, I learned how to model the problem in ASP. I learned how to break down the problem into simpler steps and executing the the project in a step wise manner. I got introduced to declarative programming paradigm which gave my a new perspective on how problems can be modelled and solved. I learned how to integrate different steps of the project correctly. I learned how ASP programs should be structured, so it is easy to read and be made sense of. I understood how real life constraints can be expressed in ASP and how useful the declarative programming approach can be to solve industrial problems.

This project has taught me the applications and importance of Knowledge representation and reasoning in artificial intelligence system. I also learned about new research trends in ASP like the intersection of neural networks and ASP, which can be used to extract meaningful reasoning from data which can be classified using neural networks. This highlighted that knowledge representation can be used to add depth to the other techniques in AI like machine learning. I also learned that using ASP can also increase training efficiency of machine learning model. This project has certainly increased my interest in Knowledge representation.

REFERENCES

[1] Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: Logic Programming: Proceedings Fifth Intl Conference and Symposium, Cambridge, Mass., MIT Press (1988) 1070–1080.

[2] Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM TODS 22, 364–418 (1997)

[3] Faber, W., Ricca, F. (2005). Solving Hard ASP Programs Efficiently. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds) Logic Programming and Nonmonotonic Reasoning. LPNMR 2005. Lecture Notes in Computer Science(), vol 3662. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11546207_19

[4] Rintanen, J.: Improvements to the Evaluation of Quantified Boolean Formulae. In: Dean, T. (ed.) IJCAI 1999, Sweden, pp. 1192–1197 (1999)

[5] Eiter, T., Gottlob, G.: The Complexity of Logic-Based Abduction. JACM 42, 3–42 (1995)

[6] Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. CUP (2002)Magnetics Japan, p. 301, 1982].

[7] Leone, N., Rosati, R., Scarcello, F.: Enhancing Answer Set Planning. In: IJCAI 2001 Workshop on Planning under Uncertainty and Incomplete Information, pp. 33–42 (2001)